ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА по курсу «Data Science»

Васильев Александр Иванович

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Начальные условия

Даны два датасета X_bp.xlsx и X_nup.xlsx.

Датасет X_bp.xlsx содержит 11 столбцов (10 признаков) и 1023 строки.

Датасет X_nup.xlsx содержит 4 столбца (3 признака) и 1040 строк.

Необходимо объединить датасеты по индексу (тип объединения INNER).

В результате объединения получен датасет с 13 признаками и 1023 строками.

Все признаки содержат значения float64, за исключением признака «Угол нашивки, град», у которого целочисленный тип int64, качественные характеристики отсутствуют, пропусков нет.

Признак «Угол нашивки, град» имеет два уникальных значения: 0 и 90.

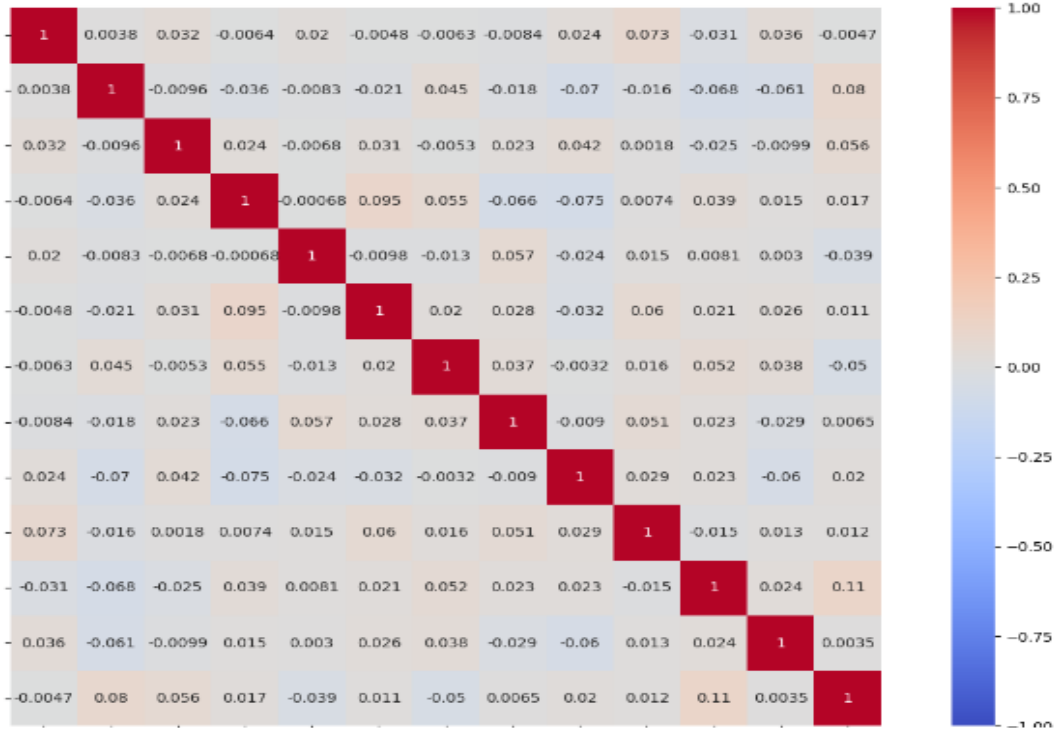
Данные в датасете представлены в разном масштабе.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разведочный анализ


Гистограмма распределения параметров.

Распределение величин близко к нормальному для большей части признаков, за исключением поверхностной плотности – большое смещением влево и угла нашивки – дискретная величина.




Тепловая карта коэффициентов корреляции.

Коэффициенты корреляции предварительно показывают, что явная зависимость между переменными датасета отсутствует.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Предобработка данных

```
le = LabelEncoder()
df['Угол нашивки, град'] = le.fit_transform(df['Угол нашивки, град'])
```

```
# Посмотрим, что получилось

df['Угол нашивки, град'].unique()
```

```
array([0, 1], dtype=int64)
```

Кодирование категориальных данных

В результате кодирования с помощью метода LabelEncoder() мы получили признак «Угол нашивки, град» со значениями 0 и 1.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Предобработка данных

```
Соотношение матрица-наполнитель , z-score:  0
Соотношение матрица-наполнитель , IQR:  6
Плотность, кг/м3 , z-score:  3
Плотность, кг/м3 , IQR:  9
модуль упругости, ГПа , z-score:  2
модуль упругости, ГПа , IQR:  2
Количество отвердителя, м.% , z-score:  2
Количество отвердителя, м.% , IQR:  14
Содержание эпоксидных групп,%_2 , z-score:  2
Содержание эпоксидных групп,%_2 , IQR:  2
Температура вспышки, С_2 , z-score:  3
Температура вспышки, С_2 , IQR:  8
Поверхностная плотность, г/м2 , z-score:  2
Поверхностная плотность, г/м2 , IQR:  2
Модуль упругости при растяжении, ГПа , z-score:  1
Модуль упругости при растяжении, ГПа , IQR:  6
Прочность при растяжении, МПа , z-score:  0
Прочность при растяжении, МПа , IQR:  11
Потребление смолы, г/м2 , z-score:  3
Потребление смолы, г/м2 , IQR:  8
Угол нашивки, град , z-score:  0
Угол нашивки, град , IQR:  0
Шаг нашивки , z-score:  0
Шаг нашивки , IQR:  4
Плотность нашивки , z-score:  7
Плотность нашивки , IQR:  21
Стандартизированная оценка (z-score), выбросов: 25
Межквартильное расстояние (IQR), выбросов: 93
```

```python
z_score_clean = pd.DataFrame(index = df.index)
for column in df:
    z = stats.zscore(df[column])
    z_score_clean[column] = z.abs() > 3
df_clean = df[z_score_clean.sum(axis=1)==0]

# Посмотрим форму очищенного датасета.
df_clean.shape
```

```
(999, 13)
```

Удаление выбросов.

Удалим выбросы методом z-score.

В результате удаления выбросов мы получили датасет из 13 признаков и 999 строк.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка и обучение модели ML

```python
# Целевой параметр - "Модуль упругости при растяжении, ГПа"
y1 = df_clean.iloc[:, df_clean.columns == 'Модуль упругости при растяжении, ГПа']

# Целевой параметр - "Прочность при растяжении, МПа"
y2 = df_clean.iloc[:, df_clean.columns == 'Прочность при растяжении, МПа']

# Входные параметры
X = df_clean.drop(columns=['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'])

# Посмотрим форму полученных данных.
print(y1.shape)
print(y2.shape)
print(X.shape)
```

```
(999, 1)
(999, 1)
(999, 11)
```

Проведём разбиение датасета на входные (X) и целевые (y) данные в двух вариантах, для каждого целевого параметра.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка и обучение модели ML

## «Модуль упругости при растяжении, ГПа»

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| LinearRegression | -7.754544 | -2.518390 | -3.140033 | -0.050320 |
| DecisionTreeRegressor | -11.837542 | -3.799079 | -4.746385 | -1.437139 |
| RandomForestRegressor | -7.907961 | -2.554298 | -3.199981 | -0.091714 |
| Ridge | -7.753969 | -2.518327 | -3.139936 | -0.050255 |
| Lasso | -7.687178 | -2.506801 | -3.114610 | -0.033390 |
| ElasticNet | -7.687178 | -2.506801 | -3.114610 | -0.033390 |
| SVR | -8.007060 | -2.523153 | -3.160472 | -0.066064 |
| SGDRegressor | -7.748267 | -2.517246 | -3.137990 | -0.048817 |
| KNeighborsRegressor | -8.613586 | -2.741288 | -3.440714 | -0.269092 |

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| LinearRegression(positive=True) | -7.709031 | -2.505067 | -3.127999 | -0.041764 |
| DecisionTreeRegressor(max_depth=1, max_features=2, random_state=42) | -7.837971 | -2.498890 | -3.117294 | -0.035270 |
| RandomForestRegressor(criterion='absolute_error', max_depth=3, max_features=1,\n n_estimators=200, random_state=42) | -7.723437 | -2.502208 | -3.117585 | -0.036191 |
| Ridge(alpha=91, positive=True, solver='lbfgs') | -7.693051 | -2.503336 | -3.123893 | -0.039065 |
| Lasso(alpha=1, positive=True) | -7.687178 | -2.506801 | -3.114610 | -0.033390 |
| ElasticNet(alpha=1, l1_ratio=0.0, positive=True, tol=4000.0) | -7.662191 | -2.502964 | -3.116174 | -0.034063 |
| SVR(C=1, coef0=0, gamma='auto', kernel='sigmoid') | -8.003318 | -2.516191 | -3.176419 | -0.075434 |
| SGDRegressor(alpha=1, l1_ratio=0.2, learning_rate='constant', max_iter=10000000,\n penalty='elasticnet', random_state=42) | -7.684682 | -2.501460 | -3.104073 | -0.026146 |
| KNeighborsRegressor(algorithm='ball_tree', n_neighbors=23, p=1,\n weights='distance') | -7.892295 | -2.543740 | -3.168132 | -0.071450 |

В работе применялись модели машинного обучения из библиотеки scikit-learn, применяемые для задач регрессии.

Был проведен поиск оптимальных гиперпараметров моделей с помощью поиска по сетке с перекрестной проверкой (GridSearchCV).

Перед обучением моделей датасеты были разделены на обучающую и тестовую выборки (70% на обучение и 30% на тестирование).

Обучающая выборка была масштабирована с помощью StandardScaler().

Выбор лучшей модели проводился по метрикам max_error, MAE, RMSE, R2.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка и обучение модели ML

## «Прочность при растяжении, МПа»

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| LinearRegression | -1278.790633 | -389.140981 | -489.393761 | -0.023017 |
| DecisionTreeRegressor | -1853.212070 | -565.241047 | -705.793418 | -1.187462 |
| RandomForestRegressor | -1295.309292 | -397.583551 | -496.374281 | -0.056399 |
| Ridge | -1278.701017 | -389.121962 | -489.376075 | -0.022937 |
| Lasso | -1276.501200 | -388.572626 | -488.798449 | -0.020419 |
| ElasticNet | -1260.882923 | -385.862788 | -486.452377 | -0.009559 |
| SVR | -1256.587914 | -386.793138 | -487.032580 | -0.010721 |
| KNeighborsRegressor | -1404.589902 | -422.401615 | -529.352566 | -0.201183 |

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| LinearRegression(positive=True) | -1258.054267 | -385.924801 | -488.176644 | -0.014997 |
| DecisionTreeRegressor(criterion='absolute_error', max_depth=3, max_features=7,\n random_state=42, splitter='random') | -1267.015475 | -381.101592 | -482.397665 | 0.008114 |
| RandomForestRegressor(criterion='poisson', max_depth=2, max_features=2,\n n_estimators=200, random_state=42) | -1247.264367 | -383.937358 | -484.824258 | -0.002294 |
| Ridge(alpha=91, positive=True, solver='lbfgs') | -1255.130626 | -385.688933 | -487.804637 | -0.013487 |
| Lasso(alpha=21) | -1250.154999 | -383.695991 | -484.101871 | 0.000730 |
| ElasticNet(alpha=21, l1_ratio=1.0, tol=4000.0) | -1249.856166 | -383.695406 | -484.127785 | 0.000621 |
| SVR(C=11, coef0=0, degree=4, kernel='poly') | -1224.265261 | -383.879245 | -485.013673 | -0.001935 |
| KNeighborsRegressor(algorithm='ball_tree', n_neighbors=27, p=1) | -1256.245922 | -386.941168 | -486.491225 | -0.009783 |

В работе применялись модели машинного обучения из библиотеки scikit-learn, применяемые для задач регрессии.

Был проведен поиск оптимальных гиперпараметров моделей с помощью поиска по сетке с перекрестной проверкой (GridSearchCV).

Перед обучением моделей датасеты были разделены на обучающую и тестовую выборки (70% на обучение и 30% на тестирование).

Обучающая выборка была масштабирована с помощью StandardScaler().

Выбор лучшей модели проводился по метрикам max_error, MAE, RMSE, R2.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка и обучение модели ML

Модуль упругости при растяжении, ГПа

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| **SGDRegressor** | -9.40674 | -2.482018 | -3.078749 | -0.020254 |

Прочность при растяжении, МПа

| | max_error | MAE | RMSE | R2 |
|---|---|---|---|---|
| **DecisionTreeRegressor** | -1468.095589 | -380.172727 | -477.100938 | -0.010834 |

Модели были обучены и применены на тестовой выборке для прогноза целевого признака.

Далее мы рассчитали показатели качества моделей, сравнив результаты прогноза с целевыми данными.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка нейронной сети

```
# Целевой параметр - "Соотношение матрица-наполнитель"
y = df_clean.iloc[:, df_clean.columns == 'Соотношение матрица-наполнитель']

# Входные параметры
X = df_clean.drop(columns=['Соотношение матрица-наполнитель'])

# Посмотрим форму полученных данных.
print(y.shape)
print(X.shape)
```

```
(999, 1)
(999, 12)
```

```
X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

Разработаем нейронную сеть для рекомендации соотношения матрица-наполнитель.

Для этого проведём разбиение датасета на входные (X) и целевые (y) данные.

Далее масштабируем входные данные.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка нейронной сети

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1664

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 32)                2080

 dropout_2 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 16)                528

 dropout_3 (Dropout)         (None, 16)                0

 dense_4 (Dense)             (None, 1)                 17

=================================================================
Total params: 12,545
Trainable params: 12,545
Non-trainable params: 0
```

```
# Компиляция

model.compile(loss = keras.losses.MeanSquaredError(),
              optimizer = keras.optimizers.Adam(),
              metrics = [keras.metrics.MeanAbsoluteError()])
```

```
# метод EarlyStopping

earlystopping = keras.callbacks.EarlyStopping(monitor='val_loss',
                                              min_delta=0, patience=10,
                                              verbose=0, mode='auto',
                                              restore_best_weights=True)
```

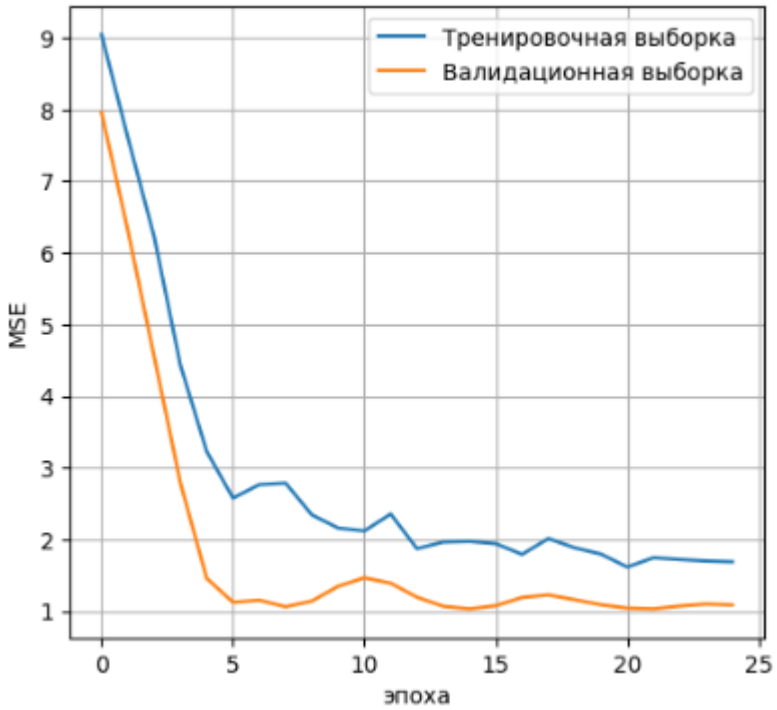
```
# Обучение

history = model.fit(X_train_scaled, y_train_z,
                    batch_size = 150, epochs = 1000,
                    validation_split=0.2,
                    callbacks=[earlystopping], verbose = 1)
```

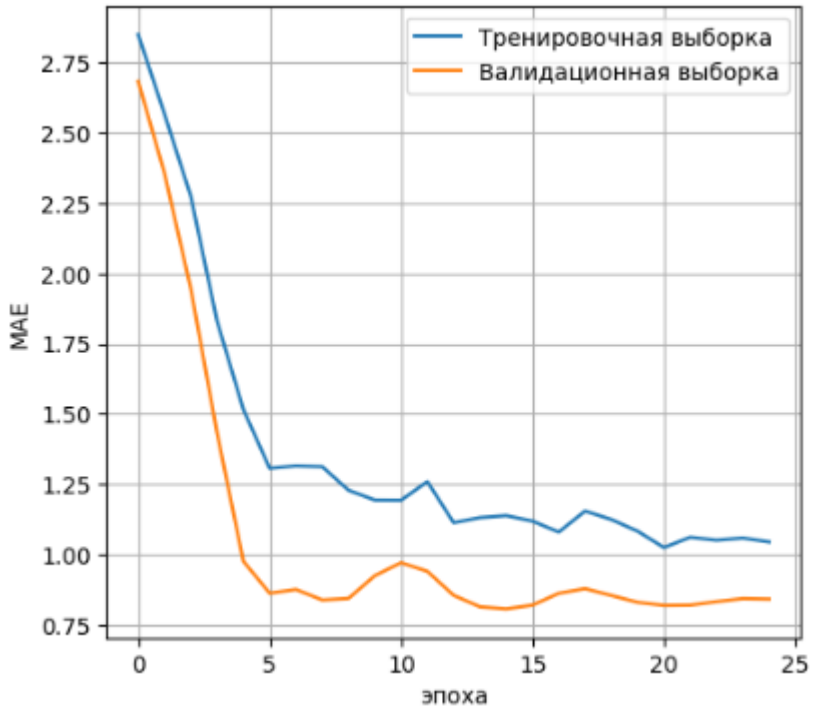
В данной работе используется последовательная модель Sequential() из библиотеки «Tensorflow» с пятью полносвязными слоями Dense(), четырьмя вспомогательными слоями Dropout. Во входном и внутренних слоях используется функция активации «relu», в выходном слое –«linear».

В обучении модели используется метод EarlyStopping.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка нейронной сети



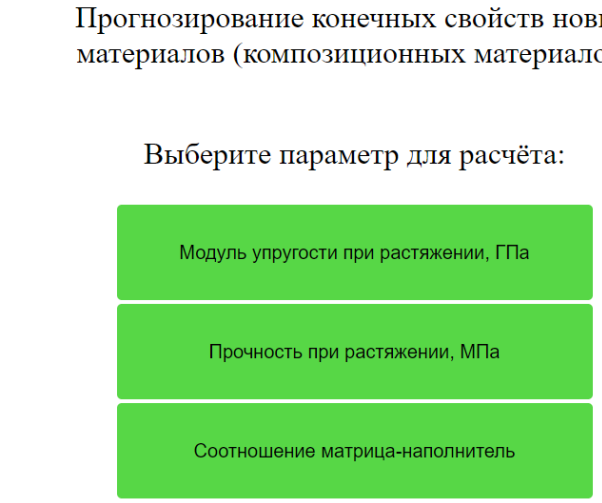

```
6/6 [==============================] - 0s 8ms/step - loss: 1.0781 - mean_absolute_error: 0.8279
MSE и MAE на тестовой выборке: [1.0780550241470337, 0.827873706817627]
```

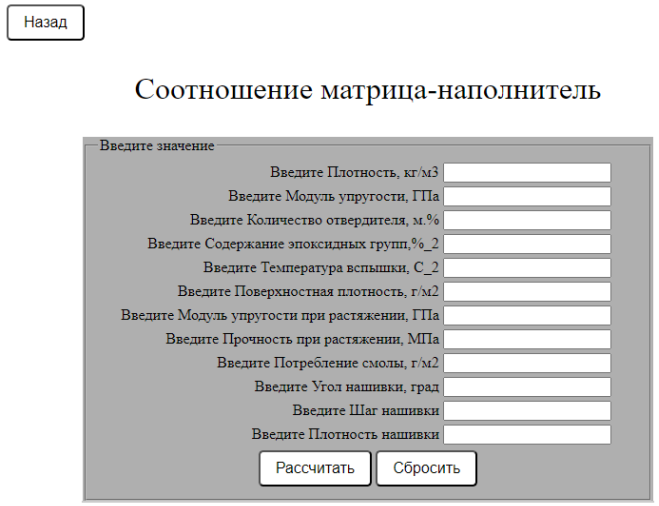
График показывает уменьшение метрик MSE и MAE в процессе обучения.

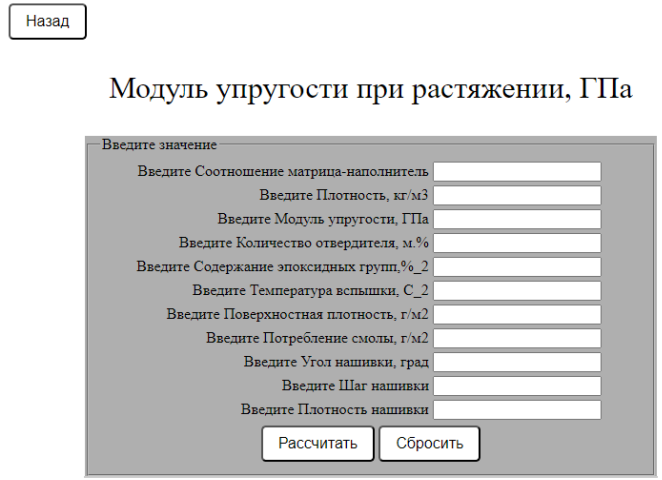
При применении модели на тестовой выборке MSE составил 1.08, а MAE составил 0.83

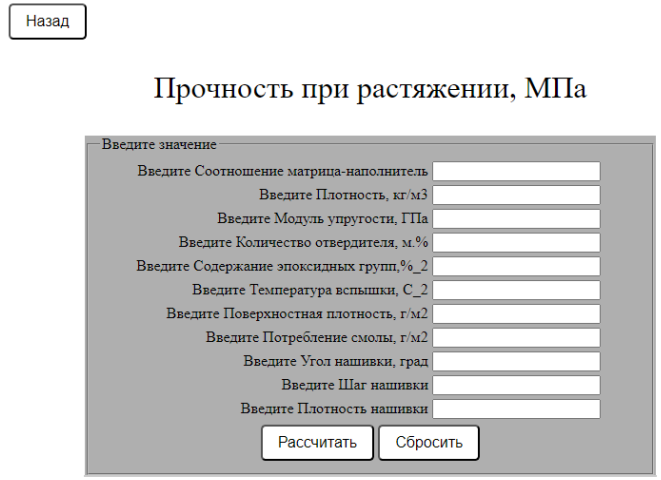
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка приложения




Web-приложение с использованием библиотеки Flask для применения наших моделей на практике.
Приложение запускается локально в браузере из среды Python.
Web-приложение выполнено по многостраничной схеме.
На стартовой странице выбирается признак для расчета.
После выбора признака пользователь перенаправляется на соответствующую страницу.
Для расчета необходимо ввести данные в соответствующие поля и нажать на кнопку «Рассчитать» в нижней части формы ввода данных.
Кнопка «Сбросить» в нижней части формы ввода данных очищает поля.
Результат расчета отображается ниже формы ввода данных.
Для возврата на стартовую страницу нажмите на кнопку «Назад» в верхнем левом углу.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Репозиторий

Файлы исследования и приложение размещены в репозитории:

https://github.com/Alvas01/BMSTU

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Заключение

Примененные нами методы машинного обучения и нейронная сеть показали невысокую эффективность.

Разведочный анализ данных показал низкую (околонулевую) корреляцию признаков в датасете. Думаю, что это стало причиной низкой эффективности моделей, которые строят свою работу на выявлении взаимосвязей в данных.

Предполагаю, что в предоставленных данных имеются скрытые недостатки. Для повышения эффективности моделей необходима дополнительная информация и, возможно, более тонкая и трудоемкая настройка моделей.

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru