ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Слушатель: Ющенко Борис Евгеньевич

Этапы работы

Аналитическая часть:

- **Постановка задачи**
- **Подбор методов обучения**

Практическая часть:

- **Разведочный анализ, предобработка данных**
- **Разработка, обучение и тестирование моделей**
- **Нейронная сеть для рекомендации соотношения матрица-наполнитель**
- **Разработка приложения на фреймфорке Flask**

Задачи исследования:

разработка моделей прогнозирования

- ✓ модуля упругости при растяжении
- ✓ прочности при растяжении
- ✓ соотношения матрица-наполнитель.

Задача регрессии в машинном обучении – предсказание одного параметра (Y) по известному параметру X, где X – набор параметров, характеризующих наблюдение.

Методы машинного обучения:

- ❖ Линейная регрессия (Linear Regression)
- ❖ Метод ближайших соседей (KNeighborsRegressor)
- ❖ Гребневая регрессия (Ridge)
- ❖ Стохастический градиентный спуск (SGD)
- ❖ Градиентный бустинг (Gradient Boosting)
- ❖ Метод опорных векторов для регрессии (SVR)
- ❖ «Случайный лес» (Random Forest)
- ❖ XGBRegressor
- ❖ Дерево принятия решений (Decision Tree Regressor)

+

Нейронная сеть

```
X_bp = pd.read_excel('initial_dataset/X_bp.xlsx', index_col=0)
X_bp.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
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0	4.0	57.0
1	0	4.0	60.0
2	0	4.0	70.0
3	0	5.0	47.0
4	0	5.0	57.0

	0	1	2	3	4
Соотношение матрица-наполнитель	1.857143	1.857143	1.857143	1.857143	2.771331
Плотность, кг/м3	2030.000000	2030.000000	2030.000000	2030.000000	2030.000000
Модуль упругости, ГПа	738.736842	738.736842	738.736842	738.736842	753.000000
Количество отвердителя, м.%	30.000000	50.000000	49.900000	129.000000	111.860000
Содержание эпоксидных групп, %	22.267857	23.750000	33.000000	21.250000	22.267857
Температура вспышки, С	100.000000	284.615385	284.615385	300.000000	284.615385
Поверхностная плотность, г/м2	210.000000	210.000000	210.000000	210.000000	210.000000
Модуль упругости при растяжении, ГПа	70.000000	70.000000	70.000000	70.000000	70.000000
Прочность при растяжении, МПа	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
Потребление смолы, г/м2	220.000000	220.000000	220.000000	220.000000	220.000000
Угол нашивки, град	0.000000	0.000000	0.000000	0.000000	0.000000
Шаг нашивки	4.000000	4.000000	4.000000	5.000000	5.000000
Плотность нашивки	57.000000	60.000000	70.000000	47.000000	57.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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, %         1023 non-null   float64
 5   Температура вспышки, С                 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: 111.9 KB
```

	count	mean	std	min	25%	50%	75%	max
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, %	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С	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

```
data.isna().sum()
```

```
Соотношение матрица-наполнитель          0
Плотность, кг/м3                         0
Модуль упругости, ГПа                    0
Количество отвердителя, м.%              0
Содержание эпоксидных групп, %           0
Температура вспышки, С                   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
```

Как видим, таковых действительно нет.

```
# Проанализируем количество дубликатов.

data.duplicated().sum()
```

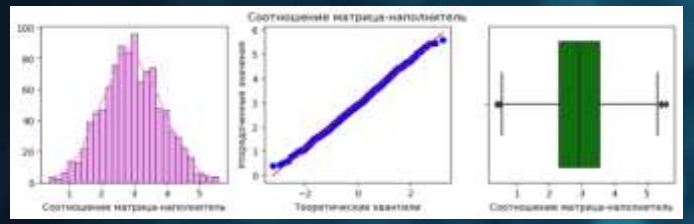
```
0
```

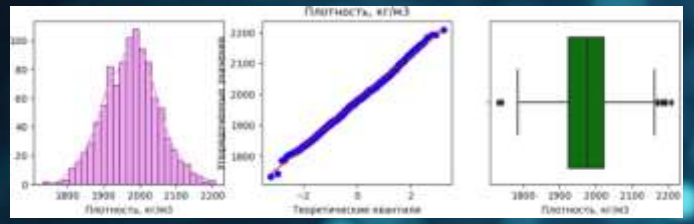
Объединение датасета

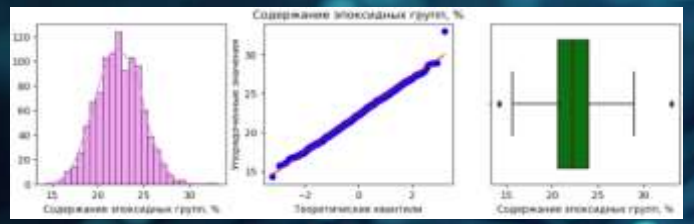
- ✓ На входе дано 2 датасета
- ✓ Объединение выполнено по индексу, тип объединения – INNER
- ✓ Удалено 17 строк
- ✓ Объединенный датасет получил 1023 строки, 13 столбцов

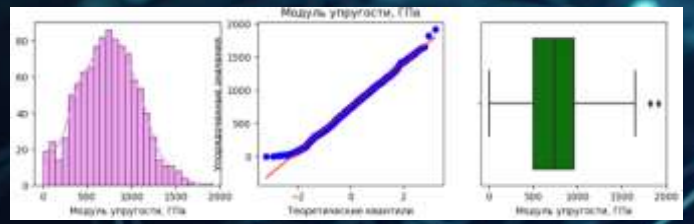
Разведочный анализ

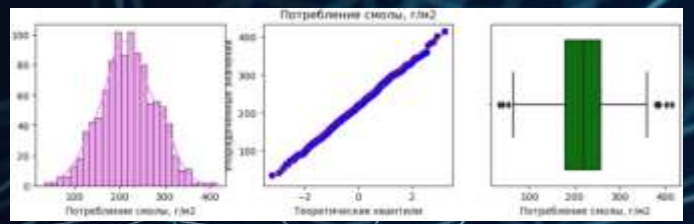
- ➢практически все столбцы имеют тип данных "float64", кроме столбца "Угол нашивки" - "int64"
- ➢пропущенных значений и дубликатов нет, чистка не требуется

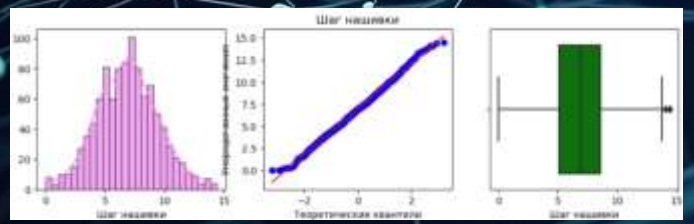


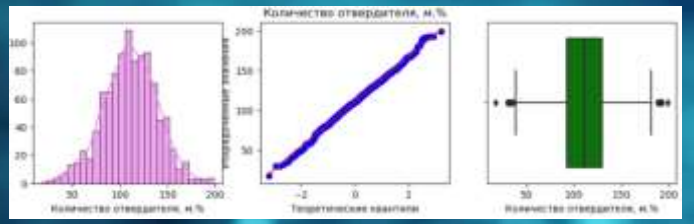


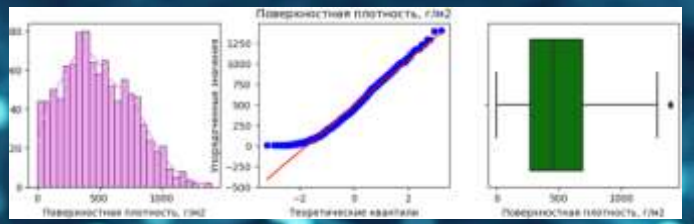


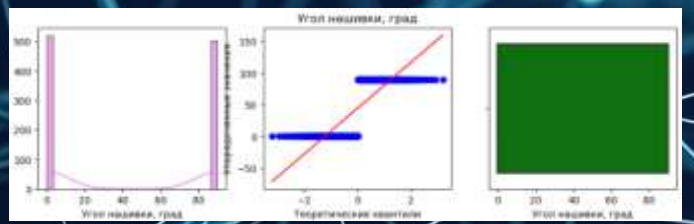


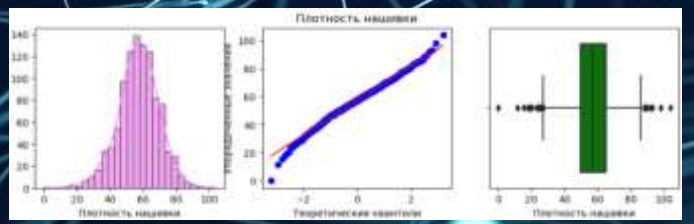








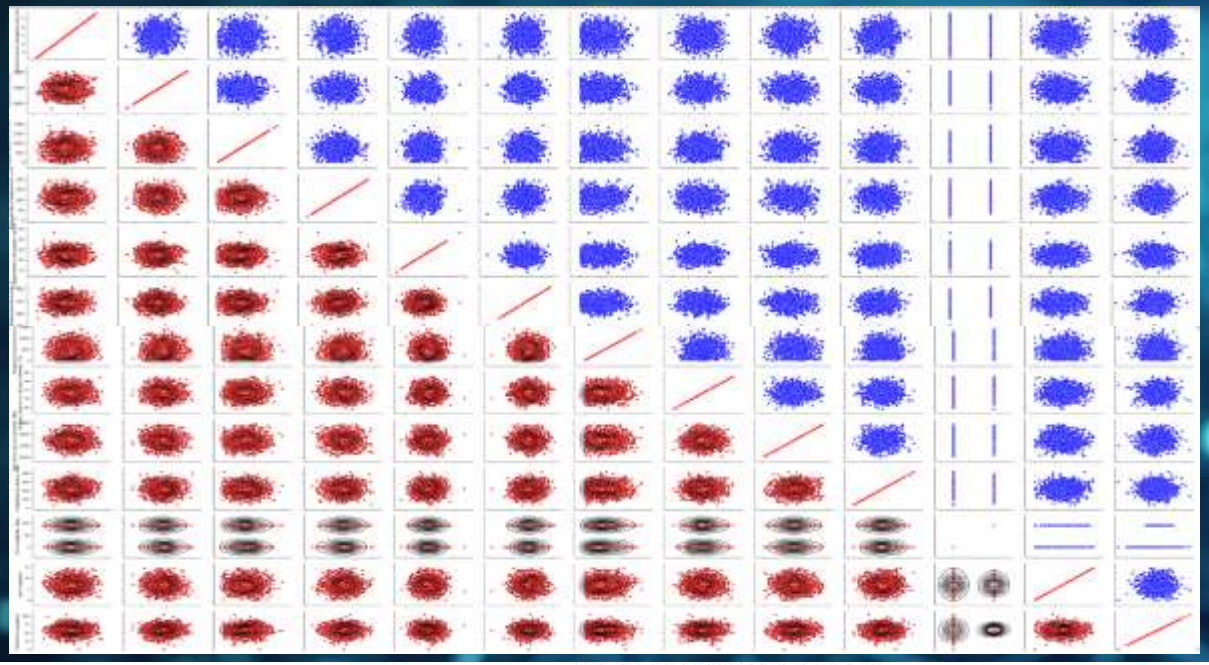


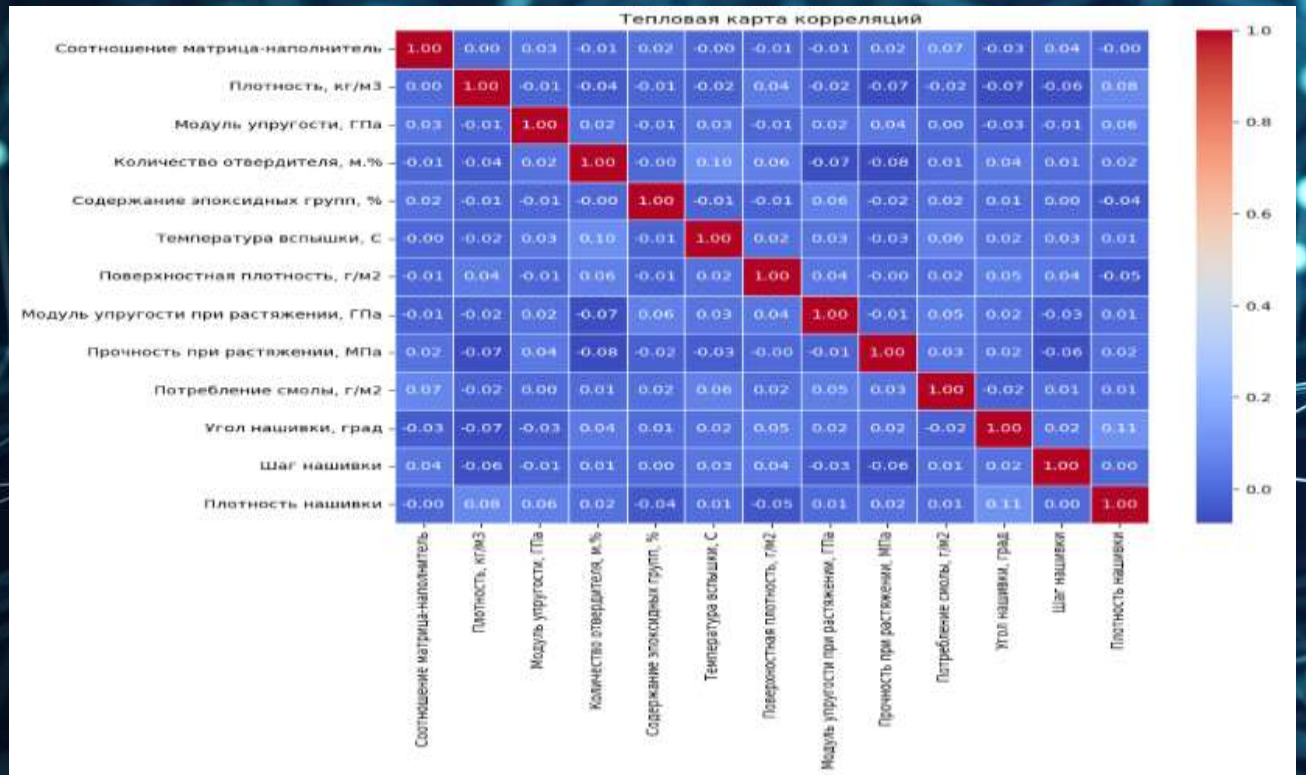


- Построены гистограммы распределения точек, вероятностные графики и диаграммы размаха так называемые «ящики с усами»

- Распределение подавляющего большинства параметров является нормальным или близким к нему (кроме параметра «Угол нашивки», принимающего лишь 2 значения)

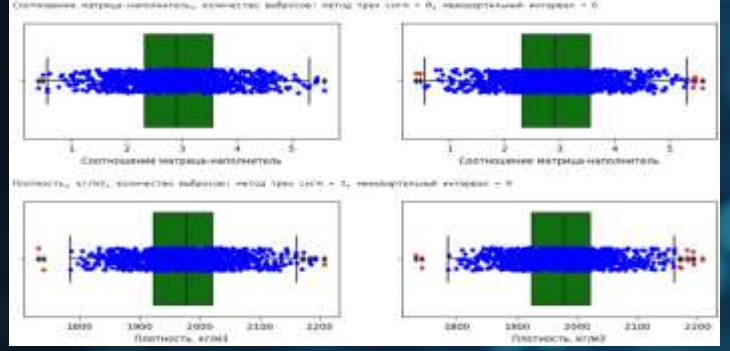
- Диаграммы «Ящик с усами» показали, что у всех признаков имеются выбросы

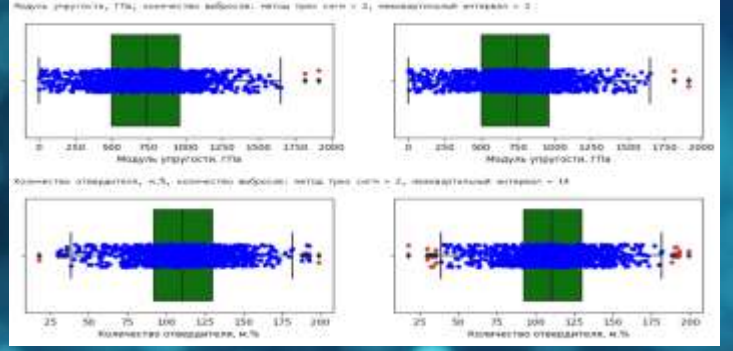


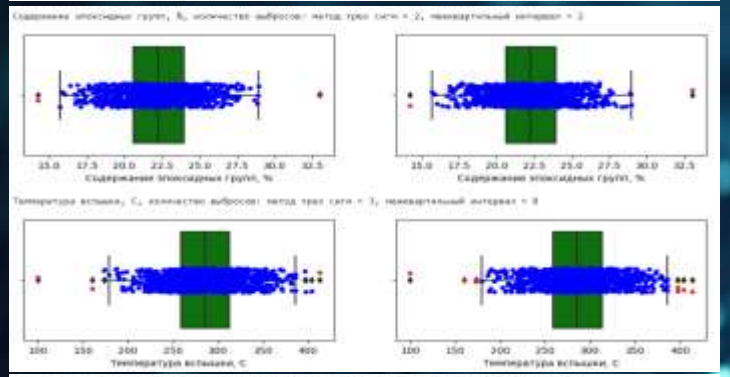


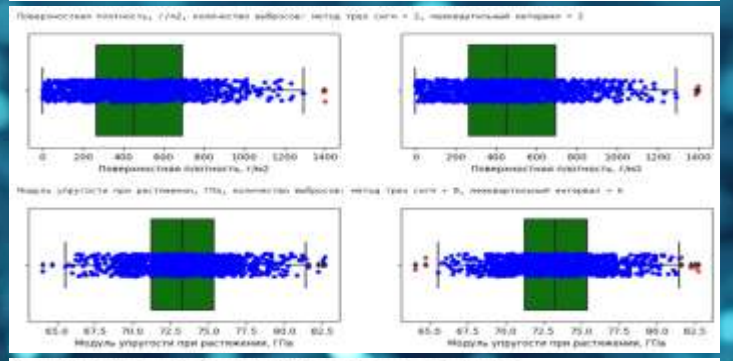
❖По представленным рисункам попарного сравнения признаков также выявлено наличие выбросов

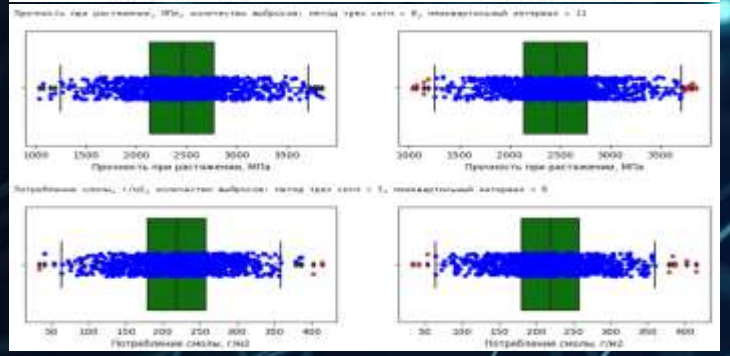
❖Корреляционная матрица каких-либо четких зависимостей зависимости между параметрами не выявила

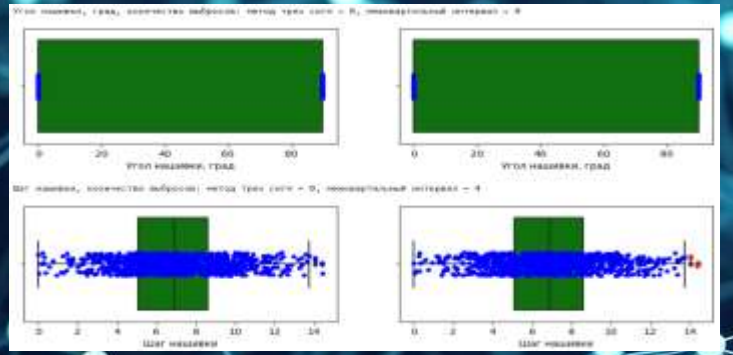


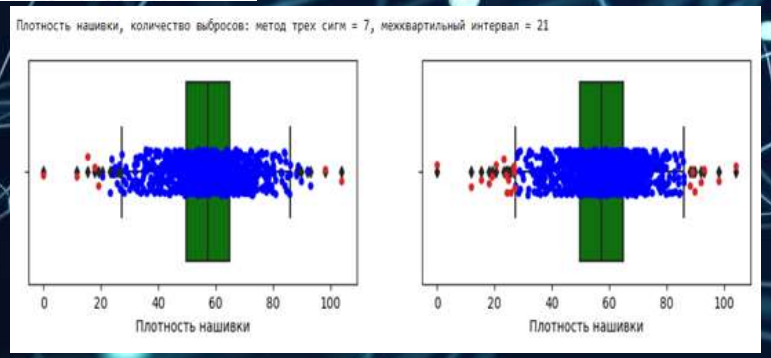










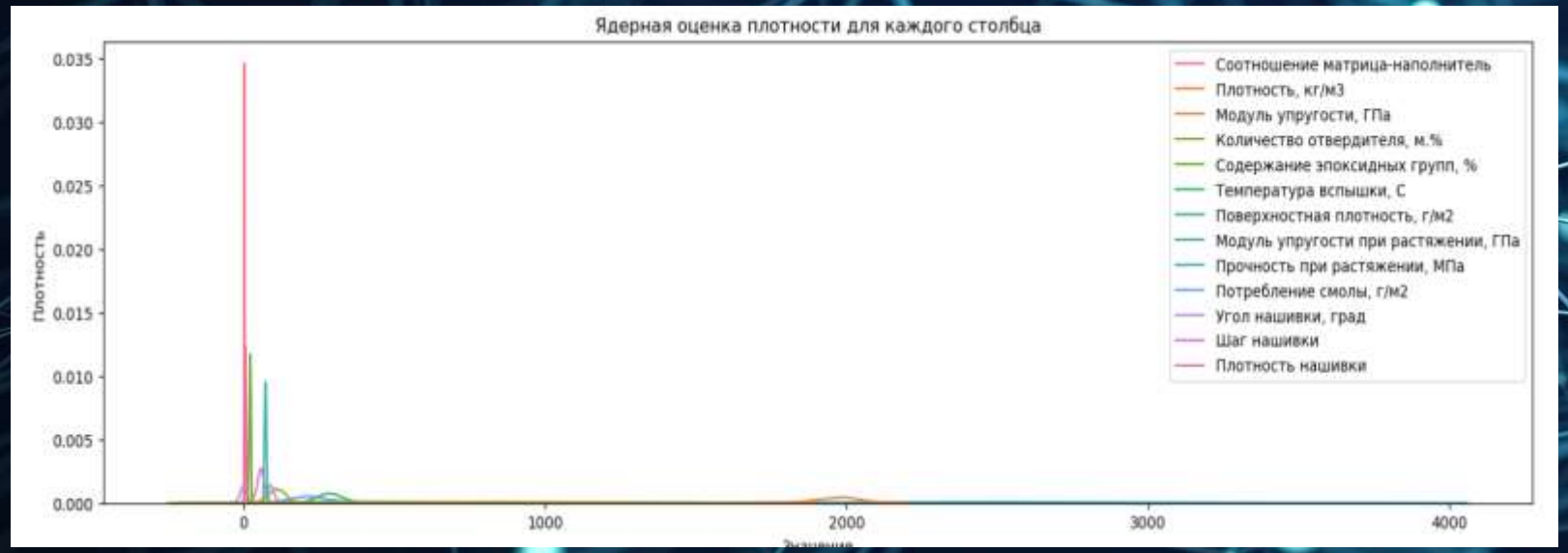


Предобработка данных

- Для подсчета выбросов используем методы трех сигм и межквартильного интервала
- Визуализируем попарное сравнение методов (красными точками показаны выбросы, найденные этими методами).
- Количество выбросов по методу трех сигм: 24
- Количество выбросов по методу межквартильного интервала: 93

```python
# Проведем статистические тесты на нормальность распределения данных для каждого столбца в датасете с использованием тестов Пирсо

result_data = pd.DataFrame(columns=['pearson p-value', 'shapiro p-value'])

# Итерируем по столбцам в DataFrame
for column in data.columns:
    # Тест по Пирсону
    _, pearson_p_value = normaltest(data[column])

    # Тест по Шапиро-Уилку
    _, shapiro_p_value = shapiro(data[column])

    # Записываем p-value в таблицу
    result_data.loc[column] = [round(pearson_p_value, 6), round(shapiro_p_value, 6)]

# Выводим результат
print(result_data)
```

	pearson p-value	shapiro p-value
Соотношение матрица-наполнитель	0.064087	0.086759
Плотность, кг/м3	0.350121	0.094850
Модуль упругости, ГПа	0.019018	0.006075
Количество отвердителя, м.%	0.072428	0.025072
Содержание эпоксидных групп, %	0.064916	0.237023
Температура вспышки, С	0.084906	0.085307
Поверхностная плотность, г/м2	0.000000	0.000000
Модуль упругости при растяжении, ГПа	0.040235	0.023090
Прочность при растяжении, МПа	0.237467	0.078110
Потребление смолы, г/м2	0.012970	0.015642
Угол нашивки, град	0.000000	0.000000
Шаг нашивки	0.638223	0.349185
Плотность нашивки	0.376078	0.052167



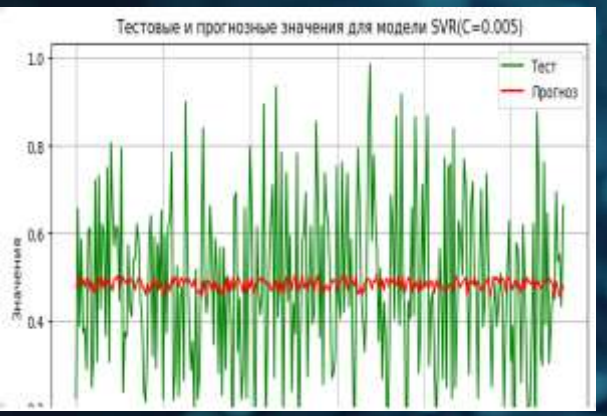
✓ После удаления выбросов проведены статистические тесты Пирсона и Шапиро-Уилка на нормальность распределения данных для каждого столбца

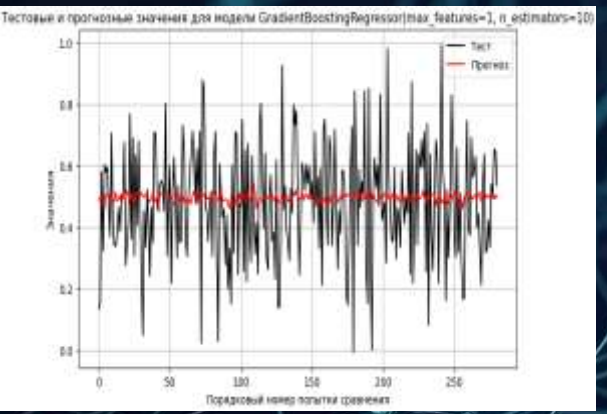
✓ Как видим, в двух столбцах «Поверхностная плотность» и «Угол нашивки» распределение вообще не соответствует нормальному.

✓ Далее был построен график распределения плотности ядра, для оценки необходимости нормализации

✓ Данные находятся в разных диапазонах, необходима нормализация данных

✓ Использовался «масштабатор» MinMaxScaler для приведения в диапазон от 0 до 1.





	MSE_train	MSE_test	MAE_train	MAE_test	R2_train	R2_test
LinearRegression	0.0369	0.0379	0.1532	0.1605	-0.0185	-0.0193
KNeighborsRegressor	0.0364	0.0380	0.1518	0.1620	-0.0060	-0.0233
SGDRegressor	0.0421	0.0386	0.1632	0.1625	-0.1696	-0.0378
GradientBoostingRegressor	0.0364	0.0376	0.1524	0.1600	-0.0025	-0.0123
SVR	0.0365	0.0371	0.1525	0.1589	-0.0059	0.0025
RandomForestRegressor	0.0362	0.0380	0.1516	0.1606	0.0013	-0.0221
XGBRegressor	0.0379	0.0400	0.1543	0.1641	-0.0435	-0.0773
Ridge	0.0366	0.0375	0.1526	0.1598	-0.0092	-0.0088
DecisionTreeRegressor	0.0410	0.0406	0.1598	0.1659	-0.1330	-0.0918

	MSE_train	MSE_test	MAE_train	MAE_test	R2_train	R2_test
LinearRegression	0.0380	0.0354	0.1550	0.1515	-0.0520	-0.0183
KNeighborsRegressor	0.0369	0.0353	0.1528	0.1513	-0.0219	-0.0172
SGDRegressor	0.0369	0.0351	0.1526	0.1508	-0.0204	-0.0110
GradientBoostingRegressor	0.0368	0.0345	0.1517	0.1499	-0.0171	0.0055
SVR	0.0366	0.0349	0.1514	0.1504	-0.0127	-0.0057
RandomForestRegressor	0.0369	0.0351	0.1522	0.1513	-0.0218	-0.0109
XGBRegressor	0.0368	0.0350	0.1521	0.1514	-0.0170	-0.0078
Ridge	0.0374	0.0351	0.1536	0.1509	-0.0349	-0.0094
DecisionTreeRegressor	0.0409	0.0405	0.1604	0.1596	-0.1279	-0.1669

Обучение моделей

- Было использовано 9 методов обучения для каждого целевого параметра
- Для подбора оптимальных параметров использовался метод кросс-валидации Grid Search CV
- Использованные метрики качества обучения:
  - Средняя абсолютная ошибка (Mean Absolute Error, MAE)
  - Среднеквадратичная ошибка MSE (Mean Squared Error, MSE)
  - Коэффициент детерминации (R2)

Лучшие модели

- ✓ Метод опорных векторов (SVR) для «Модуля упругости при растяжении»
- ✓ Градиентный бустинг для «Прочности при растяжении»

```python
model_1 = Sequential()
model_1.add(Dense(X_train_scl_mn.shape[1]))
model_1.add(Dense(16, activation='relu'))
model_1.add(Dense(8, activation='relu'))
model_1.add(Dense(1))
model_1.compile(optimizer='adam', loss='mean_squared_error')
history_1 = model_1.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3)
```

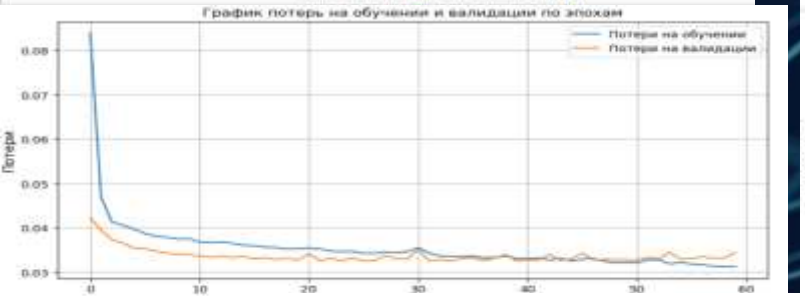
```python
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)

model_2 = Sequential()
model_2.add(Dense(X_train_scl_mn.shape[1]))
model_2.add(Dense(16, activation='relu'))
model_2.add(Dense(8, activation='relu'))
model_2.add(Dense(1))
model_2.compile(optimizer='adam', loss='mean_squared_error')
history_2 = model_2.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3,
                        callbacks=[early_stop])
```

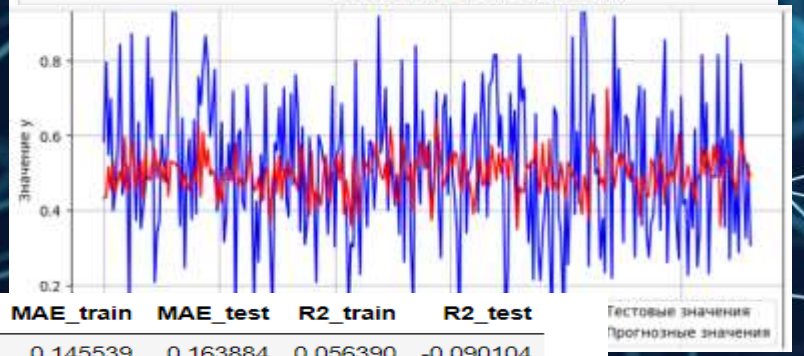
```python
# Попробуем увеличить количество нейронов

model_3 = Sequential()
model_3.add(Dense(X_train_scl_mn.shape[1]))
model_3.add(Dense(32, activation='relu'))
model_3.add(Dense(16, activation='relu'))
model_3.add(Dense(1))
model_3.compile(optimizer='adam', loss='mean_squared_error')
history_3 = model_3.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3,
                        callbacks=[early_stop])
```

```python
# Попробуем еще увеличить количество нейронов

model_4 = Sequential()
model_4.add(Dense(X_train_scl_mn.shape[1]))
model_4.add(Dense(64, activation='relu'))
model_4.add(Dense(32, activation='relu'))
model_4.add(Dense(1, activation='linear'))

model_4.compile(optimizer='adam', loss='mean_squared_error')

history_4 = model_4.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3,
                        callbacks=[early_stop])
```

```python
# Поменяем метод активации скрытых слоев на 'tanh'

model_5 = Sequential()
model_5.add(Dense(X_train_scl_mn.shape[1]))
model_5.add(Dense(32, activation='tanh'))
model_5.add(Dense(16, activation='tanh'))
model_5.add(Dense(1))
model_5.compile(optimizer='adam', loss='mean_squared_error')
history_5 = model_5.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3,
                        callbacks=[early_stop])
```

```python
# Попробуем поменять оптимайзер на 'sgd'

model_6 = Sequential()
model_6.add(Dense(X_train_scl_mn.shape[1]))
model_6.add(Dense(64, activation='tanh'))
model_6.add(Dense(32, activation='tanh'))
model_6.add(Dense(1))
model_6.compile(optimizer='sgd', loss='mean_squared_error')
history_6 = model_6.fit(X_train_scl_mn,
                        y_train_scl_mn,
                        batch_size=32,
                        epochs=100,
                        validation_split=0.3,
                        callbacks=[early_stop])
```

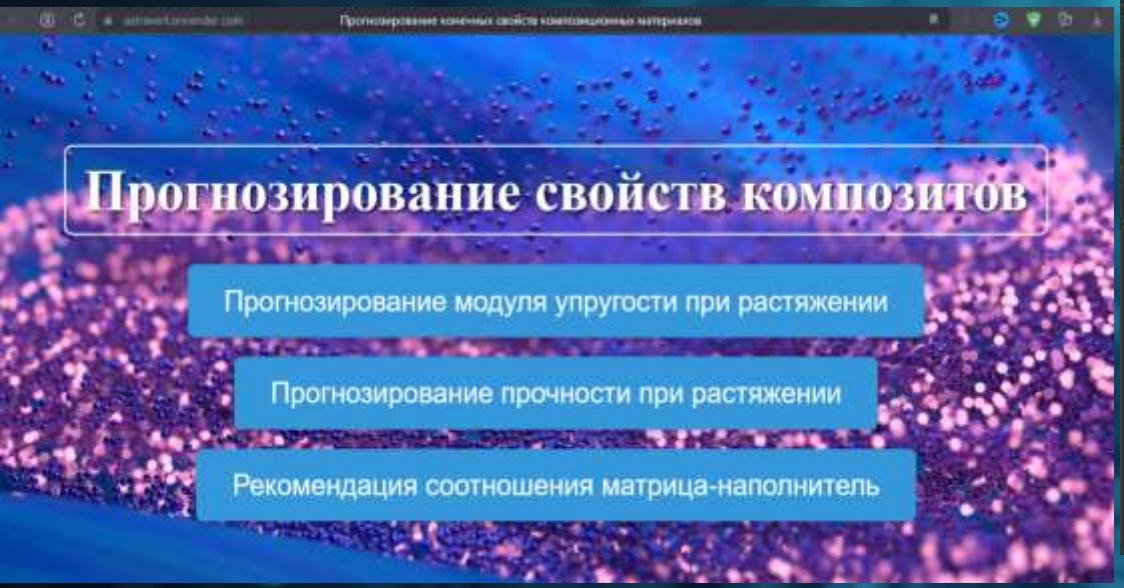


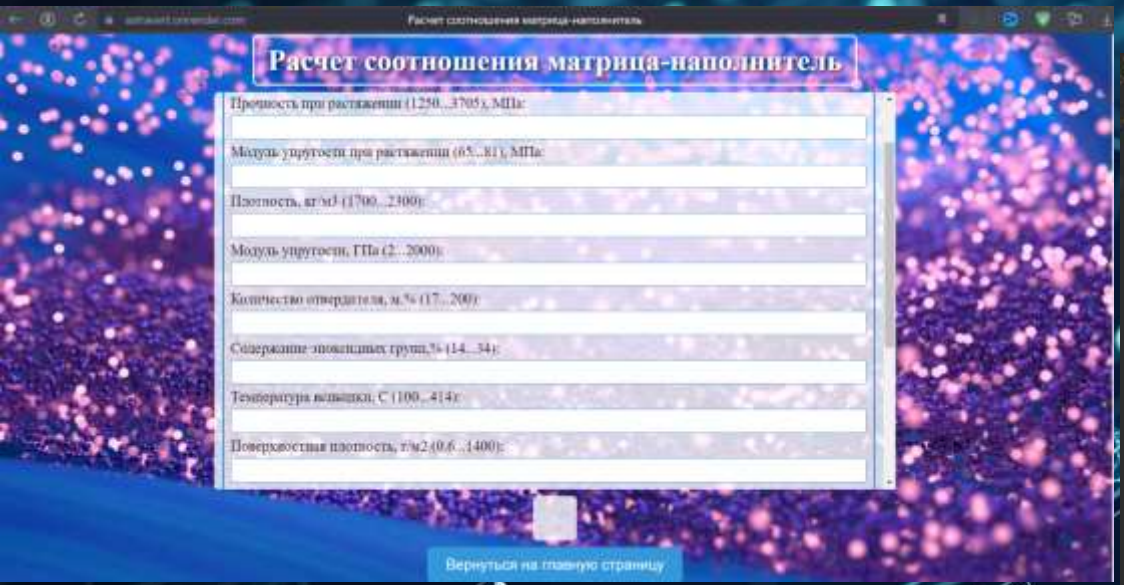


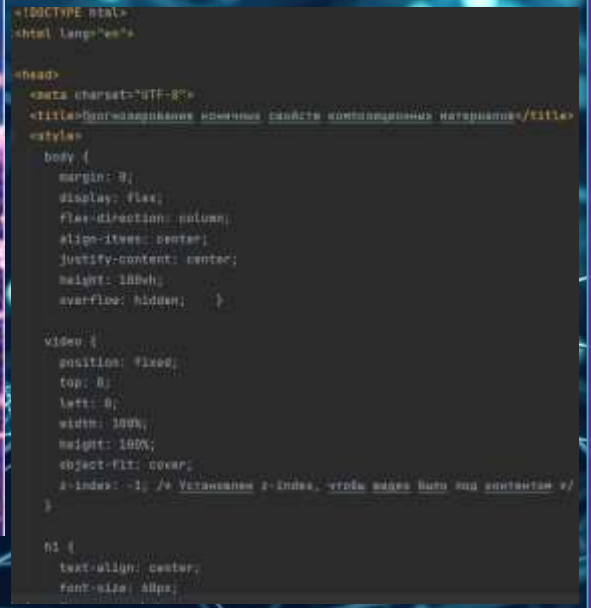
	MSE_train	MSE_test	MAE_train	MAE_test	R2_train	R2_test
Нейронная сеть_1	0.032856	0.039658	0.145539	0.163884	0.056390	-0.090104
Нейронная сеть_2	0.032987	0.038811	0.146960	0.160538	0.052637	-0.066808
Нейронная сеть_3	0.032694	0.037385	0.144849	0.157800	0.061042	-0.027605
Нейронная сеть_4	0.032281	0.040680	0.144856	0.161888	0.072903	-0.118175
Нейронная сеть_5	0.034110	0.037595	0.148250	0.158662	0.020387	-0.033389
Нейронная сеть_6	0.034297	0.037866	0.149332	0.159458	0.015004	-0.040827

Написание нейронной сети

- ✓ Было обучено 6 моделей
- ✓ Лучшей моделью стал вариант № 3 со следующими параметрами: 4 слоя - один входной, два скрытых по 32 и 16 нейроновс с методом активации «relu», один выходной с методом активации «linear», оптимайзер –“adam“.







Разработка приложения

- ✓ Был использован фреймворк Flask
- ✓ Рассчет производится по всем трем целевым парметрам
- ✓ В случае указания чисел, выходящих за необходимые диапазоны значений или при вводе иных символов выводится ошибка с указанием строк, где указаны неверные значения
- ✓ Предусмотрена кнопка "Вернуться на главную страницу"
- ✓ Приложение было размещено на сайте: https://astravert.onrender.com/

Создание репозитория

- ✓ репозиторий создан на GitHub: https://github.com/Astravert/Composites_VKR

Заключение

❖Несмотря на то, что распределение данных в получившемся датасете демонстрирует близкое к нормальному, какой-либо существенной корреляции между признаками посредством стандартных методов найти не удалось

❖Обученные модели, полученные в исследовании, к сожалению, не продемонстрировали высокой эффективности в предсказании свойств композитов

❖Вероятно, при предоставлении дополнительных вводных данных, проведении консультаций с профильными экспертами удалось бы выйти на более эффективные предсказательные модели

Спасибо за внимание !