

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу
«Data Science»

**Прогнозирование конечных свойств
новых материалов (композиционных материалов)**

Слушатель

В.С. Благодатских

Москва, 2022

Содержание

Введение.....	3
1. Постановка задачи, описание исходных данных.....	4
2. Разведочный анализ данных и визуализация исходных данных	5
3. Предобработка данных.....	10
4. Обучение моделей для прогноза модуля упругости при растяжении и прочности при растяжении.	11
4.1. Разбиение датасета для дальнейшего обучения моделей.	11
4.2. Обучение моделей линейной регрессии.....	13
4.3. Обучение модели К-ближайших соседей (KNeighborsRegressor) .	17
4.4. Обучение модели «Случайного леса» (Random Forest Regressor).	
Подбор параметров.	18
5. Разработка нейронных сетей	21
5.1. Нейронные сети с одним выходом для параметра соотношение матрица-наполнитель.....	21
5.2. Построение нейронной сети с один выходом для предсказания модуля упругости при растяжении	30
6. Создание приложения с графическим интерфейсом для предсказания модуля упругости	33
7. Заключение	34
Список литературы	35

Введение

В рамках выпускной квалификационной работы по курсу «Data Science» на тему «Прогнозирование конечных свойств новых материалов (композиционных материалов)» проведены исследования и анализ предоставленных данных о начальных свойствах компонентов композиционных материалов с использованием методов, изученных на курсе «Data Science».

Композиционный материал, композит — многокомпонентный материал, изготовленный (человеком или природой) из двух или более компонентов с существенно различными физическими и/или химическими свойствами, которые, в сочетании, приводят к появлению нового материала с характеристиками, отличными от характеристик отдельных компонентов и не являющимися простой их суперпозицией. При этом отдельные компоненты остаются таковыми в структуре композитов, отличая их от смесей и твёрдых растворов. В составе композита принято выделять матрицу/матрицы и наполнитель/наполнители. Варьируя состав матрицы и наполнителя, их соотношение, ориентацию наполнителя, получают широкий спектр материалов с требуемым набором свойств. Многие композиты превосходят традиционные материалы и сплавы по своим механическим свойствам и в то же время они легче. Использование композитов обычно позволяет уменьшить массу конструкции при сохранении или улучшении её механических характеристик.

1. Постановка задачи, описание исходных данных

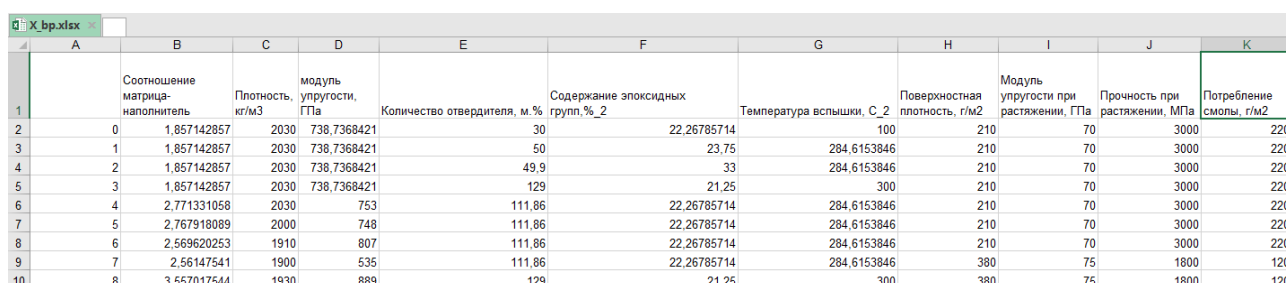
Целью данной работы является обработка данных и построение при помощи методов машинного обучения модели прогнозирования характеристик «модуль упругости при растяжении» и «прочность при растяжении», рекомендации «соотношение матрица-наполнитель».

Исходные данные свойств композиционных материалов получены от структурного подразделения МГТУ им. Н.Э. Баумана Центра НТИ «Цифровое материаловедение: новые материалы и вещества».

Данные представляют собой характеристики и свойства композитных материалов, содержащиеся в двух таблицах Excel.

Исходные данные включают в себя:

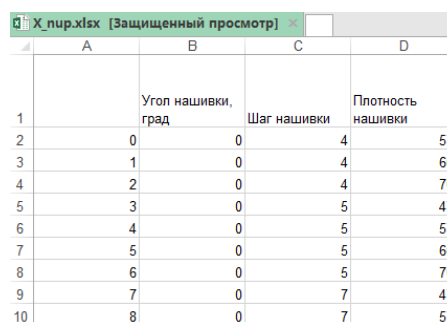
1. Таблицу X_br.xlsx с количеством строк 1023, где находятся три исследуемых параметра (характеристики композита), семь входных (химико-физические соотношения и свойства используемых компонент), одна колонка с индексом (Рисунок 1).



	A	B	C	D	E	F	G	H	I	J	K
1		Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м. %	Содержание эпоксидных групп, %_2	Температура вспышки, C_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2
2	0	1,857142857	2030	738,7368421	30	22,26785714	100	210	70	3000	220
3	1	1,857142857	2030	738,7368421	50	23,75	284,6153846	210	70	3000	220
4	2	1,857142857	2030	738,7368421	49,9	33	284,6153846	210	70	3000	220
5	3	1,857142857	2030	738,7368421	129	21,25	300	210	70	3000	220
6	4	2,771331058	2030	753	111,86	22,26785714	284,6153846	210	70	3000	220
7	5	2,767918089	2000	748	111,86	22,26785714	284,6153846	210	70	3000	220
8	6	2,569620253	1910	807	111,86	22,26785714	284,6153846	210	70	3000	220
9	7	2,56147541	1900	535	111,86	22,26785714	284,6153846	380	75	1800	120
10	8	3,557017544	1930	889	129	21,25	300	380	75	1800	120

Рисунок 1 – Таблица X_br.xlsx

2. Таблицу X_nur.xlsx с количеством элементов 1040, где находится три входных параметра (физические характеристики исследуемых образцов) (Рисунок 2) и одна колонка с индексом.



	A	B	C	D
1		Угол нашивки, град	Шаг нашивки	Плотность нашивки
2	0	0	4	57
3	1	0	4	60
4	2	0	4	70
5	3	0	5	47
6	4	0	5	57
7	5	0	5	60
8	6	0	5	70
9	7	0	7	47
10	8	0	7	57

Рисунок 1 – Таблица X_nur.xlsx

Колонка «А» с индексом в обеих таблицах служит для их объединения.

Анализ, предобработка данных, построение моделей выполнены посредством языка программирования Python, с применением библиотек numpy, pandas, matplotlib, seaborn, sklearn.

2. Разведочный анализ данных и визуализация исходных данных

Разведочный анализ данных в рамках поставленной задачи необходимо провести способом объединения по типу inner по полю индекса таблиц исходных данных

17 строк данных не были включены в общий датасет, так как они не имеют отличных строк в таблице X_br.xlsx

Сформированный исходный датафрейм содержит 1023 записи с входными параметрами и 3 выходными параметрами, пропуски значений отсутствуют, дублирующих значений нет (Рисунок 3).

```
df.isna().sum()
Соотношение матрица-наполнитель      0
Плотность, кг/м3                      0
модуль упругости, ГПа                 0
Количество отвердителя, м.%           0
Содержание эпоксидных групп,%_2      0
Температура вспышки, C_2              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
```

Пропусков нет

```
df.duplicated().sum()
0
```

Дубликатов нет

Рисунок 3 – Количество пропусков и дубликатов

Значения колонки 'Угол нашивки, град' были перекодированы в бинарный вид (Рисунок 4).

```
# Преобразуем значения "Угол нашивки" с помощью LabelEncoder
le = preprocessing.LabelEncoder()
df['Угол нашивки, град'] = le.fit_transform(df['Угол нашивки, град'].values)
df['Угол нашивки, град']
```

Рисунок 4 – Преобразование значений «Угол нашивки»

Описательная статистика (Рисунок 5), гистограммы распределения (Рисунок 6), диаграммы размаха «Ящик с усами» (Рисунок 7), позволяют получить наглядное представление о характерах распределений переменных.

df.describe().T

	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,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	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	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

Рисунок 5 – Описательная статистика

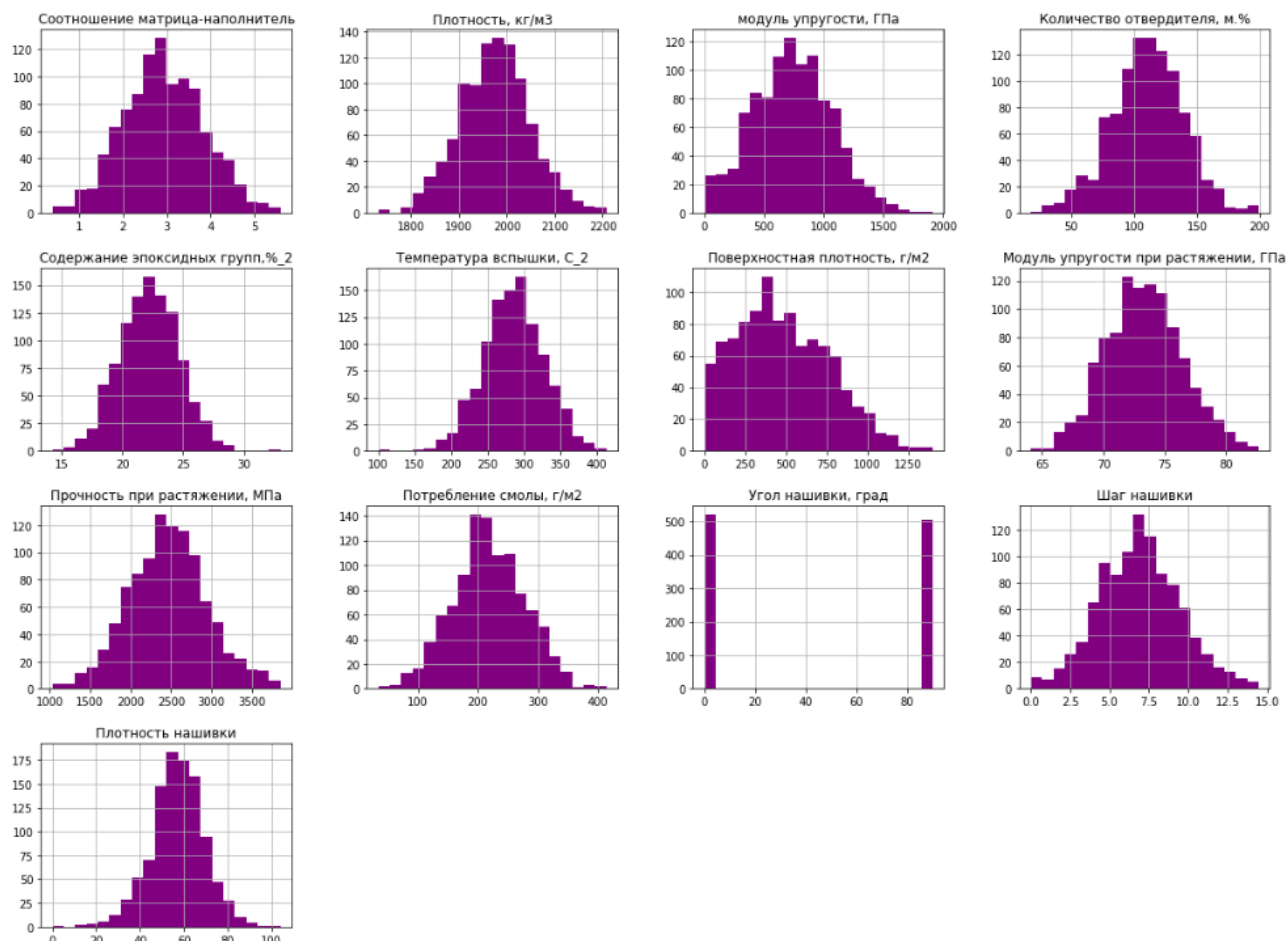


Рисунок 6 – Гистограммы распределения переменных

Из всех параметров, помимо «Угла нашивки», имеющего всего два значения, выделяются «Поверхностная плотность, г/м2» и «модуль упругости, ГПа» форма распределения менее других походит на нормальное.

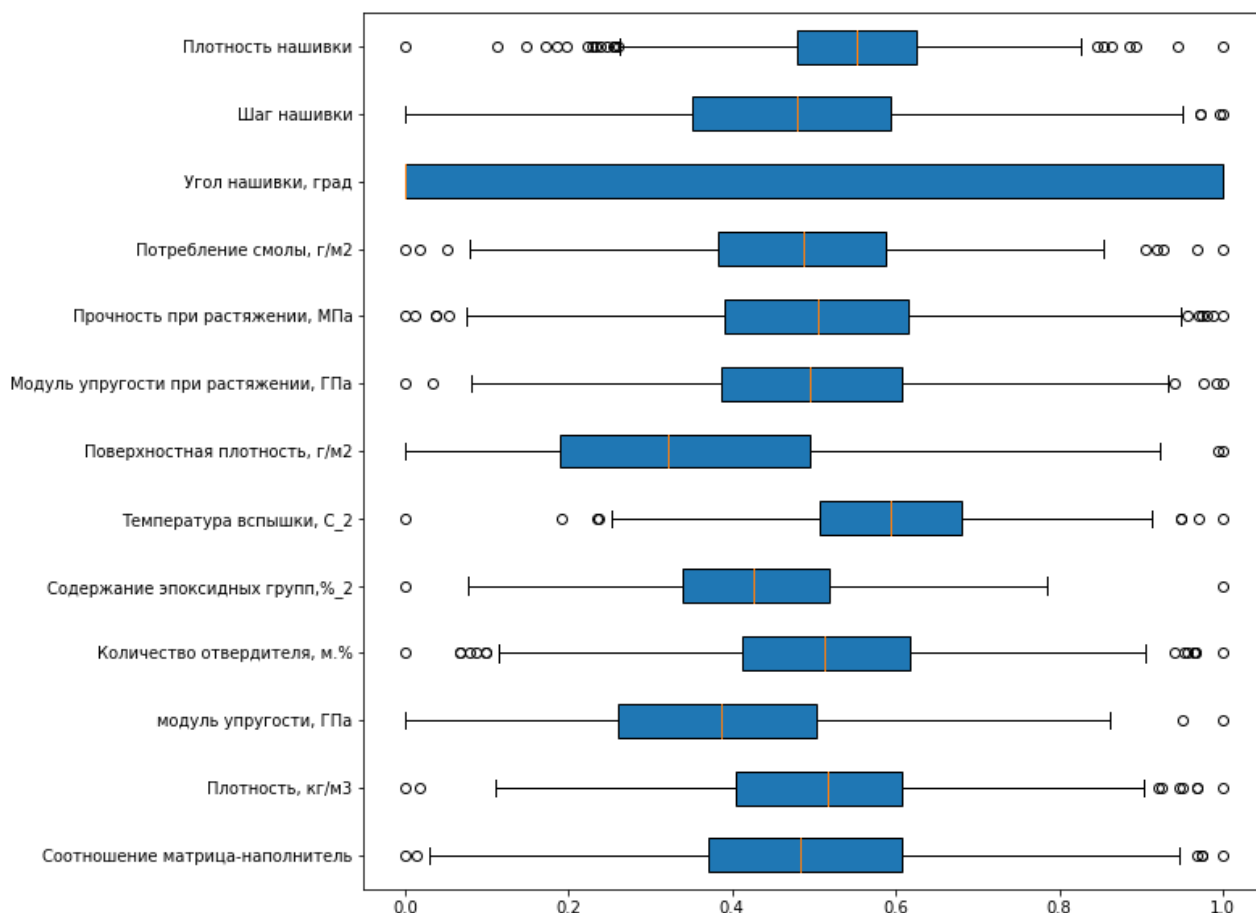


Рисунок 7 – Диаграммы размаха «Ящик с усами»

По диаграммам размаха видно, что по каждой характеристике, кроме «Угол нашивки, град», имеется наличие значений, находящихся за пределами полутора межквартильных расстояний от первого и третьего квартилей. Избавиться от них нам поможет «правило трех сигм» (Рисунок 8). Посчитаем и удалим выбросы (Рисунок 9).

```
# Пробуем межквартильное расстояние. Вычислим межквартильный диапазон и заменим значения вне диапазона NaN
for x in df.columns:
    q75, q25 = np.percentile(df.loc[:,x], [75,25])
    qr = q75-q25

    max = q75+(1.5*qr)
    min = q25-(1.5*qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
```

Рисунок 8 – Используем «правило трех сигм»

```
# Смотрим сколько выбросов получилось по каждому столбцу
df.isnull().sum()
```

```
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
dtype: int64
```

```
# Удаляем выбросы
df = df.dropna(axis = 0)
```

```
# Проверяем удаление
df.isnull().sum()
```

```
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          0
Потребление смолы, г/м2               0
Угол нашивки, град                     0
Шаг нашивки                            0
Плотность нашивки                      0
dtype: int64
```

Рисунок 9 – Расчет и удаление выбросов

Построим матрицу корреляции (Рисунок 10), посмотрим, какие линейные связи есть между не стандартизированными и не нормализованными данными

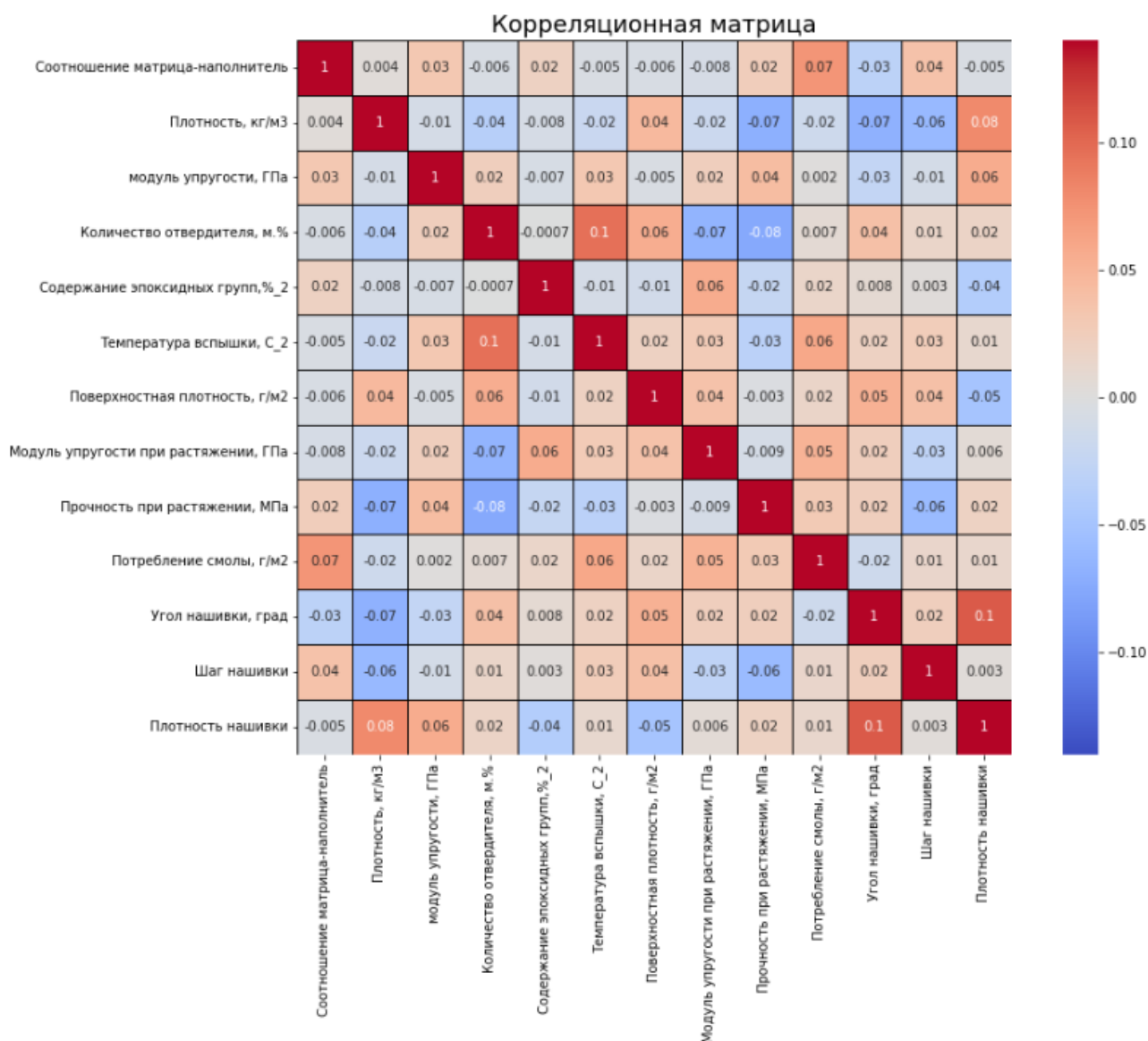


Рисунок 10 – Матрица корреляции

Видим очень слабую линейную зависимость. Максимальная положительная связь (0,1) есть между углом нашивки и поверхностной плотностью, что скорее всего, если угол нашивки равен 0° , то поверхностная плотность ниже, если угол нашивки 90° градусов, то поверхностная плотность выше.

Также есть положительная связь 0,1 между количеством отвердителя и температурой вспышки, т.е. если температура выше, то количество отвердителя выше.

Строим матрицу рассеяния scatter_matrix (Рисунок 11)

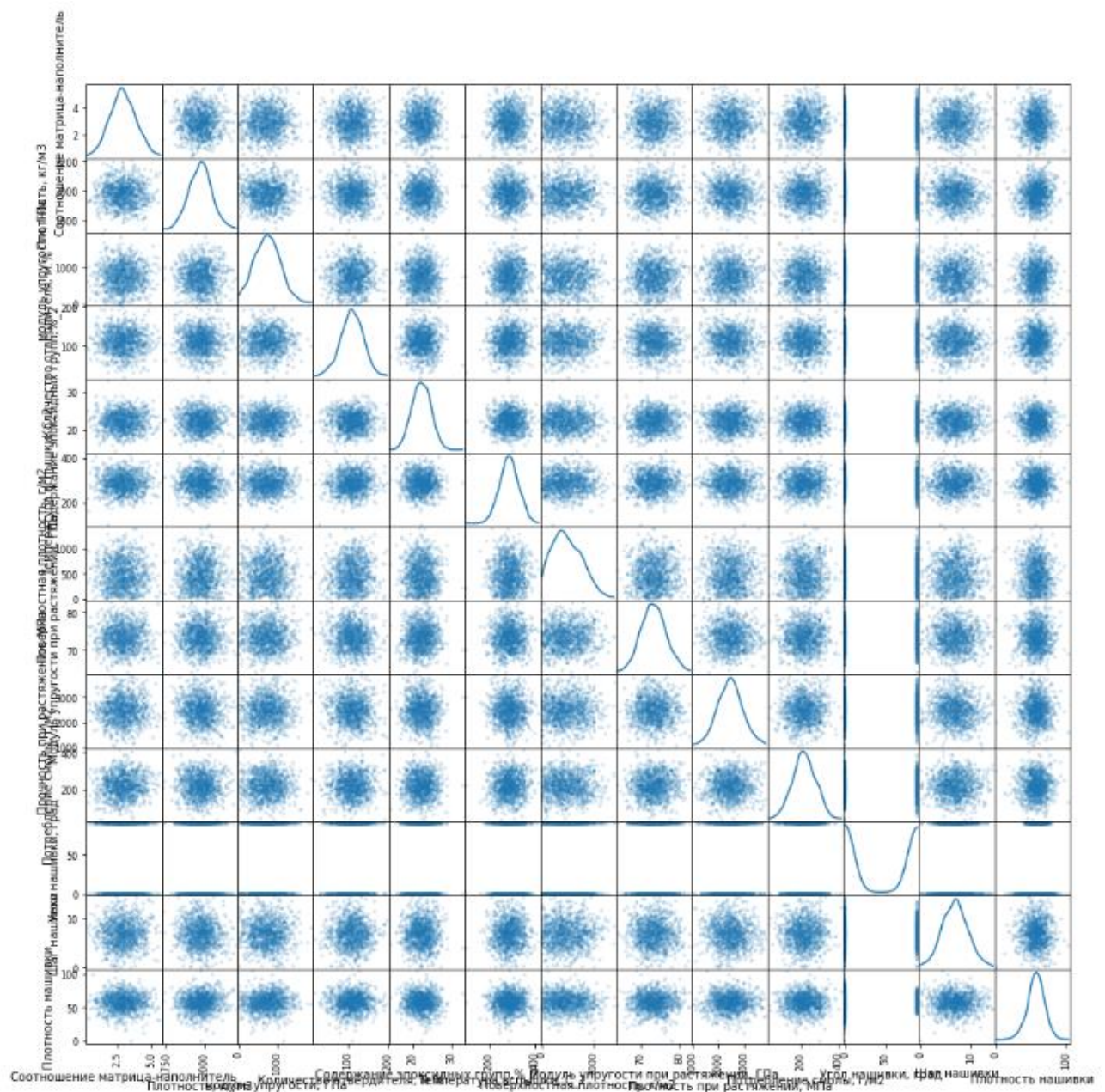


Рисунок 11 – Матрица рассеяния

Попарные графики рассеяния также не показывают линейной взаимосвязи между данными.

3. Предобработка данных

В ходе анализа данных было выполнено удаление выбросов, произведен поиск пропусков и дубликатов, на данном этапе проведем нормализацию и стандартизацию данных.

Выполним нормализацию с помощью метода MinMaxScaler. (Рисунок 12).

	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
0	0.274768	0.651097	0.447061	0.079153	0.607435	0.509184	0.162230	0.280303	0.712590	0.529221	0.0
1	0.274768	0.651097	0.447061	0.630983	0.418887	0.583596	0.162230	0.280303	0.712590	0.529221	0.0
2	0.466552	0.651097	0.455721	0.511257	0.495653	0.509184	0.162230	0.280303	0.712590	0.529221	0.0
3	0.466536	0.571539	0.452685	0.511257	0.495653	0.509184	0.162230	0.280303	0.712590	0.529221	0.0
4	0.424236	0.332865	0.486508	0.511257	0.495653	0.509184	0.162230	0.280303	0.712590	0.529221	0.0
...
931	0.361662	0.444480	0.552781	0.337550	0.333908	0.703458	0.161609	0.475147	0.463043	0.207613	1.0
932	0.607674	0.704373	0.268550	0.749605	0.294428	0.362087	0.271207	0.464422	0.452087	0.182974	1.0
933	0.573391	0.498274	0.251612	0.501991	0.623085	0.334063	0.572959	0.578740	0.575296	0.585446	1.0
934	0.662497	0.748688	0.448724	0.717585	0.267818	0.466417	0.496511	0.535142	0.334513	0.451779	1.0
935	0.664036	0.280923	0.251903	0.632264	0.888354	0.588206	0.587373	0.551972	0.654075	0.443749	1.0

936 rows x 13 columns

Рисунок 12 – Нормализация данных

Выполним стандартизацию данных с помощью StandardScaler (Рисунок 13)

	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
0	-1.196260	0.790727	0.001489	-2.254199	0.643790	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787
1	-1.196260	0.790727	0.001489	0.669189	-0.400666	0.354488	-0.974837	-1.088732	1.148666	0.041294	-1.023787
2	-0.172802	0.790727	0.044904	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787
3	-0.176623	0.368820	0.029685	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787
4	-0.398622	-0.904900	0.209271	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787
...
931	-0.732548	-0.310188	0.531478	-0.885307	-0.871403	0.983609	-0.977698	-0.070548	-0.172989	-1.602276	0.976766
932	0.580295	1.074592	-0.893411	1.297605	-1.090103	-0.808159	-0.472549	-0.126591	-0.231014	-1.728193	0.976766
933	0.397344	-0.023557	-0.976322	-0.014163	0.730480	-0.955250	0.918256	0.470784	0.421529	0.328626	0.976766
934	0.872860	1.310716	0.009825	1.127974	-1.237507	-0.260555	0.565897	0.242965	-0.853711	-0.354474	0.976766
935	0.987800	-1.181662	-0.976862	0.675976	2.199930	0.378680	0.984693	0.330915	0.838757	-0.395510	0.976766

936 rows x 13 columns

Рисунок 13 – Стандартизация данных

После анализа и предобработки данных получаем датасет df_standart с 936 уникальными строками.

4. Обучение моделей для прогноза модуля упругости при растяжении и прочности при растяжении.

4.1. Разбиение датасета для дальнейшего обучения моделей.

Для обучения моделей прогноза параметров разобьем ранее подготовленный датасет следующим образом:

- Удаляем столбцы "Модуль упругости при растяжении" и "Прочность при растяжении". Назовем новый датасет "mu" и "pr".
- Запишем переменную "Модуль упругости при растяжении" в "mu_Y", а датасет "mu" с удаленными столбцами в "mu_X".
- Запишем переменную "Прочность при растяжении" в "pr_Y", а датасет "pr" с удаленными столбцами в "pr_X".

```
mu = df_standart.drop(['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'], axis = 1)
pr = df_standart.drop(['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'], axis = 1)
mu
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	7.248098	10813.703381	3701.144201	282.207503	128.841077	1491.164899	986.189146	1121.640029	-1.023787	19.098805	311.055804
1	7.248098	10813.703381	3701.144201	680.719519	114.992429	1571.914821	986.189146	1121.640029	-1.023787	24.556516	243.078293
2	12.126673	10813.703381	3772.647266	589.918304	120.630807	1491.164899	986.189146	1121.640029	-1.023787	24.556516	295.368888
3	12.108480	10853.855354	3747.581616	589.918304	120.630807	1491.164899	986.189146	1121.640029	-1.023787	24.556516	311.055804
4	11.050241	10174.311331	4043.356291	589.918304	120.630807	1491.164899	986.189146	1121.640029	-1.023787	24.556516	363.346198
...
931	9.458500	10398.588904	4574.023908	458.178046	108.750837	1701.950251	982.495870	838.184810	0.976766	46.812375	243.181673
932	15.716497	10920.743825	2227.262859	770.683594	105.851041	1331.603138	1614.509459	598.992810	0.976766	54.943161	278.378438
933	14.844416	10506.850805	2087.416030	582.890891	129.990523	1301.200650	3409.672097	1208.508392	0.976766	19.978659	350.951723
934	17.111081	11009.782373	3714.873533	746.399220	103.896587	1444.789376	2954.870482	1004.743489	0.976766	31.726213	301.962891
935	17.658975	10089.949080	2089.820871	681.891189	149.474248	1576.915341	3495.424886	992.622864	0.976766	30.447007	402.221323

936 rows x 11 columns

```
mu_X = mu
mu_Y = df_standart['Модуль упругости при растяжении, ГПа']
pr_X = pr
pr_Y = df_standart['Прочность при растяжении, МПа']
```

Рисунок 14 – Разбивка датасета и запись в переменные

Разобьем датасет mu_X и датасет pr_X на тестовую и тренировочную выборки, 30 процентов данных оставим на тестирование модели, на остальных происходит обучение моделей.

```
# mu делим на тестовую и тренировочную выборки, зависящая mu_Y - Модуль упругости при растяжении#
mu_X_train, mu_X_test, mu_Y_train, mu_Y_test = train_test_split(mu_X, mu_Y, test_size = 0.30, random_state=1)

# pr делим на тестовую и тренировочную выборки, зависящая pr_Y Прочность при растяжении
pr_X_train, pr_X_test, pr_Y_train, pr_Y_test = train_test_split(pr_X, pr_Y, test_size = 0.30, random_state=1)
```

Рисунок 15 – Разбивка датасетов на тренировочную и обучающую выборки

Пишем функцию, которая рассчитывает среднее значение по тестовой выборке. Со средним будем сравнивать результаты предсказаний моделей (Рисунок 16)


```
def mean_model(mu_Y_test):
    return [np.mean(mu_Y_test) for _ in range(len(mu_Y_test))]
mu_Y_pred_mean = mean_model(mu_Y_test)
mean_absolute_error(mu_Y_test, mu_Y_pred_mean)

0.8289382478364875
```

```
def mean_model(pr_Y_test):
    return [np.mean(pr_Y_test) for _ in range(len(pr_Y_test))]
pr_Y_pred_mean = mean_model(pr_Y_test)
mean_absolute_error(pr_Y_test, pr_Y_pred_mean)

0.7880988878953844
```

Рисунок 16 – MAE для модели среднего значения.

4.2. Обучение моделей линейной регрессии.

Построим модель линейной регрессии для модуля упругости при растяжении (Рисунок 17)

```
lin_reg = LinearRegression()
lin_reg.fit(mu_X_train, mu_Y_train)
```

```
# Score для тренировочной выборки
print("Train score:", lin_reg.score(mu_X_train, mu_Y_train))
```

Train score: 0.021099855832543812

```
# Score для тестовой выборки
print("Test score:", lin_reg.score(mu_X_test, mu_Y_test))
```

Test score: -0.019301402088411246

```
# Предсказание значения для mu
mu_Y_pred = lin_reg.predict(mu_X_test).round(3)
mu_lin = pd.DataFrame({'Actual': mu_Y_test, 'Predicted': mu_Y_pred})
mu_lin.head()
```

	Actual	Predicted
386	-1.384355	0.109
41	0.873854	-0.026
725	-0.528017	0.081
605	0.504928	0.130
35	-0.801519	-0.072

```
# Результаты модели
mu_mse_lin_elast = mean_squared_error(mu_Y_test, mu_Y_pred)
print("MAE: ", mean_absolute_error(mu_Y_test, mu_Y_pred))
print("MSE: ", mu_mse_lin_elast)
print("RMSE: ", np.sqrt(mu_mse_lin_elast))
```

MAE: 0.8385264893496368
MSE: 1.0347000196871035
RMSE: 1.0172020545039728

Рисунок 17 - Модель линейной регрессии для модуля упругости при растяжении



Рисунок 18 – Разброс точек, предсказанных и фактических данных

Видим, что разброс у фактических данных большой – линейная регрессия не очень хорошо прогнозирует зависимости.

```
lin_mu = GSCV_lin_mu.best_estimator_
print(f'R2-score линейной регрессии модуль упругости при растяжении: {lin_mu.score(mu_X_test, mu_Y_test).round(3)}')
```

R2-score линейной регрессии модуль упругости при растяжении: -0.019

Рисунок 18 – R2-score

Для линейной регрессии – MSE 1.035, R-score -0.019. Для модели, что использует среднее, MAE – 0.83, что говорит, что линейная регрессия плохо предсказывает поведение зависимой переменной от входных данных.

```
def mean_model(mu_Y_test):
    return [np.mean(mu_Y_test) for _ in range(len(mu_Y_test))]
mu_Y_pred_mean = mean_model(mu_Y_test)
mean_absolute_error(mu_Y_test, mu_Y_pred_mean)
```

0.8289382478364875

Рисунок 19 – MAE средняя

Построим линейную регрессию для Прочности при растяжении (Рисунок 20).

```
lin_reg2= LinearRegression()
lin_reg2.fit(PPR_X_train, PPR_y_train)

LinearRegression()

#score для тренировочной выборки
print("Train score:", lin_reg2.score(PPR_X_train, PPR_y_train))

Train score: 0.025484538437679283

#Score для тестовой выборки
print("Test score:", lin_reg2.score(PPR_X_test, PPR_y_test))

Test score: -0.018288843718546444

Результат линейной регрессии PPR:

# Предсказание значения для PPR
PPR_y_pred = lin_reg2.predict(PPR_X_test).round(2)
PPR_lin = pd.DataFrame({'Actual': PPR_y_test, 'Predicted': PPR_y_pred})
PPR_lin.head(4)
```

	Actual	Predicted
Index		
431	-1.880558	0.01
45	-1.785143	0.03
803	0.488423	0.05
674	-0.884391	0.48

```
#результаты модели

PPR_mse_lin_elast = mean_squared_error(PPR_y_test, PPR_y_pred)
print("MAE: ", mean_absolute_error(PPR_y_test, PPR_y_pred))
print("MSE: ", PPR_mse_lin_elast)
print("RMSE: ", np.sqrt(PPR_mse_lin_elast))

MAE: 0.7975565769023175
MSE: 0.9809694074478643
RMSE: 0.9904389973379806
```

Рисунок 20 – Модель линейной регрессии для прочности при растяжении



Рисунок 21 – Разброс точек фактических и прогнозных данных

Видим, что разброс у фактических данных большой – линейная регрессия не очень хорошо прогнозирует зависимости.

```
lin_pr = GSCV_lin_pr.best_estimator_  
print(f'R2-score линейной регрессии прочность при растяжении: {lin_pr.score(pr_X_test, pr_Y_test).round(3)}')
```

R2-score линейной регрессии прочность при растяжении: -0.018

Рисунок 22 – R2-score

Для линейной регрессии PPR - MAE 0.98, R-score -0.018. Для модели, что использует среднее, MAE – 0.79, что говорит, что линейная регрессия плохо предсказывает поведение зависимой переменной от входных данных.

```
def mean_model(pr_Y_test):  
    return [np.mean(pr_Y_test) for _ in range(len(pr_Y_test))]  
pr_Y_pred_mean = mean_model(pr_Y_test)  
mean_absolute_error (pr_Y_test, pr_Y_pred_mean)
```

0.7880988878953844

Рисунок 23 – MAE средняя

4.3. Обучение модели К-ближайших соседей (KNeighborsRegressor)

Построим модель К-ближайших соседей для модуля упругости при растяжении (Рисунок 24).

```
kn = KNeighborsRegressor()
kn_params = {'n_neighbors' : range(1, 301, 2),
             'weights' : ['uniform', 'distance'],
             'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']}
GSCV_kn_mu = GridSearchCV(kn, kn_params, n_jobs=-1, cv=10)
GSCV_kn_mu.fit(mu_X_train, mu_Y_train)
GSCV_kn_mu.best_params_

{'algorithm': 'auto', 'n_neighbors': 71, 'weights': 'uniform'}

kn_mu = GSCV_kn_mu.best_estimator_
print(f'R2-score KNR для модуля упругости при растяжении: {kn_mu.score(mu_X_test, mu_Y_test).round(3)}')

R2-score KNR для модуля упругости при растяжении: -0.015

kn_mu_result = pd.DataFrame({
    'Model': 'KNeighborsRegressor_mu',
    'MAE': mean_absolute_error(mu_Y_test, kn_mu.predict(mu_X_test)),
    'R2 score': kn_mu.score(mu_X_test, mu_Y_test).round(3)
}, index=['Модуль упругости при растяжении'])

models3 = models2.append(kn_mu_result)
models3
```

	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_mu	0.838530	-0.019
Прочность при растяжении	LinearRegression_pr	0.797591	-0.018
Модуль упругости при растяжении	KNeighborsRegressor_mu	0.840831	-0.015

Рисунок 24 – Модель К-ближайших соседей для модуля упругости при растяжении

Построим модель К-ближайших соседей для прочности при растяжении (Рисунок 25).

```
GSCV_kn_pr = GridSearchCV(kn, kn_params, n_jobs=-1, cv=10)
GSCV_kn_pr.fit(pr_X_train, pr_Y_train)
GSCV_kn_pr.best_params_

{'algorithm': 'brute', 'n_neighbors': 235, 'weights': 'distance'}
```

```
kn_pr = GSCV_kn_pr.best_estimator_
print(f'R2-score KNR для прочности при растяжении: {kn_pr.score(pr_X_test, pr_Y_test).round(3)}')

R2-score KNR для прочности при растяжении: -0.009
```

```
kn_pr_result = pd.DataFrame({
    'Model': 'KNeighborsRegressor_pr',
    'MAE': mean_absolute_error(pr_Y_test, kn_pr.predict(pr_X_test)),
    'R2 score': kn_pr.score(pr_X_test, pr_Y_test).round(3)
}, index=['Прочность при растяжении'])

models4 = models3.append(kn_pr_result)
models4
```

	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_mu	0.838530	-0.019
Прочность при растяжении	LinearRegression_pr	0.797591	-0.018
Модуль упругости при растяжении	KNeighborsRegressor_mu	0.840831	-0.015
Прочность при растяжении	KNeighborsRegressor_pr	0.792277	-0.009

Рисунок 25 – Модель К-ближайших соседей для прочности при растяжении

Если коэффициент детерминации R^2 – score равен нулю, то это говорит о том, что модель прогнозирует с таким же результатом, что, если бы мы взяли среднее значение прогнозируемого параметра. Если R^2 – score меньше 0, это значит, что разработанная модель дает прогноз хуже, чем обычное усреднение.

4.4. Обучение модели «Случайного леса» (Random Forest Regressor). Подбор параметров.

Построим модель случайного леса для модуля упругости при растяжении (Рисунок 26).

```
# Создаем сетку параметров на основе случайного поиска
# Создаем модель поиска по сетке с перекрестной проверкой, количество блоков равно 10
rf = RandomForestRegressor()
rf_param = {
    'n_estimators': range(10, 1000, 10),
    'criterion': ['squared_error', 'absolute_error', 'poisson'],
    'max_depth': range(1, 7),
    'min_samples_split': range(20, 50, 5),
    'min_samples_leaf': range(2, 8),
    'bootstrap': ['True', 'False']
}
mu_rf = RandomizedSearchCV(rf, rf_param, n_jobs=-1, cv=10, verbose=4)
# Обучаем модель
mu_rf.fit(mu_X_train, mu_Y_train)
# Ищем лучшие параметры для модели
mu_rf.best_params_

Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

```
{'n_estimators': 220,
 'min_samples_split': 40,
 'min_samples_leaf': 5,
 'max_depth': 1,
 'criterion': 'absolute_error',
 'bootstrap': 'True'}
```

```
#Предсказываем значения
mu_rf_pred = mu_rf.predict(mu_X_test).round(3)

mu_grid_rf = mu_rf.best_estimator_
print(f'R2-score RFRegr для модуля упругости при растяжении: {mu_grid_rf.score(mu_X_test, mu_Y_test).round(3)}')

R2-score RFRegr для модуля упругости при растяжении: -0.006

mu_rf_result = pd.DataFrame({
    'Model': 'Random Forest Regressor_mu',
    'MAE': mean_absolute_error(mu_Y_test, mu_grid_rf.predict(mu_X_test)),
    'R2_score': mu_grid_rf.score(mu_X_test, mu_Y_test).round(3)
}, index=['Модуль упругости при растяжении'])

models5 = models4.append(mu_rf_result)
models5
```

Рисунок 26 – Модель «Случайного леса» для модуля упругости при растяжении
Построим модель случайного леса для прочности при растяжении (Рисунок 27).

```
# Создаем сетку параметров на основе случайного поиска
# Создаем модель поиска по сетке с перекрестной проверкой, количество блоков равно 10
rf = RandomForestRegressor()
rf_param = {
    'n_estimators': range(10, 1000, 10),
    'criterion': ['squared_error', 'absolute_error', 'poisson'],
    'max_depth': range(1, 7),
    'min_samples_split': range(20, 50, 5),
    'min_samples_leaf': range(2, 8),
    'bootstrap': ['True', 'False']
}
pr_rf = RandomizedSearchCV(rf, rf_param, n_jobs=-1, cv=10, verbose=4)
# Обучаем модель
pr_rf.fit(pr_X_train, pr_Y_train)
# Ищем лучшие параметры для модели
pr_rf.best_params_

{'n_estimators': 250,
 'min_samples_split': 45,
 'min_samples_leaf': 5,
 'max_depth': 1,
 'criterion': 'absolute_error',
 'bootstrap': 'True'}

Fitting 10 folds for each of 10 candidates, totalling 100 fits

# Предсказываем значения
pr_rf_pred = pr_rf.predict(pr_X_test).round(3)

pr_rf_grid = pr_rf.best_estimator_
print(f'R2-score RFRegr для прочности при растяжении: {pr_rf_grid.score(pr_X_test, pr_Y_test).round(3)}')

R2-score RFRegr для прочности при растяжении: -0.004

pr_rf_result = pd.DataFrame({
    'Model': 'Random Forest Regressor_pr',
    'MAE': mean_absolute_error(pr_Y_test, pr_rf.predict(pr_X_test)),
    'R2_score': pr_rf_grid.score(pr_X_test, pr_Y_test).round(3)
}, index=['Прочность при растяжении'])

models6 = models5.append(pr_rf_result)
models6
```

Рисунок 27 – Модель «Случайного леса» для прочности при растяжении
Все построенные модели дают прогноз хуже, чем обычное усреднение (Рисунок 27).

	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_mu	0.838530	-0.019
Прочность при растяжении	LinearRegression_pr	0.797591	-0.018
Модуль упругости при растяжении	KNeighborsRegressor_mu	0.840831	-0.015
Прочность при растяжении	KNeighborsRegressor_pr	0.792277	-0.009
Модуль упругости при растяжении	Random Forest Regressor_mu	0.832147	-0.006
Прочность при растяжении	Random Forest Regressor_pr	0.791913	-0.004

Рисунок 28 – Сравнение моделей

Построенные модели прогнозирования не демонстрируют приемлемого уровня точности, и свидетельствуют о слабой линейной связи между данными датасета. Об этом на этапе разведывательного анализа данных говорит слабая попарная корреляция выходных данных с входными (матрица корреляции). Однако разведывательный анализ данных и теоретическая база композитных материалов также позволяет делать допущение, что связь возможна, но для ее обнаружения необходимо использовать более сложные способы исследования.

Также можно провести дополнительные способы разбивки данных на более малые выборки, уменьшение размерности. Разведочный анализ данных показывает, что Угол нашивки равен 0 градусов (520 наблюдений исходной выборки), а другая часть Угол нашивки равен 90 градусов (520 наблюдений исходной выборки). Так как данный входной параметр четко делится на две подгруппы, то имеет смысл, опираясь на теорию, рассмотреть модели для каждой из этих подвыборок. Максимальная положительная связь (0,1) есть между Углом нашивки и Поверхностной плотностью, что скорее всего, если Угол нашивки 0, то Поверхностная плотность ниже, если Угол нашивки 90 градусов, то Поверхностная плотность выше. Также есть положительная связь (0,1) между Количеством отвердителя и Температурой вспышки, т.е. если температура выше, значит, и Количество отвердителя выше. В теории, если Температура вспышки выше, то, нужно большее Количество отвердителя (как канифоль). Ничего нового, кроме обычной физики, данная связь не показывает.

Кроме того, теоретические исследования в области композитных материалов позволяют предположить, что не хватает дополнительных параметров (например, коэффициент Пуассона или данных об угле и направлении, под которыми располагают нити накладываемых слоёв), что также влияет на прочность при растяжении и модуль упругости при растяжении.

Из аналогичных исследований можно взять для тестирования подход решения задач моделирования композиционного материала с заданными электрофизическими свойствами (композиты на основе нанопористого кремния), на основе генетического алгоритма или рекуррентного типа нейронной сети (диэлектрическая проницаемость, проводимость).

5. Разработка нейронных сетей

5.1. Нейронные сети с одним выходом для параметра соотношение матрица-наполнитель.

Обобщающая способность (generalization) - способность сети работать на тех данных, что она не видела. Если нейронная сеть переобучена, то она адаптируется к данным, которые она видела (training set) и хорошо работает на них, но ее обобщающая способность снижается (плохо работает на данных, которые она не видела). Оценивают качество сети на тех, данных, которые она не видела (test set).

Если модель дает более стабильный результат, значит, данная модель является более лучшим решением.

Построим нейронную сеть с одним выходом для параметра соотношение матрица-наполнитель (Рисунок 29.1-29.3).

```
# удалим столбец "Соотношение матрица-наполнитель", запишем датасет с удаленным столбцом в mf_X.  
# Запишем переменную "Соотношение матрица-наполнитель" в mf_Y,  
  
mf_X = df_standart.drop(['Соотношение матрица-наполнитель'], axis = 1)  
mf_Y = df_standart['Соотношение матрица-наполнитель']  
  
# mf делим на тестовую и тренировочную выборки, зависимая mf_Y - соотношение матрица наполнитель.  
mf_X_train, mf_X_test, mf_Y_train, mf_Y_test = train_test_split(mf_X, mf_Y, test_size = 0.30, random_state=1)  
  
normalizer = tf.keras.layers.Normalization(axis=-1)  
mf_X_train_norm = normalizer.adapt(np.array(mf_X_train))  
  
mf_model_1 = Sequential(mf_X_train_norm)  
  
mf_model_1.add(Dense(128))  
mf_model_1.add(BatchNormalization())  
mf_model_1.add(LeakyReLU())  
mf_model_1.add(Dense(128, activation='selu'))  
mf_model_1.add(BatchNormalization())  
mf_model_1.add(Dense(64, activation='selu'))  
mf_model_1.add(BatchNormalization())  
mf_model_1.add(Dense(32, activation='selu'))  
mf_model_1.add(BatchNormalization())  
mf_model_1.add(LeakyReLU())  
mf_model_1.add(Dense(16, activation='selu'))  
mf_model_1.add(BatchNormalization())  
mf_model_1.add(Dense(1))  
mf_model_1.add(Activation('selu'))  
  
mf_early = EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1, mode='auto')  
  
mf_model_1.compile(  
    optimizer=tf.optimizers.SGD(learning_rate=0.02, momentum=0.5),  
    loss='mean_absolute_error')
```

Рисунок 29.1 - Нейронная сеть с одним выходом для соотношения матрица-наполнитель (модель 1)

```

%%time
history_mf_model_1 = mf_model_1.fit(
    mf_X_train,
    mf_Y_train,
    batch_size = 64,
    epochs=30,
    verbose=1,
    validation_split = 0.2,
    callbacks = [mf_early ]
)

Epoch 1/30
9/9 [=====] - 2s 33ms/step - loss: 1.1257 - val_loss: 0.9885
Epoch 2/30
9/9 [=====] - 0s 8ms/step - loss: 0.9302 - val_loss: 0.9739
Epoch 3/30
9/9 [=====] - 0s 7ms/step - loss: 0.8387 - val_loss: 0.8746
Epoch 4/30
9/9 [=====] - 0s 7ms/step - loss: 0.7772 - val_loss: 0.8829
Epoch 5/30
9/9 [=====] - 0s 6ms/step - loss: 0.7636 - val_loss: 0.8783
Epoch 6/30
9/9 [=====] - 0s 7ms/step - loss: 0.7339 - val_loss: 0.8969
Epoch 7/30
9/9 [=====] - 0s 6ms/step - loss: 0.7147 - val_loss: 0.9107
Epoch 8/30
9/9 [=====] - 0s 7ms/step - loss: 0.7017 - val_loss: 0.8668
Epoch 9/30
9/9 [=====] - 0s 6ms/step - loss: 0.6971 - val_loss: 0.8988
Epoch 10/30
9/9 [=====] - 0s 7ms/step - loss: 0.6871 - val_loss: 0.8923
Epoch 11/30
9/9 [=====] - 0s 7ms/step - loss: 0.6690 - val_loss: 0.8718
Epoch 12/30
9/9 [=====] - 0s 7ms/step - loss: 0.6657 - val_loss: 0.8838
Epoch 13/30
9/9 [=====] - 0s 7ms/step - loss: 0.6792 - val_loss: 0.9063
Epoch 14/30
9/9 [=====] - 0s 7ms/step - loss: 0.6546 - val_loss: 0.8790
Epoch 15/30
9/9 [=====] - 0s 6ms/step - loss: 0.6445 - val_loss: 0.9004
Epoch 16/30
9/9 [=====] - 0s 7ms/step - loss: 0.6441 - val_loss: 0.9114
Epoch 17/30
9/9 [=====] - 0s 6ms/step - loss: 0.6246 - val_loss: 0.9005
Epoch 18/30
9/9 [=====] - 0s 7ms/step - loss: 0.6191 - val_loss: 0.8952
Epoch 18: early stopping
Wall time: 2.8 s

```

Рисунок 29.2 - Нейронная сеть с одним выходом для соотношения матрица-наполнитель (модель 1)

```
mf_model_1.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1664
batch_normalization (Batch Normalization)	(None, 128)	512
leaky_re_lu (LeakyReLU)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
batch_normalization_1 (Batch Normalization)	(None, 128)	512
dense_2 (Dense)	(None, 64)	8256
batch_normalization_2 (Batch Normalization)	(None, 64)	256
dense_3 (Dense)	(None, 32)	2080
batch_normalization_3 (Batch Normalization)	(None, 32)	128
leaky_re_lu_1 (LeakyReLU)	(None, 32)	0
dense_4 (Dense)	(None, 16)	528
batch_normalization_4 (Batch Normalization)	(None, 16)	64
dense_5 (Dense)	(None, 1)	17
activation (Activation)	(None, 1)	0

=====
Total params: 30,529
Trainable params: 29,793
Non-trainable params: 736
=====

Рисунок 29.3 - Нейронная сеть с одним выходом для соотношения матрица-наполнитель (модель 1)

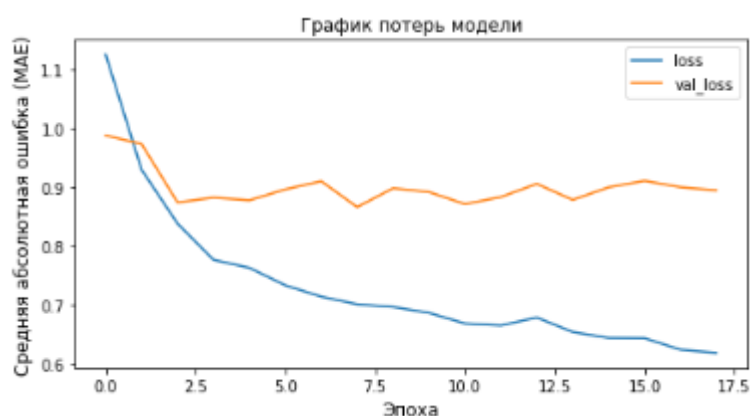


Рисунок 30 – График потерь модели 1

В примере видно (Рисунок 29-30), что после 3-й эпохи ($val_loss = 0.8746$) доля потерь на проверочной выборке начинает расти и к 13-й эпохе становится

равной 0,963. Это означает, что модель переобучена и обучение необходимо останавливать после второй эпохи. И данная структура модели очень сложная для данного датасета.

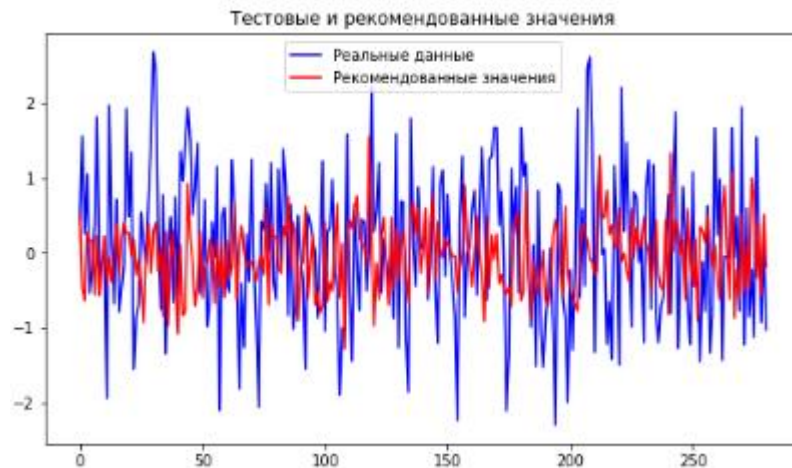


Рисунок 31 – Поведение тестовых и прогнозных значений



Рисунок 32 - Рассеяния тестовых и прогнозных значений

Метод `evaluate()` возвращает значения функции потерь и метрики для обученной модели.

```
mf_model_1.evaluate(mf_X_test, mf_Y_test, verbose=1)
9/9 [=====] - 0s 2ms/step - loss: 0.9214
0.9213865995407104

np.mean(np.abs(mf_Y_test-np.mean(mf_Y_test)))
0.8303402063681543
```

Нейронная сеть (Модель 1) хуже ($\text{loss} = 0.921$), чем средняя предсказывает поведение зависимой переменной ($\text{MAE} = 0.83$).

Сохраним модель


```
mf_model_1.save('models/mf_model_1')
```

```
INFO:tensorflow:Assets written to: models/mf_model_1/assets
```

```
mf_model_1.save_weights('models/mf_model_1/weights')
```

Построим нейронную сеть попроще с одним выходом для параметра соотношение матрица-наполнитель (Рисунок 33)

```

mf_model_2 = Sequential(
[Dense(12, activation = 'relu', input_dim=12),
 Dense(64, activation = 'relu'),
 Dense(64, activation = 'relu'),
 Dense(1),
])

mf_early_2= EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1, mode='auto')

mf_model_2.compile(
    optimizer='adam',
    loss='mean_absolute_error')

%%time
history_mf_model_2 = mf_model_2.fit(
    mf_X_train,
    mf_Y_train,
    epochs=50,
    verbose=1,
    validation_split = 0.2,
    callbacks = [mf_early_2]
)

Epoch 1/50
17/17 [=====] - 1s 10ms/step - loss: 0.8071 - val_loss: 0.7882
Epoch 2/50
17/17 [=====] - 0s 4ms/step - loss: 0.7828 - val_loss: 0.7749
Epoch 3/50
17/17 [=====] - 0s 4ms/step - loss: 0.7767 - val_loss: 0.7922
Epoch 4/50
17/17 [=====] - 0s 4ms/step - loss: 0.7652 - val_loss: 0.7866
Epoch 5/50
17/17 [=====] - 0s 4ms/step - loss: 0.7589 - val_loss: 0.7929
Epoch 6/50
17/17 [=====] - 0s 4ms/step - loss: 0.7538 - val_loss: 0.7988
Epoch 7/50
17/17 [=====] - 0s 4ms/step - loss: 0.7434 - val_loss: 0.7925
Epoch 8/50
17/17 [=====] - 0s 4ms/step - loss: 0.7366 - val_loss: 0.8037
Epoch 9/50
17/17 [=====] - 0s 4ms/step - loss: 0.7297 - val_loss: 0.8110
Epoch 10/50
17/17 [=====] - 0s 4ms/step - loss: 0.7210 - val_loss: 0.8056
Epoch 11/50
17/17 [=====] - 0s 4ms/step - loss: 0.7138 - val_loss: 0.8104
Epoch 12/50
17/17 [=====] - 0s 4ms/step - loss: 0.7047 - val_loss: 0.8119
Epoch 12: early stopping
Wall time: 1.36 s

mf_model_2.summary()

Model: "sequential_1"

```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 12)	156
dense_7 (Dense)	(None, 64)	832
dense_8 (Dense)	(None, 64)	4160
dense_9 (Dense)	(None, 1)	65

```

=====
Total params: 5,213
Trainable params: 5,213
Non-trainable params: 0

```

Рисунок 33 - Нейронная сеть с одним выходом для соотношения матрица-наполнитель (модель 2)

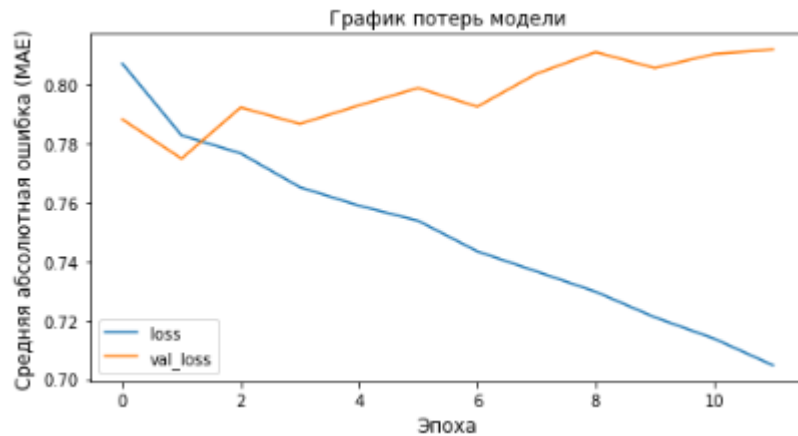


Рисунок 34 - График потерь модели 2

В примере видно, что после 4-й эпохи ($\text{val_loss} = 0.786$) доля потерь на проверочной выборке начинает расти и к 12-й эпохе становится равной 0,8119.

Это означает, что модель переобучена и обучение необходимо останавливать после четвертой эпохи. И данная структура модели очень сложная для данного датасета.

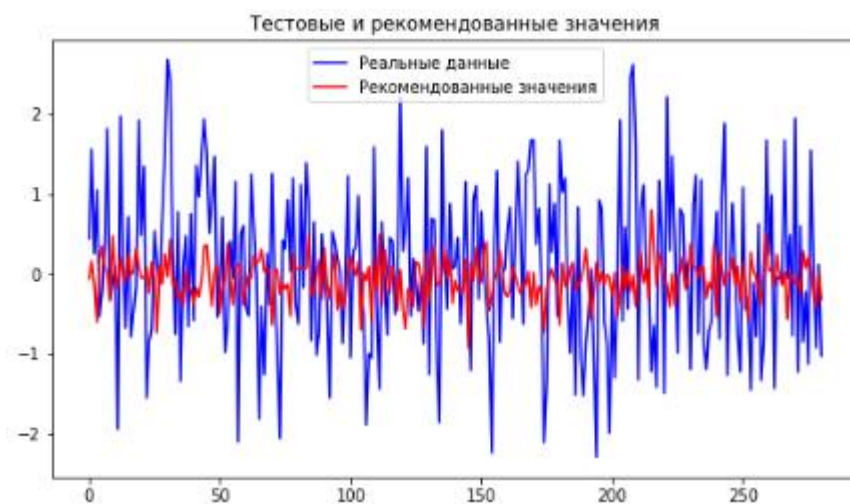


Рисунок 35 – Поведение тестовых и прогнозных значений



Рисунок 36 - Рассеяния тестовых и прогнозных значений

```
mf_model_2.evaluate(mf_X_test, mf_Y_test, verbose=1)
9/9 [=====] - 0s 2ms/step - loss: 0.8655
0.8654528260231018

np.mean(np.abs(mf_Y_test-np.mean(mf_Y_test)))
0.8303402063681543
```

Нейронная сеть (Модель 2) чуть лучше ($\text{loss}=0.87$) предсказывает поведение зависимой переменной, чем (Модель 1), но снова хуже, чем обычная средняя ($\text{MAE}=0.83$).

Сохраним модель

```
mf_model_2.save('models/mf_model_2')
INFO:tensorflow:Assets written to: models/mf_model_2/assets

mf_model_2.save_weights('models/mf_model_2/weights')
```

Сравнение качества моделей нейронных сетей, решающих задачу регрессии, проводится на показателе MSE/ MAE (на тестовой выборке). Если порядок ошибки примерно одинаковый для нескольких моделей, выбирается та сеть, которая проще по архитектуре.

Если на тренировочном наборе данных ошибка снижается, а на валидационном растет, то значит, наша модель переобучилась.

Для нахождения более удачной модели необходимо менять гиперпараметры обучения сети, а именно:

- Количество нейронов входного слоя. Количество нейронов задается при создании входного слоя: `model.add(Dense(XXX, ...))`. Использовать разные значения. Выбрать значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.
- Количество слоёв обучения (dance). Добавляем/ убираем скрытый слой. Можно добавить скрытый слой с разным количеством нейронов: 500, 700, 900 и 1200. Выбрать наиболее подходящее количество нейронов скрытого слоя. Оценить, как изменяется время обучения при добавлении/удалении скрытого слоя с разным количеством нейронов. В идеале, можно посмотреть сеть минимального размера, что может дать лучший результат.
- Количество эпох обучения (epochs). Можно обучать сеть в течение 50, 75, 100 и 125 эпох. Выбрать количество эпох, при котором самая высокая доля верных ответов нейросети на тестовых данных.

- Размер мини-выборки (`batch_size`). Использовать размер мини-выборки 50, 100, 200 и 400. Выбрать значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.
- Размер параметра скорости обучения. Он важен при углублении структуры сети.
- Архитектура нейронной сети; поменять активационную функцию.

Чтобы создать сеть с лучшими значениями всех гиперпараметров обучения, необходимо вести запись всех версий модели с изменёнными гиперпараметрами, которые определили на предыдущем шаге, и отслеживать увеличилась ли доля верных ответов нейросети.

Полученные результаты могут сократить сроки и издержки получения новых композитов с заданными характеристиками.

5.2. Построение нейронной сети с одним выходом для предсказания модуля упругости при растяжении

Построим нейронную сеть с одним выходом для модуля упругости при растяжении (Рисунок 37.1-37.2).

```
model_3 = Sequential(  
    [Dense(11, activation = 'relu', input_dim=11),  
      Dense(64, activation = 'relu'),  
      Dense(64, activation = 'relu'),  
      Dense(1),  
    ]  
)  
  
early_3= EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1, mode='auto')
```

```
model_3.compile(  
    optimizer='adam',  
    loss='mean_absolute_error')
```

```
%%time  
history_model_3 = model_3.fit(  
    mu_X_train,  
    mu_Y_train,  
    epochs=50,  
    verbose=1,  
    validation_split = 0.2,  
    callbacks = [early_3]  
)
```

```
Epoch 1/50  
17/17 [=====] - 1s 12ms/step - loss: 0.8081 - val_loss: 0.8038  
Epoch 2/50  
17/17 [=====] - 0s 7ms/step - loss: 0.7871 - val_loss: 0.8009  
Epoch 3/50  
17/17 [=====] - 0s 9ms/step - loss: 0.7741 - val_loss: 0.7923  
Epoch 4/50  
17/17 [=====] - 0s 8ms/step - loss: 0.7662 - val_loss: 0.7935  
Epoch 5/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7588 - val_loss: 0.7886  
Epoch 6/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7540 - val_loss: 0.7889  
Epoch 7/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7452 - val_loss: 0.7889  
Epoch 8/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7385 - val_loss: 0.7894  
Epoch 9/50  
17/17 [=====] - 0s 5ms/step - loss: 0.7316 - val_loss: 0.7898  
Epoch 10/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7255 - val_loss: 0.7896  
Epoch 11/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7182 - val_loss: 0.7968  
Epoch 12/50  
17/17 [=====] - 0s 8ms/step - loss: 0.7141 - val_loss: 0.8037  
Epoch 13/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7060 - val_loss: 0.8024  
Epoch 14/50  
17/17 [=====] - 0s 4ms/step - loss: 0.7001 - val_loss: 0.8036  
Epoch 15/50  
17/17 [=====] - 0s 3ms/step - loss: 0.6956 - val_loss: 0.8106  
Epoch 15: early stopping  
Wall time: 2 s
```

Рисунок 37.1 - Нейронная сеть с одним выходом для предсказания модуля упругости при растяжении

```
model_3.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
dense_18 (Dense)	(None, 11)	132
dense_19 (Dense)	(None, 64)	768
dense_20 (Dense)	(None, 64)	4160
dense_21 (Dense)	(None, 1)	65

=====
Total params: 5,125
Trainable params: 5,125
Non-trainable params: 0
=====

Рисунок 37.2 - Нейронная сеть с одним выходом для предсказания модуля упругости при растяжении

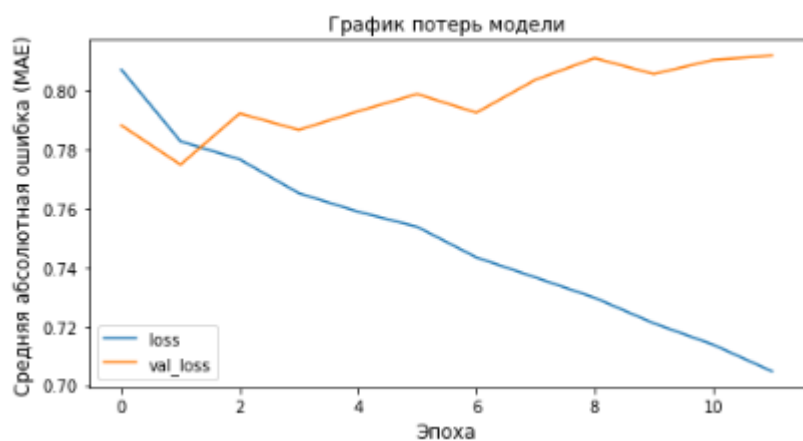


Рисунок 38 – График потерь модели

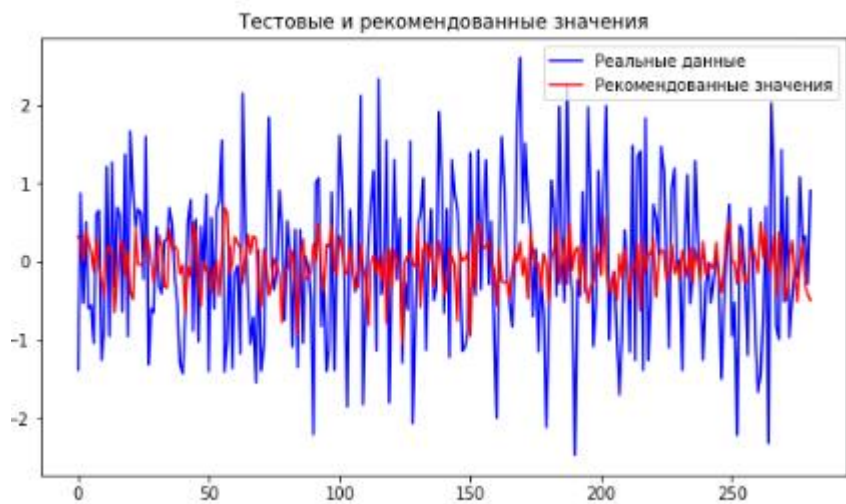


Рисунок 39 – Поведение тестовых и прогнозных значений



Рисунок 40 – Рассеяния тестовых и прогнозных значений

```
model_3.evaluate(mu_X_test, mu_Y_test, verbose=1)
9/9 [=====] - 0s 5ms/step - loss: 0.8393
0.8393213748931885

np.mean(np.abs(mu_Y_test-np.mean(mu_Y_test)))
0.8289382478364874
```

Нейронная сеть (loss=0.84) предсказывает поведение зависимой переменной хуже, чем обычная средняя (MAE =0.83), но лучше, чем предыдущие модели. Сохраним модель.

```
model_3.save('models/model_3')
INFO:tensorflow:Assets written to: models/model_3/assets

model_3.save_weights('models/model_3/weights')
```

Рисунок 41 – Сохранение модели

6. Создание приложения с графическим интерфейсом для предсказания модуля упругости

Расчет модуля упругости при растяжении

Введите параметры

Соотношение матрица-наполнитель, МПа

Плотность (кг/м³)

Модуль упругости (ГПа)

Количество отвердителя (%)

Содержание эпоксидных групп (%)

Температура вспышки (°C)

Поверхностная плотность (г/м²)

Потребление смолы (г/м²)

Угол нашивки (град)

Шаг нашивки

Плотность нашивки

Рисунок 41 – Flask-приложение

Ссылка на репозиторий GitHub <https://github.com/Whale58/BMSTU>

7. Заключение

Данные и модели показывают, что при получении большего объема данных, их более детального изучения и обработки, можно значительно улучшить модели или применить другие для предсказаний искомых параметров.

На момент написания работы мне не хватило знаний и времени для получения более качественных моделей и крайне не хватило времени на более детальное изучение и предобработку данных.

Курс «Data Science» был для меня очень полезен и в очередной раз я убедился, что при желании и достаточном количестве времени возможно освоить новую профессию. Данное обучение позволило мне приобрести необходимые навыки и заинтересовать меня для продолжения развития в данном направлении. Некоторые навыки, приобретенные на данном курсе, уже частично помогают мне в работе.

Список литературы

1. Язык программирования Python – Режим доступа: <https://www.python.org/>. (дата обращения 16.04.2022)
2. Библиотека Matplotlib – Режим доступа: <https://matplotlib.org/>. (дата обращения 16.05.2022)
3. Д. Фостер Генеративное глубокое обучение. Творческий потенциал нейронных сетей. - СПб.: Питер. - 2020. - 336 с. - ISBN: 978-5-4461-1566-2
4. С. Николенко, А. Кадури, Е. Архангельская Глубокое обучение. Погружение в мир нейронных сетей. - СПб.: Питер. - 2020. - 480 с. ISBN: 978-5-4461-1537-2
5. Андерсон, Карл Аналитическая культура. От сбора данных до бизнес-результатов / Карл Андерсон ; пер. с англ. Юлии Константиновой ; [науч. ред. Руслан Салахияев]. — М. : Манн, Иванов и Фербер, 2017. — 336 с
6. Билл Любанович. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O'Reilly»).
7. Аллен Б. Дауни – Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.
8. Библиотека Pandas – Режим доступа: <https://pandas.pydata.org/>. (дата обращения 16.05.2022)
9. Библиотека Sklearn- Режим доступа: <https://scikit-learn.org/stable/>. (дата обращения 16.5.2022)
10. Джулли, Пал: Библиотека Keras - инструмент глубокого обучения / пер. с англ. А. А. Слинкин.- ДМК Пресс, 2017. – 249 с.