ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу «Data Science»

ФИО Волотова Юлия Викторовна

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Структура работы

1. Прогнозирование конечных свойств новых материалов (композиционных материалов).
2. Исходные данные: 2 датасета с общим количеством параметров 13
3. Провести разведочный анализ данных
   Провести предобработку данных
4. Обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении
5. Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель

Компози́т — 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.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Анализ первичных данных

1. Предобработка данных

Основные задачи этапа:

Импортировать библиотеки;

Загрузить данные;

Объединить таблицы по индексу тип объединения INNER;

Исходные массивы содержит следующие данные:

FIRST (X_bp) Соотношение матрица-наполнитель

- Плотность, кг/м3, модуль упругости, ГПа
- Количество отвердителя, м.%, Содержание эпоксидных групп,
- Температура вспышки, С_2, Поверхностная плотность, г/м2
- Модуль упругости при растяжении, ГПа,
- Прочность при растяжении, МПа
- Потребление смолы, г/м2

SECOND (X_nup)

- Угол нашивки, град,
- Шаг нашивки
- Плотность нашивки

```
# Импорт библиотек для первичной обработки
import numpy as np
import pandas as pd
from sklearn import preprocessing
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
print(tf.__version__)
```

```
2.12.0
```

```
[3]  #GoogleDrive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[4]  #загрузка данных, наименование столбцов
     df_first = pd.read_excel("/content/drive/MyDrive/ВКР_Волотова/X_bp.xlsx"
     df_first
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Анализ первичных данных

Во время обработки данных были проведены следующие манипуляции

- загрузка данных из внешнего хранилища (для каждой таблицы)

  from google.colab import drive | drive.mount('/content/drive')

  df_first = pd.read_excel("/content/drive/MyDrive/ВКР_Волотова/X_bp.xlsx",index_col=0) | df_first
- сброс индекса (для каждой таблицы) и удаление значений без индексов (приведение к общему количеству)
  - df_first=df_first.reset_index(drop=True) | df
- объединение данных в одну таблицу методом INNER

  df = df_first.merge(df_second, left_index=True,right_index=True, how='inner') | df | df = df_second.join(df_first, how='inner'))
- проверка на уникальность, дублирование строк df.drop_duplicates ()
- проверка на пропуски df.isna().sum()
- вывод информации по массиву

	Угол нашивки, град	Шаг нашивки	Плотность нашивки	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	уп
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023
mean	44.252199	6.899222	57.153929	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	482.731833	73
std	45.015793	2.563467	12.350969	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3
min	0.000000	0.000000	0.000000	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64
25%	0.000000	5.080033	49.799212	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71
50%	0.000000	6.916144	57.341920	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73
75%	90.000000	8.586293	64.944961	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75
max	90.000000	14.440522	103.988901	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Анализ первичных данных

Было принято решение ввести вычисляемый параметр «удельная прочность (модуль)=модуль упругости/плотность» косвенный показатель, характеризующий прочность материала. Предполагалось посмотреть по нему зависимость между остальными параметрами.

df['Удельная прочность']=df['модуль упругости, ГПа']/df['Плотность, кг/м3'] | df

В итоге все показатели имеют очень низкую взаимосвязь друг с другом.

	Угол нашивки, град	Шаг нашивки	Плотность нашивки	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	Удельная прочность
Угол нашивки, град	1.000000	0.023616	0.107947	-0.031073	-0.068474	-0.025417	0.038570	0.008052	0.020695	0.052299	0.023003	0.023398	-0.015334	-0.021023
Шаг нашивки	0.023616	1.000000	0.003487	0.036437	-0.061015	-0.009875	0.014887	0.003022	0.025795	0.038332	-0.029468	-0.059547	0.013394	-0.004946
Плотность нашивки	0.107947	0.003487	1.000000	-0.004652	0.080304	0.056346	0.017248	-0.039073	0.011391	-0.049923	0.006476	0.019604	0.012239	0.047581
Соотношение матрица-наполнитель	-0.031073	0.036437	-0.004652	1.000000	0.003841	0.031700	-0.006445	0.019766	-0.004776	-0.006272	-0.008411	0.024148	0.072531	0.031866
Плотность, кг/м3	-0.068474	-0.061015	0.080304	0.003841	1.000000	-0.009647	-0.035911	-0.008278	-0.020695	0.044930	-0.017602	-0.069981	-0.015937	-0.094616
модуль упругости, ГПа	-0.025417	-0.009875	0.056346	0.031700	-0.009647	1.000000	0.024049	-0.006804	0.031174	-0.005306	0.023267	0.041868	0.001840	0.995693
Количество отвердителя, м.%	0.038570	0.014887	0.017248	-0.006445	-0.035911	0.024049	1.000000	-0.000684	0.095193	0.055198	-0.065929	-0.075375	0.007446	0.024870
Содержание эпоксидных групп,%_2	0.008052	0.003022	-0.039073	0.019766	-0.008278	-0.006804	-0.000684	1.000000	-0.009769	-0.012940	0.056828	-0.023899	0.015165	-0.005123
Температура вспышки, С_2	0.020695	0.025795	0.011391	-0.004776	-0.020695	0.031174	0.095193	-0.009769	1.000000	0.020121	0.028414	-0.031763	0.059954	0.033670
Поверхностная плотность, г/м2	0.052299	0.038332	-0.049923	-0.006272	0.044930	-0.005306	0.055198	-0.012940	0.020121	1.000000	0.036702	-0.003210	0.015692	-0.010099
Модуль упругости при растяжении, ГПа	0.023003	-0.029468	0.006476	-0.008411	-0.017602	0.023267	-0.065929	0.056828	0.028414	0.036702	1.000000	-0.009009	0.050938	0.023776
Прочность при растяжении, МПа	0.023398	-0.059547	0.019604	0.024148	-0.069981	0.041868	-0.075375	-0.023899	-0.031763	-0.003210	-0.009009	1.000000	0.028602	0.046957
Потребление смолы, г/м2	-0.015334	0.013394	0.012239	0.072531	-0.015937	0.001840	0.007446	0.015165	0.059954	0.015692	0.050938	0.028602	1.000000	0.003232
Удельная прочность	-0.021023	-0.004946	0.047581	0.031866	-0.094616	0.995693	0.024870	-0.005123	0.033670	-0.010099	0.023776	0.046957	0.003232	1.000000

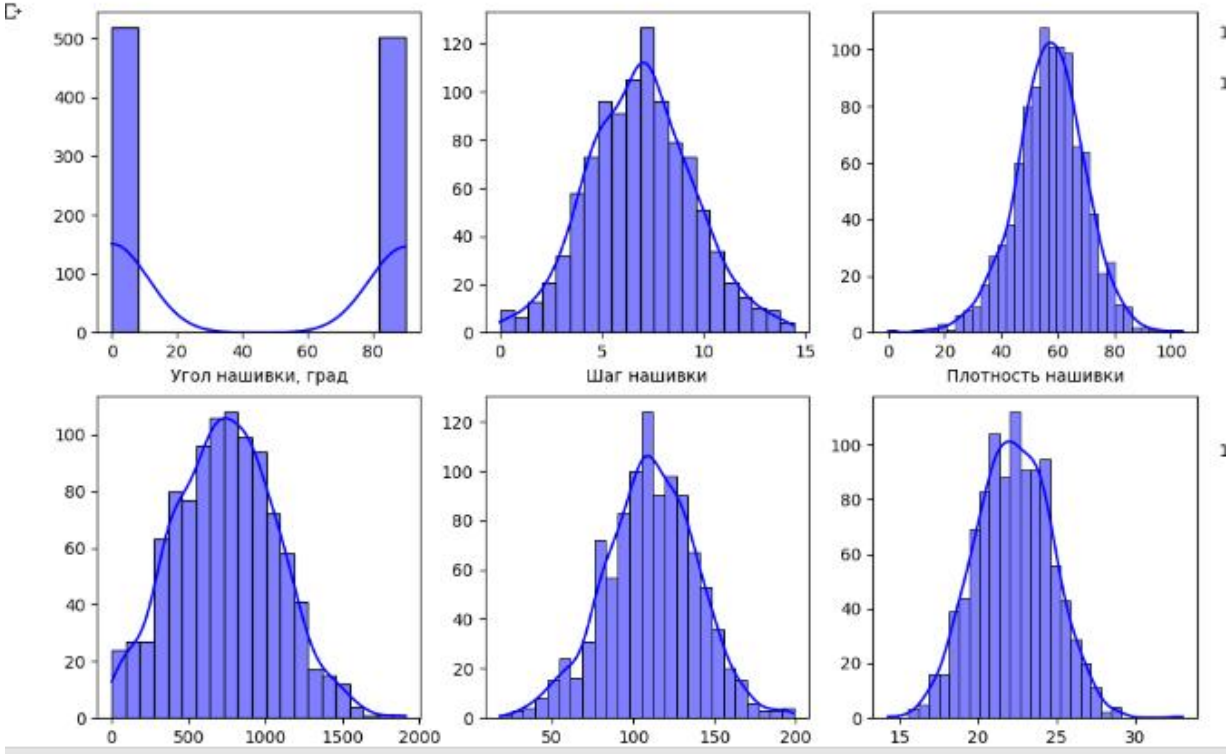
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Анализ первичных данных

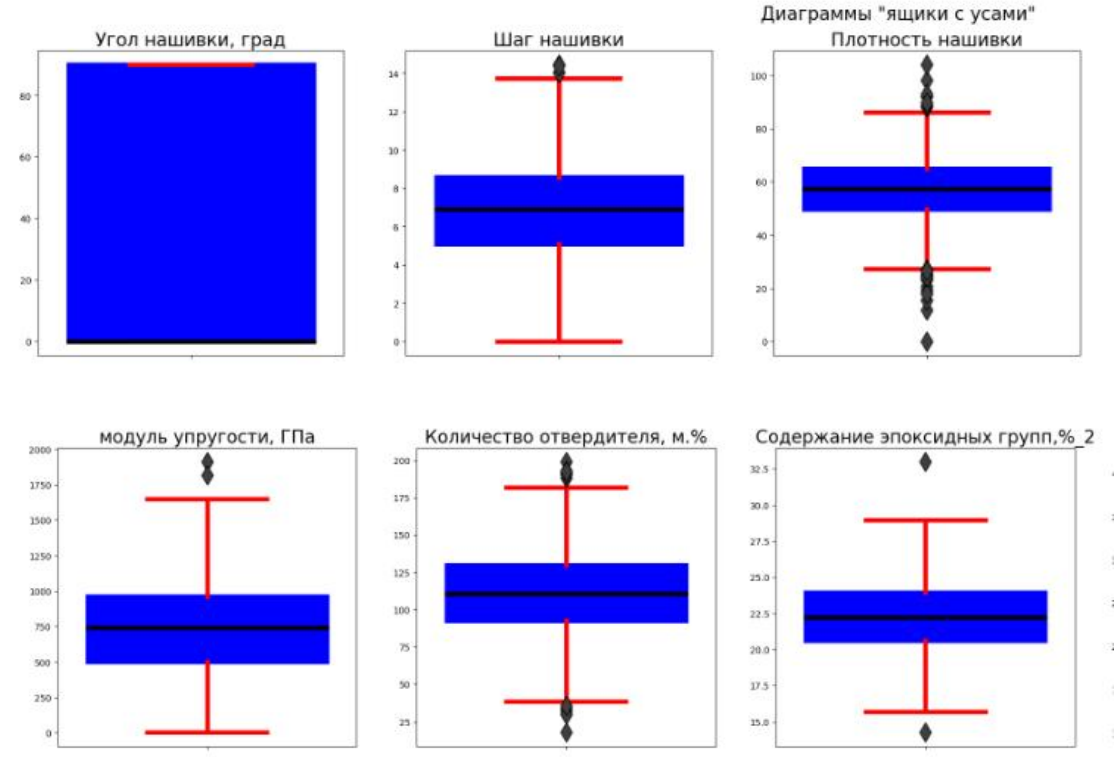
Были построены гистограммы и ящики с усами для вызуализации данных

Код + Текст

```
a=5
b=5
c=1
plt.figure(figsize = (20,20))
for col in df.columns:
 plt.subplot(a, b, c)
 #plt.figure(figsize=(7,5))
 sns.histplot(data = df[col], kde=True, color = "blue")
 plt.ylabel(None)
 c += 1
```




```
plt.figure(figsize = (35,35))
plt.suptitle('Диаграммы "ящики с усами"', y = 0.9 ,
             fontsize = 20)
for col in df.columns:
    plt.subplot(a, b, c)
    sns.boxplot(data = df, y = df[col], fliersize = 15, linewidth = 5, boxprops = dict(facecolor = 'y', color = 'blue'), medianprops = dict(color
    plt.ylabel(None)
    plt.title(col, size = 20)
    c += 1
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

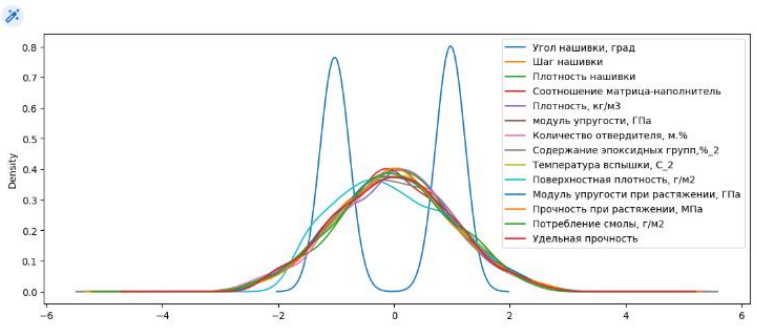
Предобработка данных (чистка массива)

Второй этап обработки данных заключался в:

- поиске выбросов/пропусков их удаления была нормализация данных методом масштабирования
- определение размера датасета после удаления выбросов и проверка наличия пропусков
- вывод информации по массиву
- поиск зависимости между показателями (повторная корреляция, метод явных компонентов для поиска скрытых зависимостей, линейные зависимости, PCA и факторный анализ, который позволят сократить измерения, но у кажого свой + и -)

```
[406] df_new.describe()
```

	Угол нашивки, град	Шаг нашивки	Плотность нашивки	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%
count	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000
mean	46.057692	6.915585	57.451895	2.925683	1974.040023	738.247627	110.916216
std	45.011619	2.509672	11.239331	0.893712	70.808120	328.708665	27.037891
min	0.000000	0.037639	27.272928	0.547391	1784.482245	2.436909	17.740275
25%	0.000000	5.134988	50.209656	2.321931	1923.443748	498.577158	92.443497
50%	90.000000	6.943337	57.584225	2.904731	1977.258043	738.736842	110.923861
75%	90.000000	8.591450	64.798211	3.546650	2020.158764	958.418993	129.082726
max	90.000000	13.732404	86.012427	5.314144	2161.565216	1649.415706	198.953207




```
#корреляция показателей
corr = df_new.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	Угол нашивки, град	Шаг нашивки	Плотность нашивки	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	Удельная прочность
Угол нашивки, град	1.000000	0.024769	0.093057	-0.035897	-0.051336	-0.021886	0.026539	0.026645	0.001721	0.045785	0.034193	0.017669	-0.009916	-0.018880
Шаг нашивки	0.024769	1.000000	0.010024	0.039109	-0.045229	0.011345	-0.021198	0.004465	0.032580	0.033065	-0.007546	-0.061219	0.007569	0.015500
Плотность нашивки	0.093057	0.010024	1.000000	0.052466	0.084077	0.078091	0.001485	-0.036126	-0.006801	-0.048670	0.013445	0.016894	0.001448	0.069478
Соотношение матрица-наполнитель	-0.035897	0.039109	0.052466	1.000000	0.001676	0.043888	0.007812	0.027587	-0.006275	0.007540	-0.019248	0.024488	0.075740	0.044546
Плотность, кг/м3	-0.051336	-0.045229	0.084077	0.001676	1.000000	0.004326	-0.050210	-0.001065	-0.023507	0.062705	-0.021316	-0.076305	-0.005133	-0.077969
модуль упругости, ГПа	-0.021886	0.011345	0.078091	0.043888	0.004326	1.000000	0.036523	-0.013271	0.029318	-0.008987	0.021086	0.036841	0.004450	0.995961
Количество отвердителя, м.%	0.026539	-0.021198	0.001485	0.007812	-0.050210	0.036523	1.000000	0.011330	0.072273	0.046211	-0.048208	-0.063256	-0.002268	0.038578
Содержание эпоксидных групп,%_2	0.026645	0.004465	-0.036126	0.027587	-0.001065	-0.013271	0.011330	1.000000	-0.019085	-0.013065	0.055271	-0.007097	0.011575	-0.012399
Температура вспышки, С_2	0.001721	0.032580	-0.006801	-0.006275	-0.023507	0.029318	0.072273	-0.019085	1.000000	0.018884	0.015361	-0.004877	0.057693	0.031863
Поверхностная плотность, г/м2	0.045785	0.033065	-0.048670	0.007540	0.062705	-0.008987	0.046211	-0.013065	0.018884	1.000000	0.029643	-0.031777	-0.008295	-0.014645
Модуль упругости при растяжении, ГПа	0.034193	-0.007546	0.013445	-0.019248	-0.021316	0.021086	-0.048208	0.055271	0.015361	0.029643	1.000000	-0.001422	0.055023	0.020909
Прочность при растяжении, МПа	0.017669	-0.061219	0.016894	0.024488	-0.076305	0.036841	-0.063256	-0.007097	-0.004877	-0.031777	-0.001422	1.000000	0.027949	0.042755
Потребление смолы, г/м2	-0.009916	0.007569	0.001448	0.075740	-0.005133	0.004450	-0.002268	0.011575	0.057693	-0.008295	0.055023	0.027949	1.000000	0.005242
Удельная прочность	-0.018880	0.015500	0.069478	0.044546	-0.077969	0.995961	0.038578	-0.012399	0.031863	-0.014645	0.020909	0.042755	0.005242	1.000000

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
pca.fit(df_new)
print(pca.explained_variance_ratio_)
```

```
[0.52012319 0.26000416 0.18911772]
```

```
11] print (pca.components_)
```

```
[[ 1.57087301e-03 -3.30436540e-04  4.95971132e-04  4.98865809e-05
  -1.20328320e-02  5.20551624e-02 -3.65060163e-03 -3.77349214e-05
  -3.48411336e-04 -3.04971199e-02 -7.24552803e-06  9.98091381e-01
   3.56605061e-03  2.86272434e-05]
 [-3.35497191e-03  1.16885150e-04  2.66179317e-03  1.14186052e-04
   2.01301418e-03  9.98373549e-01  3.34962989e-03 -9.13887998e-05
   3.57381167e-03 -2.07341864e-02  1.90917603e-04 -5.26633306e-02
   4.62222012e-04  5.04091329e-04]
 [ 7.57027653e-03  2.70786948e-04 -1.82313909e-03  3.13575470e-05
   1.59335650e-02  2.22843994e-02  4.28122238e-03 -1.17480280e-04
   2.77506696e-03  9.99142691e-01  3.25978182e-04  2.95700609e-02
  -1.47130886e-03  7.85519816e-06]]
```

```
print ('Explained variance by component: %s' % pca.explained_variance_ratio_)
```

```
Explained variance by component: [0.52012319 0.26000416 0.18911772]
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Построение и обучение моделей

Задачи:

- Обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении
- При построении модели необходимо 30% данных оставить на тестирование модели, на остальных происходит обучение моделей. При построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.
- Этапы:
- Визуализация данных по двум кластерам: угол нашивки 0’ и угол нашивки 90’
  - нет четкой зависимости при попарном сравнении
  - значения равномерно распределены внутри (нет скоплений, большинство точек лежит в центре, разрежаясь к краям робластей построения, нормальное распределение)
- Разделение выборки на 2 части 30% (тестовая) и 70% (обучающая) для каждой модели.
  - построение трех моделей для каждого показателя ("Модуль упругости при растяжении, ГПа", "Прочность при растяжении, МПа")
  - DummyRegressor - в качестве простого базиса для сравнения с другими (реальными)регрессорами
  - Линейная регрессия
  - Случайный лес

```
print("коэффициент детерминации: X1 ")
print(dummy_regr1.score(X1, y1))
print("коэффициент детерминации: X2 ")
print(dummy_regr2.score(X2, y2))
```

```
коэффициент детерминации: X1
0.0
коэффициент детерминации: X2
0.0
```

```
print(LN_1.score(X1_test, y1_test))
print(LN_2.score(X2_test, y2_test))
```

```
-0.019872015327865533
-0.025589291471932896
```

```
scaler_LN_1 = preprocessing.MinMaxScaler()
df_LN_1_1 = scaler_LN_1.fit_transform(np.array(df_new['Модуль упругости при растяжении, ГПа']).reshape(-1,1))

scaler_LN_2 = preprocessing.MinMaxScaler()
df_LN_2_1 = scaler_LN_2.fit_transform(np.array(df_new['Прочность при растяжении, МПа']).reshape(-1,1))
```

```
# выведем метрики MAE, MSE и R2 в таблицу
df_errors_LrLN_1 = errors(LN_1, X1_train, X1_test, y1_train, y1_test, name = LN_1, trg = LN_1, scaler = scaler_LN_1)
df_errors_LrLN_2 = errors(LN_2, X2_train, X2_test, y2_train, y2_test, name = LN_2, trg = LN_2, scaler = scaler_LN_2)
df_errors_Lr = pd.concat([df_errors_LrLN_1, df_errors_LrLN_2], axis=0).reset_index(drop = True)
df_errors_Lr
```

	Модель	Целевой параметр	MAE(test)	MAE(train)	MSE(test)	MSE(train)	R2 score
0	LinearRegression()	'Модуль упругост	0.840720	0.784548	1.035301	0.968276	-0.019872
1	LinearRegression()	'Прочность при р	0.802764	0.789657	0.987849	0.985416	-0.025589

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

```
[433] #разделение выборки на 2 части 30% (тестовая) и 70% (обучающая)

      #зависимая переменная "Модуль упругости при растяжении, ГПа"
      X1 = df_scaled.drop('Модуль упругости при растяжении, ГПа', axis=1)
      y1 = df_scaled['Модуль упругости при растяжении, ГПа']


      #зависимая переменная "Прочность при растяжении, МПа"
      X2 = df_scaled.drop('Прочность при растяжении, МПа', axis=1)
      y2 = df_scaled['Прочность при растяжении, МПа']
```

```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3, random_state=1)
print(f'Размер обучающей выборки: {X1_train.shape[0]}')
print(f'Размер тестовой выборки: {X1_test.shape[0]}')

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.3, random_state=1)
print(f'Размер обучающей выборки: {X2_train.shape[0]}')
print(f'Размер тестовой выборки: {X2_test.shape[0]}')
```

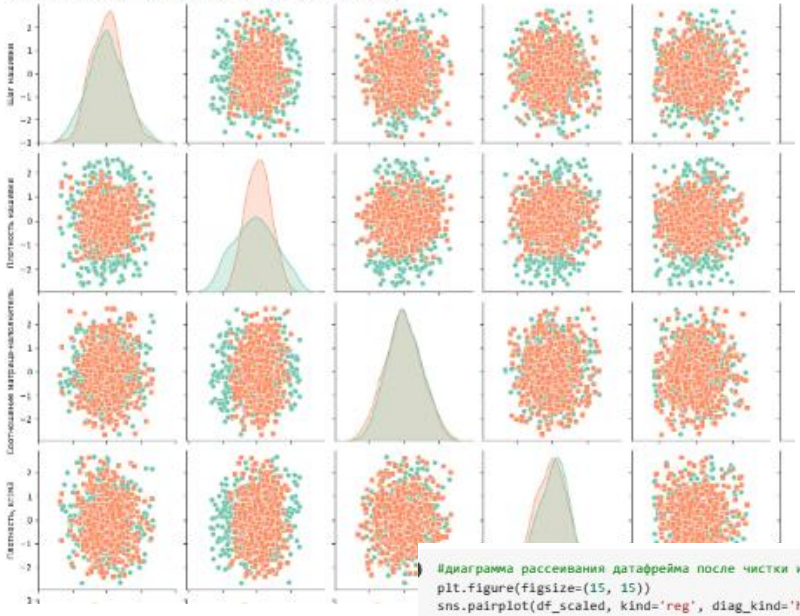
```
Размер обучающей выборки: 655
Размер тестовой выборки: 281
Размер обучающей выборки: 655
Размер тестовой выборки: 281
```

```
[435] X1_test.describe()
```

	Угол нашивки, град	Шаг нашивки	Плотность нашивки	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа
count	281.000000	281.000000	281.000000	281.000000	281.000000	281.000000
mean	0.051243	0.066795	0.005266	0.080108	0.144947	-0.046134
std	0.999259	1.037530	0.977742	1.015767	1.038215	1.003320
min	-1.023787	-2.699225	-2.554424	-2.295284	-2.431802	-2.233896
25%	-1.023787	-0.667829	-0.644702	-0.661707	-0.602821	-0.784865

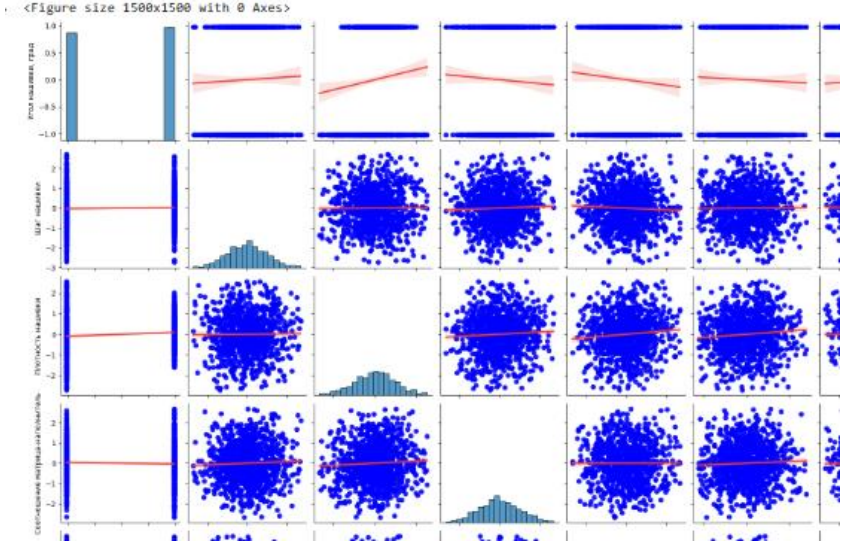
```
[431] sns.pairplot(df_scaled, hue = 'Угол нашивки, град', markers=["o", "s"], diag_kind= 'auto'
```

```
<seaborn.axisgrid.PairGrid at 0x7f8b96306590>
```



```
#диаграмма рассеивания датафрейма после чистки и масштабирования
plt.figure(figsize=(15, 15))
sns.pairplot(df_scaled, kind='reg', diag_kind='hist', plot_kws={'line_kws': {'color': '#eb4034'},
                                                                'scatter_kws': {'alpha': 0.9,
                                                                                'color': 'blue'}})

plt.show()
```

```
<Figure size 1500x1500 with 0 Axes>
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Построение и обучение моделей

В результате сравнения расчетных (предсказанных) и актуальных значений мы видим, что линейная модель не работает.

```
[455] #создадим модель, которая будет выбирать новую тестовую выборку в заданном количестве и запишем все полученны
      def sample(count, model, X, y, test_size=0.3, scaler = 'scaler'):
          sample = []
          for _ in range(count):
              X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
              model.fit(X_train, y_train)
              sample.append({'MAE':mean_absolute_error(y_test, model.predict(X_test)),'MSE':mean_squared_error(y_t
          return pd.DataFrame(sample)

[456] # зададим количество новых выборок 10
      df_sample_lrLN_1 = sample(10, LN_1, X1, y1, scaler = scaler_LN_1)
      df_sample_lrLN_2 = sample(10, LN_2, X2, y2, scaler = scaler_LN_2)

[457] # выведим минимальные значения метрик для целевого параметра'Модуль упругости при растяжении, ГПа'
      df_sample_lrLN_1.min()

      MAE    0.748661
      MSE    0.835805
      dtype: float64

[458] # выведим минимальные значения метрик для целевого параметра 'Прочность при растяжении, МПа'
      df_sample_lrLN_2.min()

      MAE    0.744184
      MSE    0.890899
      dtype: float64
```

	Модуль упругости при растяжении, ГПа |		Прочность при растяжении, МПа |	
	Актуальные y1	Предсказанные y1	Актуальные y2	Предсказанные y2
386	-1.384355	0.101702	-1.880556	0.023939
41	0.873654	0.017623	-1.785143	0.004709
725	-0.528017	0.045892	0.466423	0.063262
605	0.504928	0.213588	-0.884391	0.493296
35	-0.601519	-0.054442	0.588322	-0.155141
...	...	...	...	...
543	1.075745	-0.002902	0.252099	-0.144425
433	0.249434	-0.378322	0.017595	-0.305889
57	0.326632	-0.199294	0.847260	-0.001536
355	-0.288448	-0.114346	0.813576	-0.028703
584	0.906742	-0.245915	0.229710	-0.057714

281 rows × 4 columns

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Построение и обучение моделей

Модель Случайный лес так же показала, что предсказанные значения и актуальных различаются, модель не работает.

```
[459] ##Случайный лес

[460] #разделение выборки на 2 части 30% (тестовая) и 70% (обучающая)

      #зависимая переменная "Модуль упругости при растяжении, ГПа"
      X3 = df_scaled.drop('Модуль упругости при растяжении, ГПа', axis=1)
      y3 = df_scaled['Модуль упругости при растяжении, ГПа']

[461] X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.3, random_state=
      print(f'Размер обучающей выборки: {X3_train.shape[0]}')
      print(f'Размер тестовой выборки: {X3_test.shape[0]}')

      Размер обучающей выборки: 655
      Размер тестовой выборки: 281

[462] #зависимая переменная "Прочность при растяжении, МПа"
      X4 = df_scaled.drop('Прочность при растяжении, МПа', axis=1)
      y4 = df_scaled['Прочность при растяжении, МПа']

[463] X4_train, X4_test, y4_train, y4_test = train_test_split(X4, y4, test_size=0.3, random_state=
      print(f'Размер обучающей выборки: {X4_train.shape[0]}')
      print(f'Размер тестовой выборки: {X4_test.shape[0]}')

      Размер обучающей выборки: 655
      Размер тестовой выборки: 281

[464] from sklearn.ensemble import RandomForestRegressor

[465] # определяем X и y, строим модель по нормализованному DataFrame для каждого целевого парамет

      SL_3=RandomForestRegressor()
      SL_3.fit(X3_train, y3_train)
      y3_pred = SL_3.predict(X3_test)

[466] SL_4=RandomForestRegressor()
      SL_4.fit(X4_train, y4_train)
      y4_pred = SL_4.predict(X4_test)
```

```
      print(SL_4.score(X4_test, y4_test))

      -0.10776305408054165
      -0.05326473765055928

[468] np.mean(np.abs(y3_test-SL_3.predict(X3_test)))

      0.862138409568424

[469] np.mean(np.abs(y3_test-np.mean(y3_test)))

      0.8289382478364875

[470] np.mean(np.abs(y4_test-SL_4.predict(X4_test)))

      0.8157869149594243

[471] np.mean(np.abs(y4_test-np.mean(y4_test)))

      0.7880988878953843

[472] SL_3_mae = mean_absolute_error(y3_test, y3_pred)
      SL_3_rmse = mean_squared_error(y3_test, y3_pred, squared=False)
      print("---RANDOM FOREST REGRESSOR Модуль упругости при растяжении, ГПа")
      print("R2 score: "+str(SL_3))
      print("MAE: "+str(SL_3_mae))
      print("MSE: "+str(SL_3_rmse))

      ---RANDOM FOREST REGRESSOR Модуль упругости при растяжении, ГПа
      R2 score: RandomForestRegressor()
      MAE: 0.862138409568424
      MSE: 1.060434913869693
```

```
[473] SL_4_mae = mean_absolute_error(y3_test, y3_pred)
      SL_4_rmse = mean_squared_error(y3_test, y3_pred, squared=False)
      print("---RANDOM FOREST REGRESSOR Прочность при растяжении, МПа")
      print("R2 score: "+str(SL_4))
      print("MAE: "+str(SL_4_mae))
      print("MSE: "+str(SL_4_rmse))

      ---RANDOM FOREST REGRESSOR Прочность при растяжении, МПа
      R2 score: RandomForestRegressor()
```

```
print("Модуль упругости при растяжении, ГПа", "|", "Прочность при растяжении,
f = pd.DataFrame({'Актуальные y3': y3_test, 'Предсказанные y3': y3_pred, 'Акт
f
```

Модуль упругости при растяжении, ГПа | Прочность при растяжении, МПа |

	Актуальные y3	Предсказанные y3	Актуальные y4	Предсказанные y4
386	-1.384355	-0.109541	-1.880556	0.176896
41	0.873654	0.410010	-1.785143	-0.136159
725	-0.528017	0.232448	0.466423	-0.001878
605	0.504928	-0.095384	-0.884391	0.027301
35	-0.601519	0.126694	0.588322	0.011220
...	...	...	...	...
543	1.075745	0.051227	0.252099	-0.203719
433	0.249434	-0.120367	0.017595	0.217596
57	0.326632	-0.720803	0.847260	-0.018422
355	-0.288448	-0.160203	0.813576	0.501315
584	0.906742	-0.398649	0.229710	-0.074886

281 rows × 4 columns

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Создание нейронной сети. Соотношение матрица-наполнитель

Нейронная сеть создавалась на базе ранее вычищенного, НО! не нормированного массива, на основе которого строились модели.

- импортировались/загружались требуемые библиотеки
- создание обучающей и контрольной выборок
- компиляция модели, ее применение на данных, оценка модели

```
%%time
model.summary()
dfmodel = model.compile(optimizer='adam', loss='mae', metrics=['mae'])
history = model.fit(
    X5_train,
    y5_train,
    validation_split=0.2,
    verbose=1, epochs=60)
```

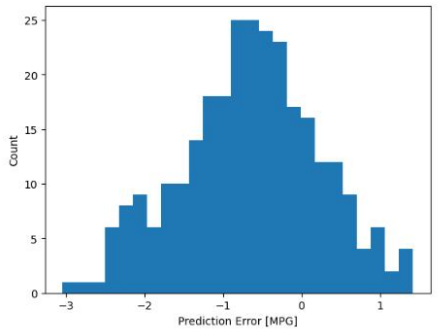
```
Model: "sequential_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_109 (Dense)           (None, 50)                700

 dropout_48 (Dropout)        (None, 50)                0

 dense_110 (Dense)           (None, 128)               6528

 dropout_49 (Dropout)        (None, 128)               0
```

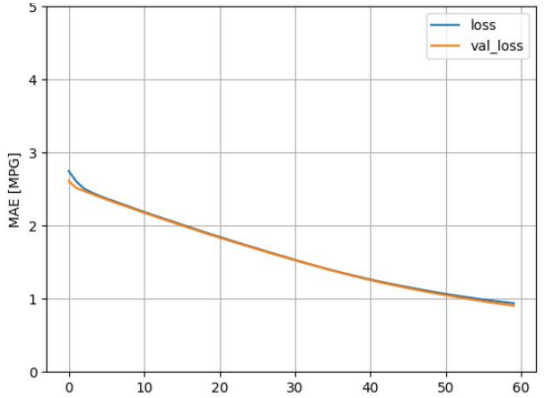
```
[484] test_predictions

array([2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554,
       2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554,
       2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554,
       2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554,
       2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554, 2.2501554,
```

```
[485] model.evaluate(X5_test, y5_test, verbose=1)

9/9 [==============================] - 0s 2ms/step - loss: 0.8767 - mae: 0.8767
[0.8766633868217468, 0.8766633868217468]
```

```
[485] model.evaluate(X5_test, y5_test, verbose=1)

9/9 [==============================] - 0s 2ms/step - loss: 0.8767 - mae: 0.8767
[0.8766633868217468, 0.8766633868217468]
```

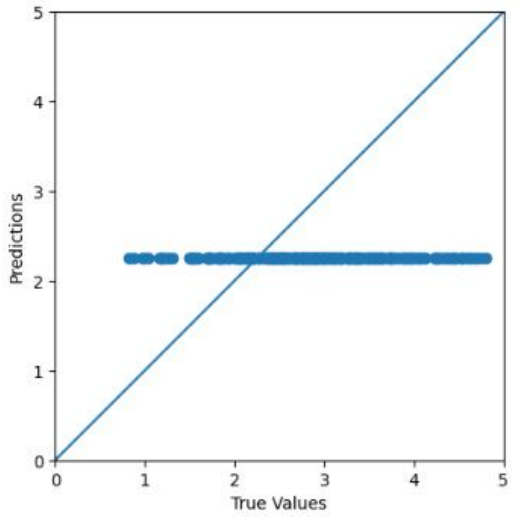
```
error = test_predictions - y5_test
plt.hist(error, bins=25)
plt.xlabel('Prediction Error [MPG]')
_ = plt.ylabel('Count')
```






```
test_predictions = model.predict(X5_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y5_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 5]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
9/9 [==============================] - 0s 2ms/step
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Создание нейронной сети. Соотношение матрица-наполнитель

Нейронная сеть создавалась на базе ранее вычищенного и нормированного массива!, на основе которого строились модели.

- импортировались/загружались требуемые библиотеки
- создание обучающей и контрольной выборок
- компиляция модели, ее применение на данных, оценка модели

Во второй модели разброс прогнозируемых и актуальных значений более разнообразный, но и как в первом случае данные расположились посередине в горизонтальной плоскости.

```
[488] ## Модель 2
```

```
[489] X6 = df_new.drop(['Соотношение матрица-наполнитель'], axis = 1)
      # Целевая переменная:
      y6 = df_new['Соотношение матрица-наполнитель']
      from sklearn.model_selection import train_test_split
      X6_train, X6_test, y6_train, y6_test = train_test_split(X6, y6, test_size=0.3, random_state=42)
```

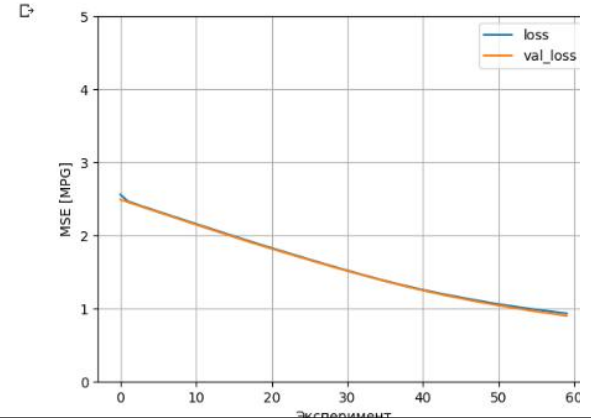
```
goal = np.array(X6_train)
goal_normalizer = layers.Normalization(input_shape=[13,], axis=None)
goal_normalizer.adapt(goal)
goal
```

```
array([[0.00000000e+00, 4.57089231e+00, 7.8547358
        3.46043821e+03, 3.40016626e+02, 3.3790378
       [9.00000000e+01, 1.17044041e+01, 5.3744192
        2.15082883e+03, 1.26966998e+02, 3.1479242
       [9.00000000e+01, 3.98006783e+00, 4.3107489
```

```
dnn_goal_model = build_and_compile_model(goal_normalizer)
dnn_goal_model.summary()
```

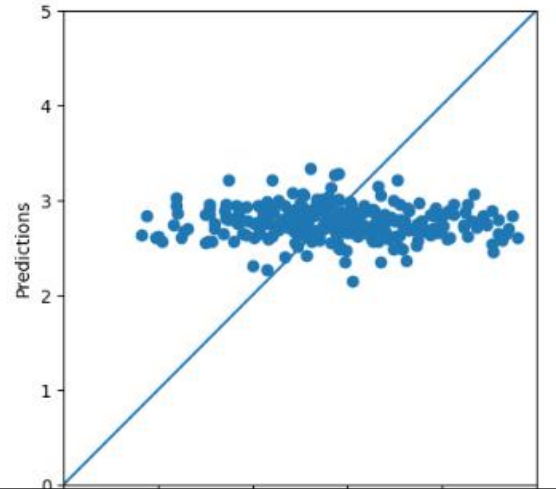
```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_7 (Normalizat  (None, 13)               3
 ion)

 dense_115 (Dense)           (None, 256)               3584

 dense_116 (Dense)           (None, 256)               65792

 dense_117 (Dense)           (None, 64)                16448

 dense_118 (Dense)           (None, 1)                 65

=================================================================
Total params: 85,892
Trainable params: 85,889
Non-trainable params: 3
_________________________________________________________________
```

```
plot_loss(history2)
```




```
test_predictions = dnn_goal_model.predict(X6_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y6_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 5]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
9/9 [==============================] - 0s 5ms/step
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

ВЫВОДЫ

Результаты анализа данных.

- данные не взаимосвязаны (корреляция очень слабая)
- введение доп параметра "удельная прочность" не позволил выявить зависимости между переменными

Разработанные модели дают результаты хуже, чем если просто брать среднее значение прогнозируемой величины.

Если данные не нормализовать, то прогнозные значения все равны при постронии модели "Соотношение матрица-наполнитель". Но как в первом, так и во втором случае прогнозные значения лежат в пределах 2-3,5 в горизонтальной плоскости.

Таким образом можно сказать, что полученные результаты не позволяют спрогнозировать/предсказать требуемые параметры. Возможные пути решения - разделить массив данных на несколько схожих групп и провести прогнозирование на них. Например провести кластеризацию по искуственному параметру "удельная прочность".

P.S.: также остается вопрос к качеству данных. Хоть они и были вычещены на предмет выбросов, сами данные имеют слишком большую вариативность. Возможно, данные были собраны при проведении экспериментов в разных условиях (без контроля условий / критериев сбора данных) либо для большого количества материалов с разными физическими свойствами.