



# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

## по курсу

### «Data Science»

Тема: Прогнозирование конечных свойств новых  
материалов  
(композиционных материалов)

Слушатель: Григорьев Сергей Алексеевич

# Постановка задачи

Постановка задач — это процесс составления точной формулировки задания с описанием условий для его выполнения.

Правильно поставленная задача помогает минимизировать количество вопросов, упростить и ускорить работу

## Постановка задачи

**Цель исследования** состоит в прогнозировании ряда конечных свойств получаемых композиционных материалов на основе пула входящих параметров.

### Задачи исследования

- Изучить теоретические основы и методы решения поставленной задачи;
- Провести разведочный анализ данных;
- Провести предобработку данных;
- Обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении;
- Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель;
- Разработать приложение;
- Оценить точность модели на тренировочном и тестовом датасете;
- Создать удаленный репозиторий и разместить там код исследования. Оформить файл readme.

P.s. Шаблон Презентации взят у слушателя О. Евдокимова



# Основные этапы работы

- Перед началом работы и в процессе ее выполнения для решения подзадач были изучены теоретические основы, методы решения и практические составляющие поставленной задачи.
- Вся работа разбита на 5 основных Блоков;
- При решении задачи часть часто повторяющихся модулей вынесена в виде функций в папку модулей и импортируется в каждый Блок по мере необходимости.
- Все графики и получаемые в конце Блока датасеты заносятся в директории соответствующие номеру Блока.
- Была проведены 3 поколения исследования. Первые два поколения приведены в директории Diploma\_naive\_old.
- Использованы 6 разных методов регрессий для каждого целевого параметра.
- Применен нестандартный метод решения выборки для регрессионного анализа.
- Создана работающая модель предсказания одного из параметров.



Рабочий Проект разделен на 5 частей:

Diploma\_block1\_clean.ipynb:

Проведено объединение двух входных df в один по методу INNER.

Проведена замена названий параметров с русского языка на английский.

Проведен разведочный анализ предложенных данных.

Продемонстрированы гистограммы распределения каждой из переменной, диаграммы ящика с усами, матричные графики рассеяния точек.

Для каждого параметра получены среднее, медианное значение, проверено наличие пропусков, проведен итерационный анализ и исключение выбросов несколькими методами.

Осуществлена замена "подозрительных значений" на средние значения ближайших 5 соседей.

По результатам работы в Блоке № 1 для дальнейшего исследования создан очищенный от выбросов df\_main\_clean.csv, который записан в папку data\_block1\_clean, которая входит в директорию верхнего уровня data\_storage.

Все полученные графики сохранены в директории save\_fig/fig\_block1\_clean.

Diploma\_block2\_corr.ipynb:

Проведено исследование корреляционных матриц очищенного от выбросов и подозрительных значений df в зависимости от количества значений в выборке.

Сделан вывод, что при рассмотрении полного df корреляция отсутствует практически полностью, однако при значениях N = 100 и 200 еще наблюдается некоторая корреляционная зависимость между параметрами, с увеличением числа выборки зависимость падает до около нулевых значений.

Проведен графический анализ использованием регрессионной прямой по каждому параметру с целевыми переменными "Модуль упругости при растяжении" и "Прочности при растяжении".

На основании теоретических исследований, приведенных в научной литературе по данной теме введен новый параметр ALFA как функция Модуля упругости при растяжении и Прочности при растяжении.

Произведен выбор фичей для наших целевых параметров, что позволит сократить вычислительные ресурсы и увеличить точность расчетов.

Подготовлен df\_data\_Train\_ML.csv из 200 случайных значений (random\_state=42) в качестве обучающего массива для регрессионного анализа в оке 3 и df\_data\_Test\_ML.csv для использования в качестве тестового набора данных (random\_state=50). Полученные df записаны в соответствующую директорию data\_storage/data\_block2\_corr.

Полученные в процессе работы Блока 2 графики записаны в соответствующую директорию save\_fig/fig\_block2\_corr

Diploma\_block3\_regress.ipynb:

Проведена предобработка данных, нормализация и стандартизация.

Обучены 6 типов моделей для прогноза модуля упругости при растяжении и прочности при растяжении.

Получены и сведены в 2 df\_elgor\_calc\_EMS.csv и df\_elgor\_calc\_SS.csv значения по 7 оценочным параметрам работы разных моделей регрессии и записаны в директорию data\_storage/data\_block3\_regress/

Сделан вывод о практическом отсутствии регрессии между параметрами и целевой переменной "Модуль упругости при растяжении" и невозможно прогнозировать этот целевой параметр по имеющемуся набору данных.

В тоже время по целевому показателю 'Прочность при растяжении' считаю вполне возможно на небольшом датасете проводить прогнозирование результата с довольно большой точностью!

Полученные в процессе работы Блока 3 графики записаны в соответствующую директорию save\_fig/fig\_block3\_regress.

Diploma\_block4\_neuro.ipynb:

Для работы с нейросетью был применен полный очищенный от выбросов df\_main\_clean.csv, который получен в Блоке №1.

Проведен обоснованный расчет количества слоев нейронной сети, количества нейронов в каждом слое и выбор активационных функций для каждого слоя.

Написана нейронная сеть, которая будет рекомендовать соотношение целевого параметра "Матрица-наполнитель".

Оценены точности моделей на тренировочном и тестовом датасете.

Полученная и обученная модель нейронной сети была сохранена в директории data\_storage/data\_block5\_neuro/model\_RFNN.pkl.

Полученные в процессе работы Блока 4 графики записаны в соответствующую директорию save\_fig/fig\_block4\_neuro.

Diploma\_block5\_flask (папка):

Блок создания приложения в модуле FLASK по прогнозированию целевого параметра Соотношение матрица-наполнитель.

Создано приложение app.py и запущено в работу. Результаты отработанного приложения приведены в конце файла. В процессе работы приложения Пользователь заносит рабочие значения параметров, а Приложение сначала нормализует данные, затем передает их в сохраненную модель, а затем инвертирует расчетные значения в обычную размерность параметра "Соотношение матрица-наполнитель".

# БЛОК № 1

## ✓ Объединение по индексу INNER:

- Импортируем необходимые библиотеки и модули-функции;
- Загружаем файлы;
- Посмотрим размерность;
- Убираем детерминированные синтетические данные, обнуляем индексы
- Объединим оба файла по индексу по типу объединения INNER
- Заменим названия столбцов на английский язык

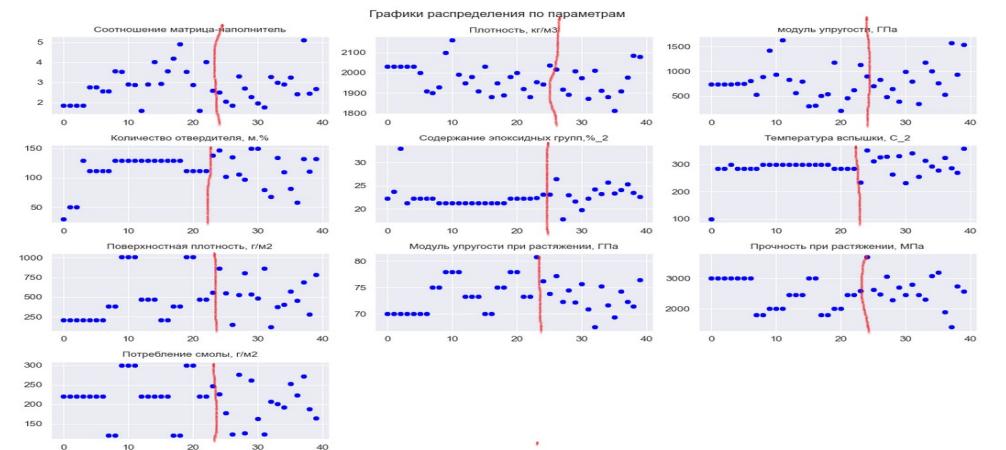


### Датасет X\_bp.xlsx:

Тип данных: float64

Количество параметров: 10

Количество записей: 1023

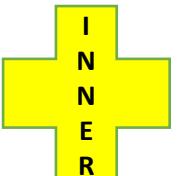


### Датасет X\_bp.xlsx:

Тип данных: float64

Количество параметров: 10

Количество записей: 1000

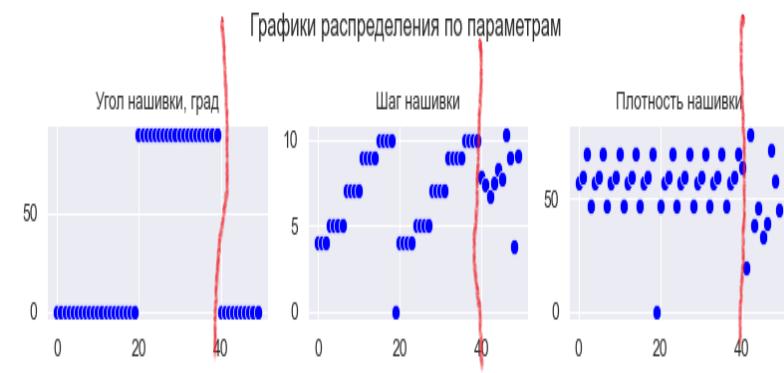


### Датасет X\_nup.xlsx:

Тип данных: float64

Количество параметров: 3

Количество записей: 1040



### Датасет X\_nup.xlsx:

Тип данных: float64

Количество параметров: 3

Количество записей: 1000

### Датафрейм ‘data\_main’:

Тип данных: float64

Количество параметров: 13

Количество записей: 1000

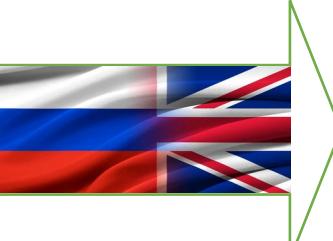


### data\_main.info()

```

<class 'pandas.core.frame.DataFrame'\>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count   Dtype  
 ---  --  
 0   Угол нашивки, град   1000 non-null    int64  
 1   Шаг нашивки          1000 non-null    float64 
 2   Плотность нашивки     1000 non-null    float64 
 3   Соотношение матрица-наполнитель 1000 non-null    float64 
 4   Плотность, кг/м3        1000 non-null    float64 
 5   модуль упругости, ГПа      1000 non-null    float64 
 6   Количество отвердителя, м.%   1000 non-null    float64 
 7   Содержание эпоксидных групп,%_2 1000 non-null    float64 
 8   Температура вспышки, C_2       1000 non-null    float64 
 9   Поверхностная плотность, г/м2     1000 non-null    float64 
 10  Модуль упругости при растяжении, ГПа 1000 non-null    float64 
 11  Прочность при растяжении, МПа      1000 non-null    float64 
 12  Потребление смолы, г/м2        1000 non-null    float64 
dtypes: float64(12), int64(1)
memory usage: 101.7 KB

```



### data\_main.info()

```

<class 'pandas.core.frame.DataFrame'\>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count   Dtype  
 ---  --  
 0   pattern_angle    1000 non-null    int64  
 1   step_strip         1000 non-null    float64 
 2   density_strip       1000 non-null    float64 
 3   ratio_filler_matrix 1000 non-null    float64 
 4   density            1000 non-null    float64 
 5   elasticity_module  1000 non-null    float64 
 6   number_hardeners  1000 non-null    float64 
 7   content_epoxy_groups 1000 non-null    float64 
 8   flash_temperature    1000 non-null    float64 
 9   surface_density      1000 non-null    float64 
 10  elasticity_module_stretching 1000 non-null    float64 
 11  strapery_strength    1000 non-null    float64 
 12  resin_consumption    1000 non-null    float64 
dtypes: float64(12), int64(1)
memory usage: 101.7 KB

```



## БЛОК № 1

### Описательная статистика

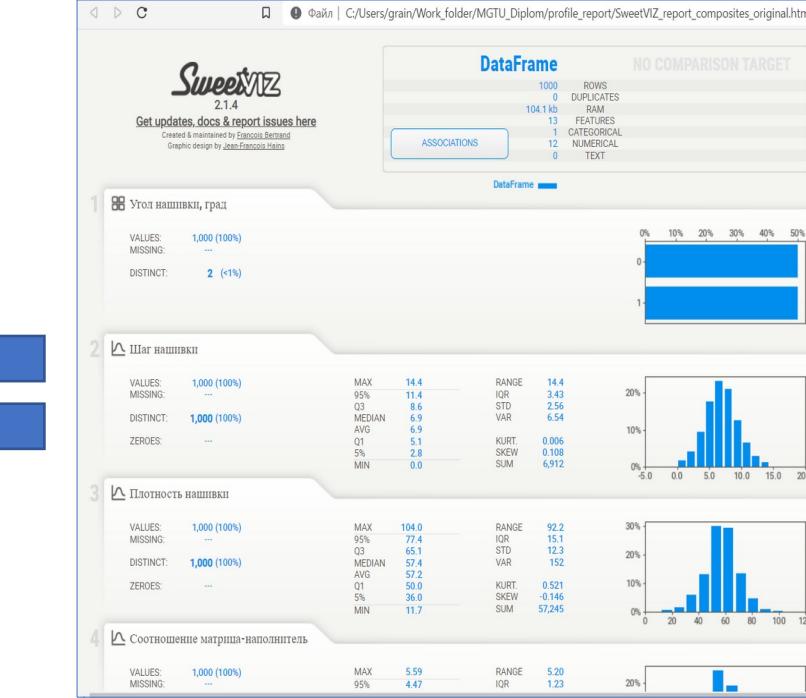
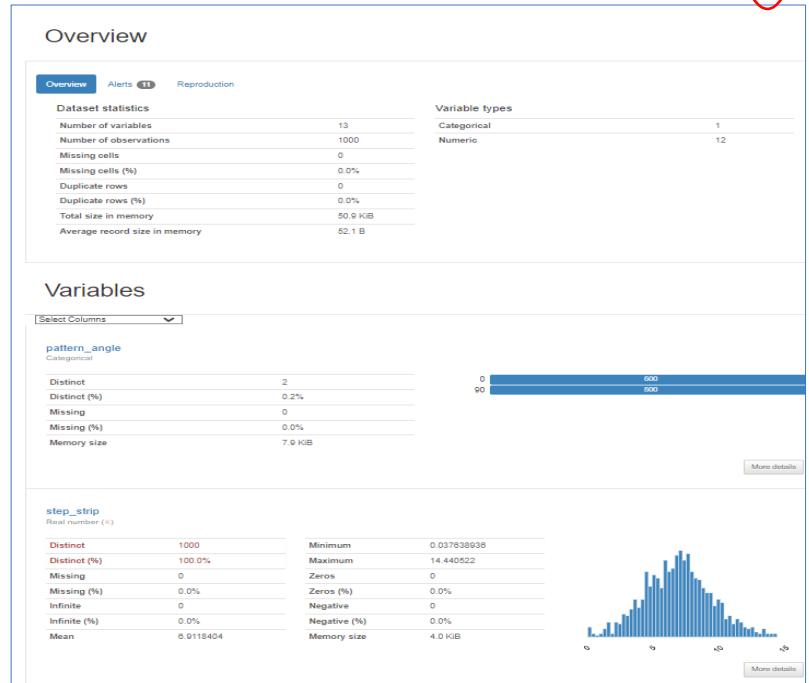
- Посмотрим описательную таблицу статистических данных датасета;
- Изучим информацию о датасете;
- Медианные и средние значения очень близки, что говорит о видимом отсутствии корреляции;
- Получим полные статистические отчеты по двум библиотекам, которыми можно было заменить весь наш разведочный анализ.



ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

# Выведем описательную таблицу в транспонированном виде (.T) и с округлением до df\_le.describe().round(2).T

	count	mean	std	min	25%	50%	75%	max
<b>pattern_angle</b>	1000.0	0.50	0.50	0.00	0.00	0.50	1.00	1.00
<b>step_strip</b>	1000.0	6.91	2.56	0.04	5.14	6.91	8.57	14.44
<b>density_strip</b>	1000.0	57.25	12.34	11.74	49.97	57.41	65.11	103.99
<b>ratio_filler_matrix</b>	1000.0	2.93	0.91	0.39	2.32	2.91	3.55	5.59
<b>density</b>	1000.0	1975.67	73.80	1731.76	1924.20	1977.57	2021.16	2207.77
<b>elasticity_module</b>	1000.0	739.95	330.33	2.44	498.44	741.15	962.85	1911.54
<b>number_hardeners</b>	1000.0	110.54	28.30	17.74	92.17	110.16	130.31	198.95
<b>content_epoxy_groups</b>	1000.0	22.24	2.41	14.25	20.56	22.23	23.98	28.96
<b>flash_temperature</b>	1000.0	285.91	40.96	160.26	258.54	285.85	313.58	413.27
<b>surface_density</b>	1000.0	483.02	280.81	0.60	268.06	452.97	694.21	1399.54
<b>elasticity_module_stretching</b>	1000.0	73.33	3.12	64.05	71.30	73.25	75.38	82.68
<b>strapery_strength</b>	1000.0	2467.18	485.62	1036.86	2143.83	2461.25	2760.16	3848.44
<b>resin_consumption</b>	1000.0	218.39	59.82	33.80	179.19	217.28	257.50	414.59



## БЛОК № 1

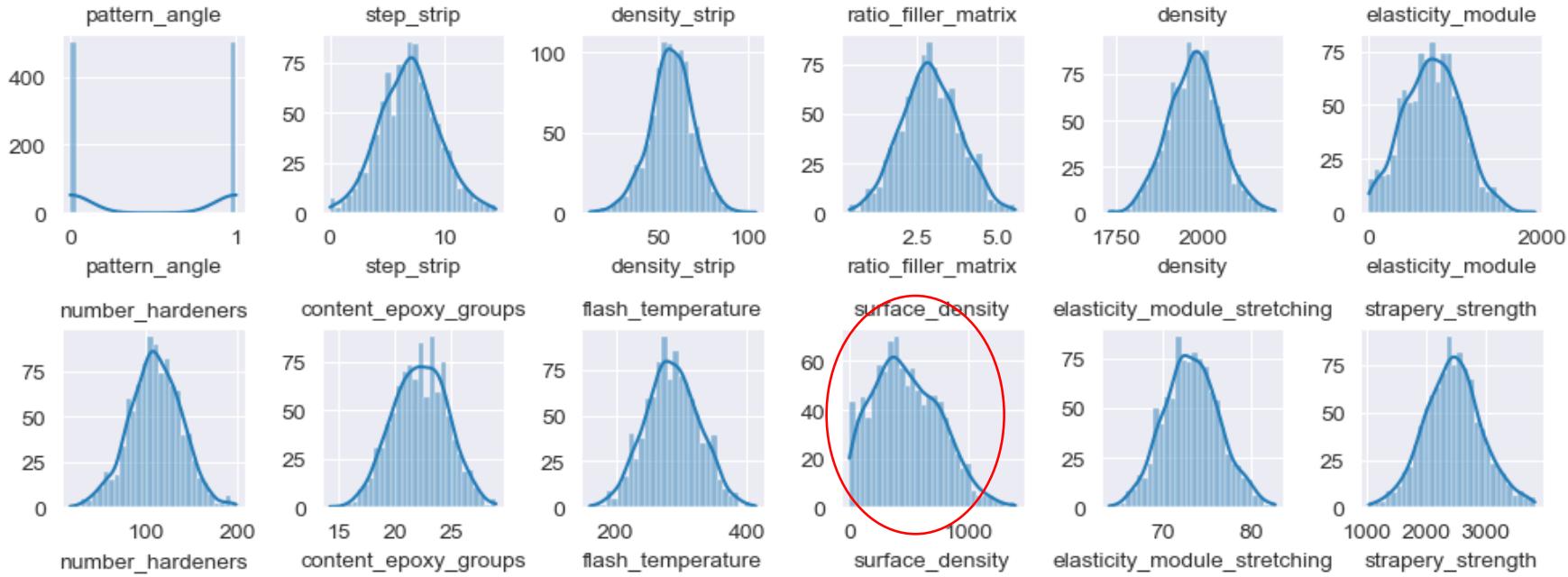
### Графическая часть статистического анализа:

- Гистограммы параметров;
- Изучим “странную” гистограмму параметра ‘surface\_density’ и его SKEW;
- Подвергнем логарифмированию этот параметр;
- Построим новую гистограмму распределения и точечное распределение;
- Мы ушли от значений близких к 0

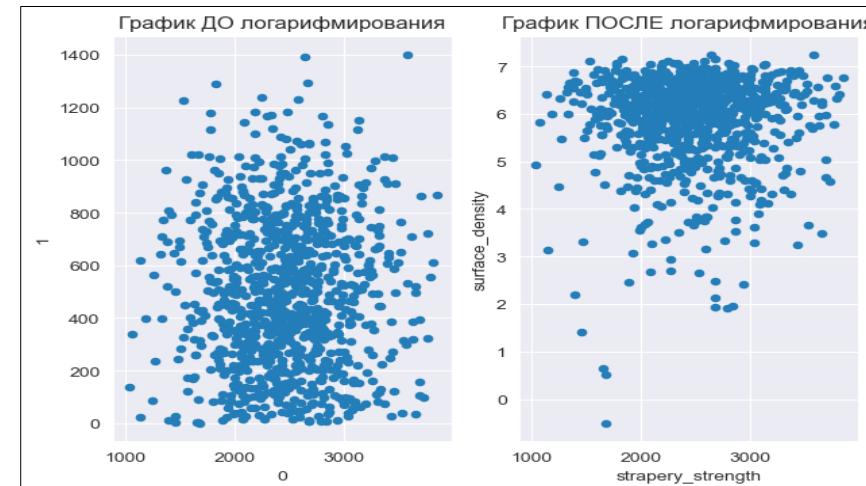
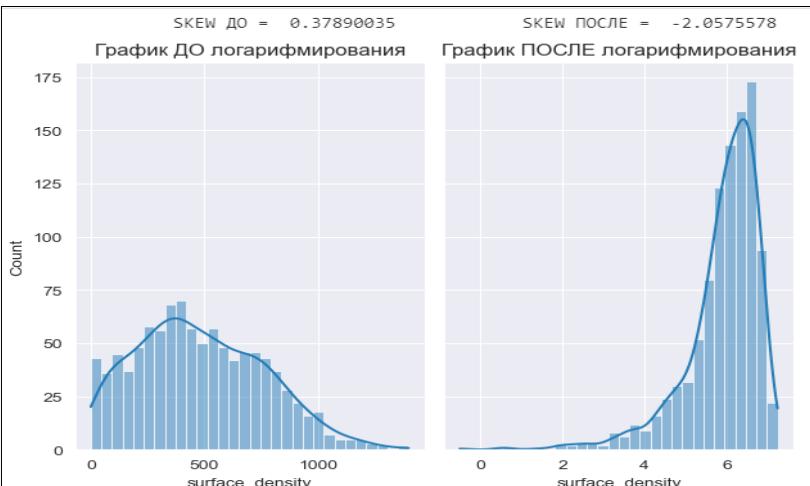


ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

Графики гистограмм по параметрам



### Блок логарифмирования параметра surface\_density



## БЛОК № 1

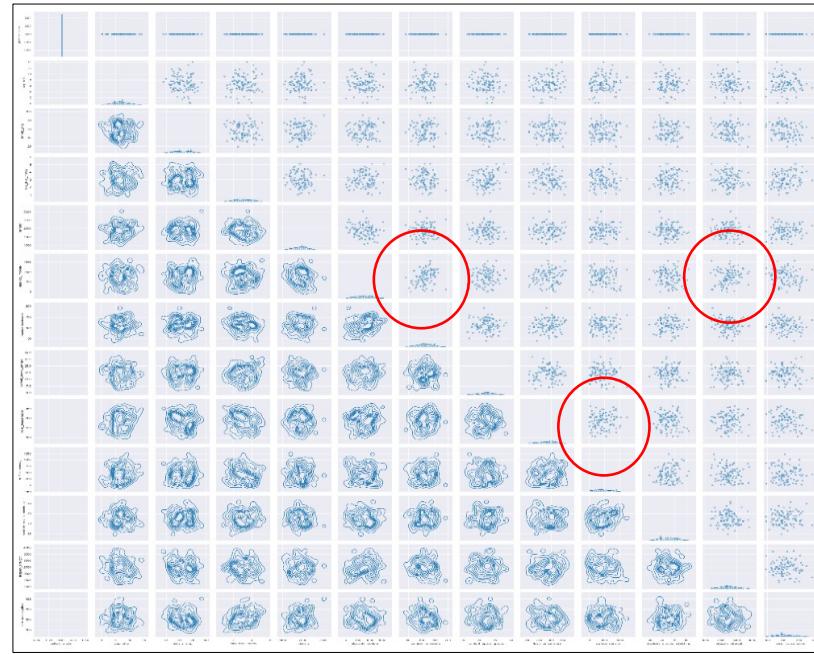
### Графическая часть статистического анализа:

- Матричные графики параметров друг от друга для полного датасета и для 500 строк;
- Матричные графики параметров в зависимости от Угла нашивки;
- На полных графиках корреляция отсутствует полностью;
- Необходимо изучить параметры корреляции на более разреженных данных;
- Создан датасет ЭКСПЕРТНЫХ краевых значений на основе гистограмм и функции расчета интервалов;

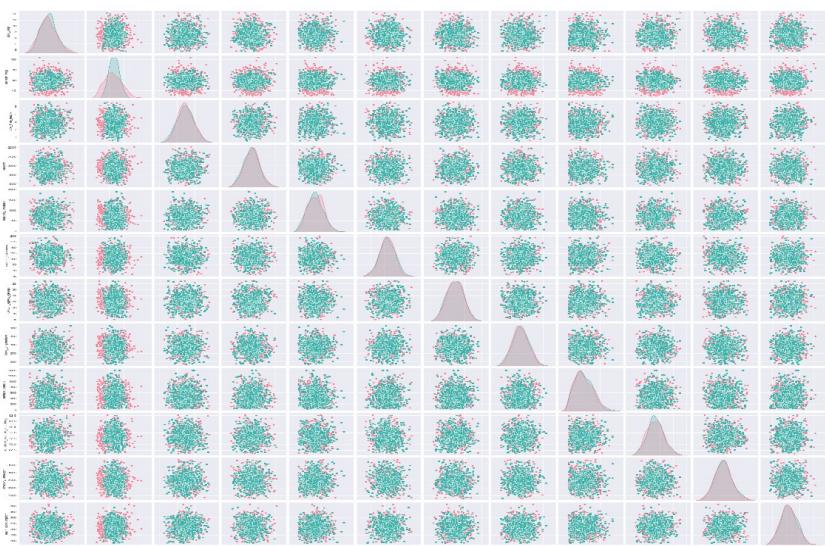


ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

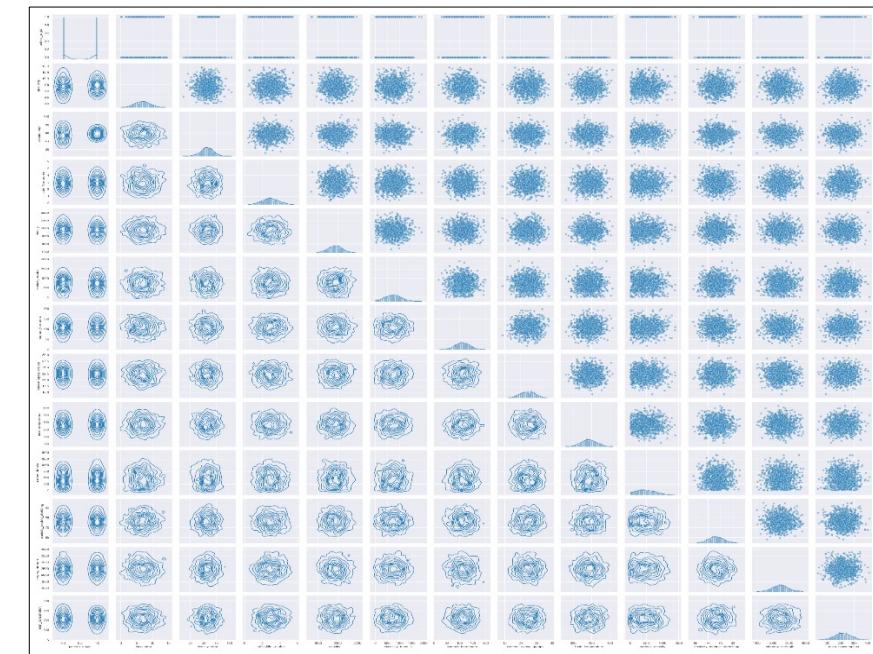
Парные графики при N = 500 строк



Парные графики для разных Углов нашивки



Парные графики Full



Экспертные МИН и МАХ границы параметров

```
df_min_max = pd.DataFrame({  
    'pattern_angle': [0, 1],  
    'step_strip': [1.5, 13.5],  
    'density_strip' : [29.0, 85.6],  
    'ratio_filler_matrix':[0.5, 6.0],  
    'density' : [1780.0, 2159.0],  
    'elasticity_module' : [50.0, 1560.0],  
    'number_hardeners':[30.0, 190.0],  
    'content_epoxy_groups':[16.0, 28.0],  
    'flash_temperature':[180.0, 394.0],  
    'surface_density' : [28.5, 1200.0],  
    'elasticity_module_stretching':[66.0, 81.0],  
    'strapery_strength' : [1326.0, 3600.0],  
    'resin_consumption':[80.0, 350.0],  
})
```

## БЛОК № 1

### Этап очистки датасета от выбросов:

- Создана функция ЗАМЕНЫ ‘подозрительно малых значений’ СЛЕВА по оценкам Экспертным МИН и МАХ на основе ближайших 5 значений вместо замены на медиану;
- Проведена очистка по методу IQR (3 сигм) и Z\_score;
- Создана функция ЗАМЕНЫ ‘подозрительно малых значений’ СПРАВА;
- Провели итерацию 4 раза с уточнением МИН и МАХ.
- Получили ‘чистый’ датасет.

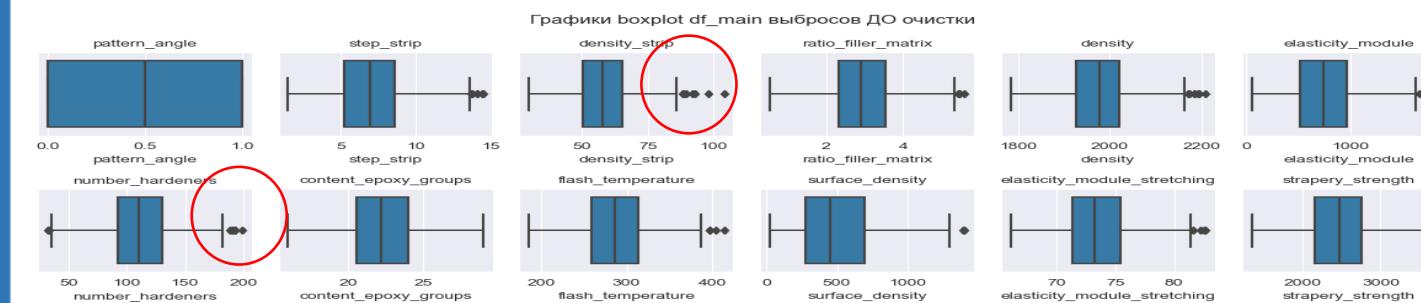
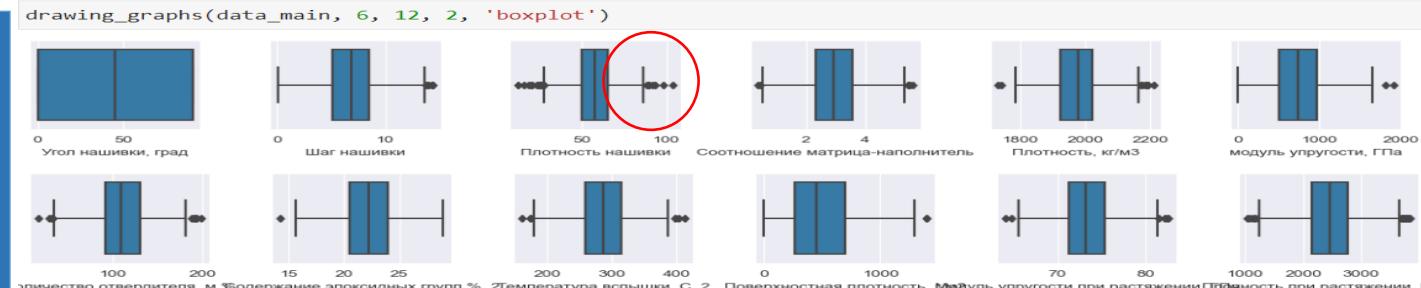


ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

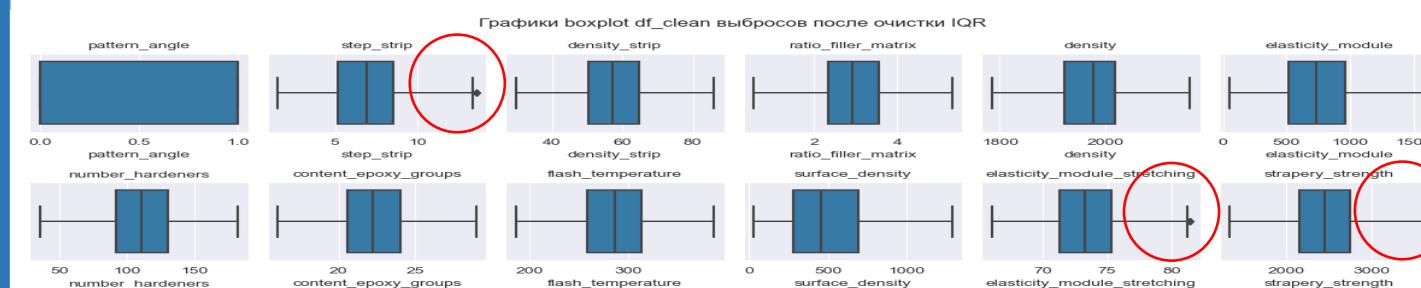
## Графики boxplot

### Этапы очистки

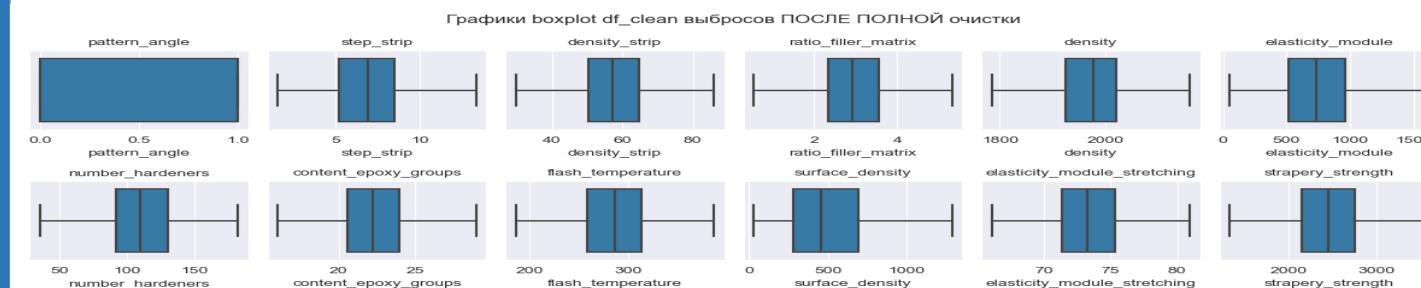
До начала мероприятий по очистке



Этап замены подозрительных значений на среднее 5 соседей (110) строк



Этап очистки методом IQR (55 строк удалено)  
Осталось 945 строк



Этап очистки СЛЕВА и СПРАВА (10 строк удалено)  
Осталось 935 строк  
**Чистый датасет!**

## БЛОК № 2

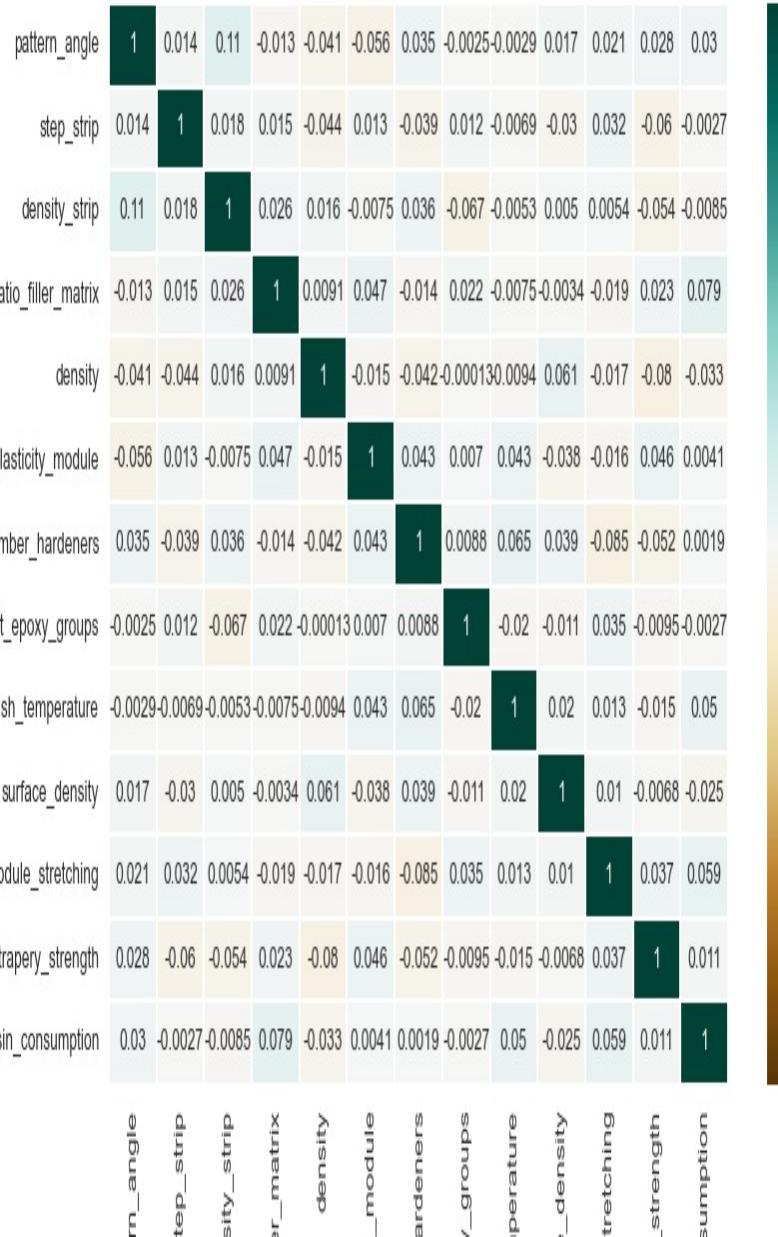
### Корреляционный анализ:

- ✓ Исследование падения корреляции о числа выборки:

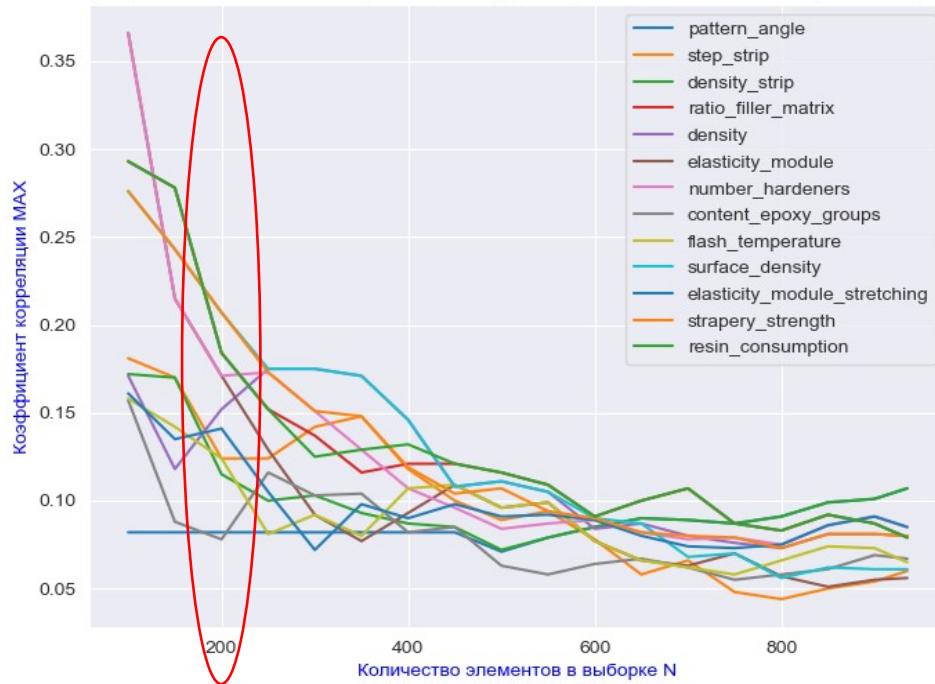
- Построили корреляционную матрицу датасета, коэффициенты близки к 0;
- Создали массив изменения MAX коэф. корреляции в зависимости от N числа выборки;
- Построим графики , как общий ,так и для каждого параметра отдельно – Коэффициенты стремительно падают с увеличением N;
- При N =200 еще сохраняется приемлемы уровень коэф.



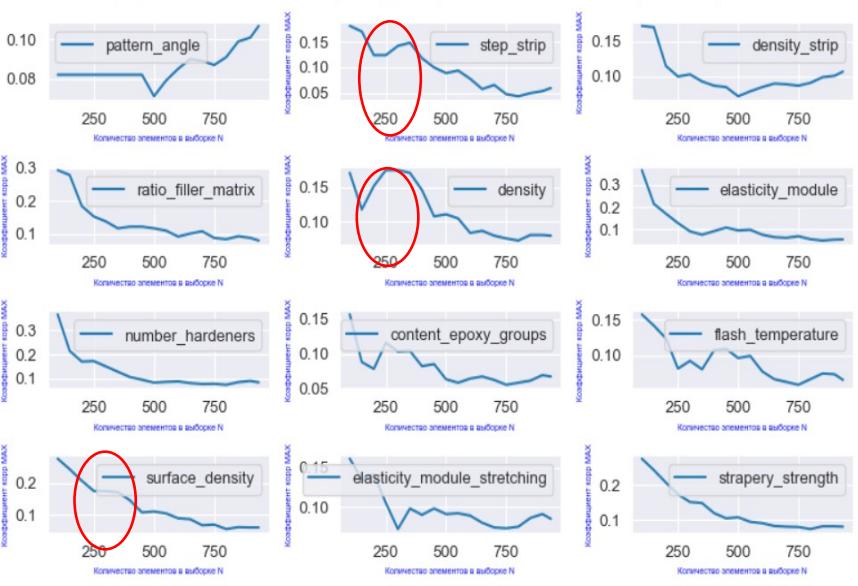
Корреляционная матрица очищенного DataSet data\_main\_clean



Графики изменения MAX размереа корреляции от количества N строк выборки



Графики изменения размера корреляции от количества N строк выборки



## БЛОК № 2

В соответствии с теорией хрупкого разрушения (теория Гриффита) [29] прочность ( $\sigma_p$ ) определяется удельной энергией ( $\alpha$ ) вновь образованной поверхности разрушения:  $(\sigma_p) = f(\alpha * E)$ , где ( $\sigma_p$ ) это наш параметр 'Прочность при растяжении' - 'strapery\_strength' (размерность в Мпа), а  $E$  это 'Модуль упругости при растяжении' - 'elasticity\_module\_stretching' (размерность в ГПа, т.е в 1000 раз больше чем  $\sigma_p$ ). Учитывая такую физическую связь между параметрами, введем один новый признак/столбец ALFA- 'удельную энергию' ( $\alpha$ ) =  $\sigma_p / E$  в наш новый датасет 'df\_add\_col'.

$$\sigma_p = f(\alpha * E),$$

Введем новый признак 'ALFA' - 'удельную энергию'  $\alpha = \sigma_p / E$

Датасфрейм df\_add\_col:

Тип данных: float64

Количество параметров: 14

Количество записей: 1000



```
df_Train_lineReg = df_add_col.sample(n=200, random_state=42)
```

```
df_Test_lineReg = df_add_col.sample(n=100, random_state=50)
```

```
# Максимальные корреляции в нашем df для полногс  
print(data_main_clean.corr().abs().apply(lambda
```

pattern_angle	0.107487
step_strip	0.060461
density_strip	0.107487
ratio_filler_matrix	0.078895
density	0.080297
elasticity_module	0.056466
number_hardeners	0.084787
content_epoxy_groups	0.066516
flash_temperature	0.065471
surface_density	0.060695
elasticity_module_stretching	0.084787
strapery_strength	0.080297
resin_consumption	0.078895
dtype: float64	

```
# Максимальные корреляции в нашем df для СЛУЧАЙНЫХ 200 строк  
print(df_add_col.sample(n=200, random_state=42).corr().abs().
```

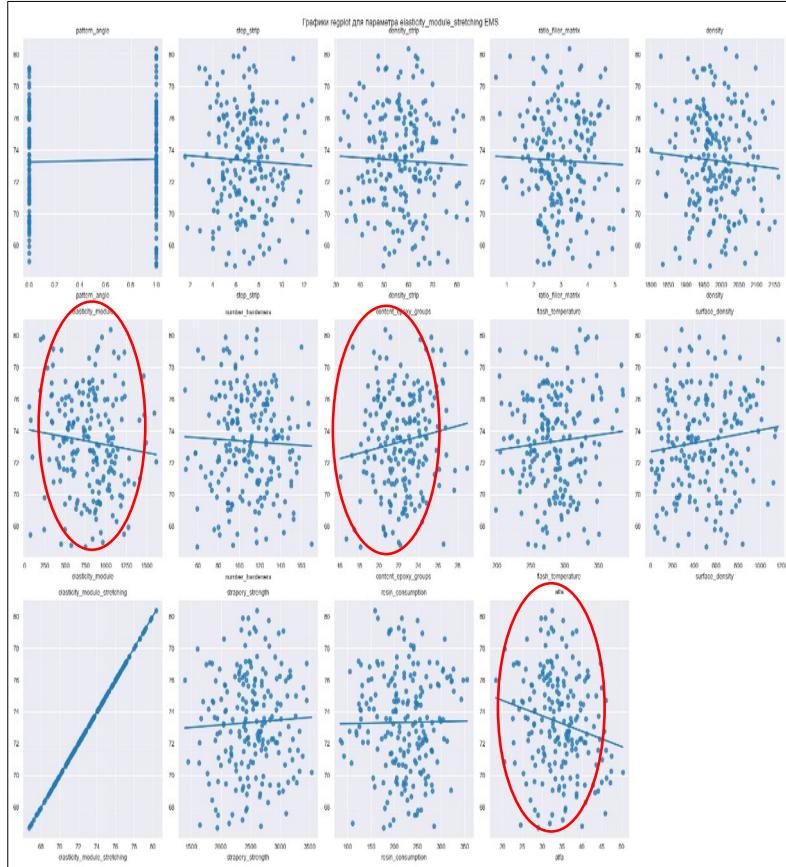
pattern_angle	0.148352
step_strip	0.098215
density_strip	0.148352
ratio_filler_matrix	0.223873
density	0.092884
elasticity_module	0.153614
number_hardeners	0.080614
content_epoxy_groups	0.132030
flash_temperature	0.097019
surface_density	0.153614
elasticity_module_stretching	0.190468
strapery_strength	0.971485
resin_consumption	0.223873
alfa	0.971485
dtype: float64	



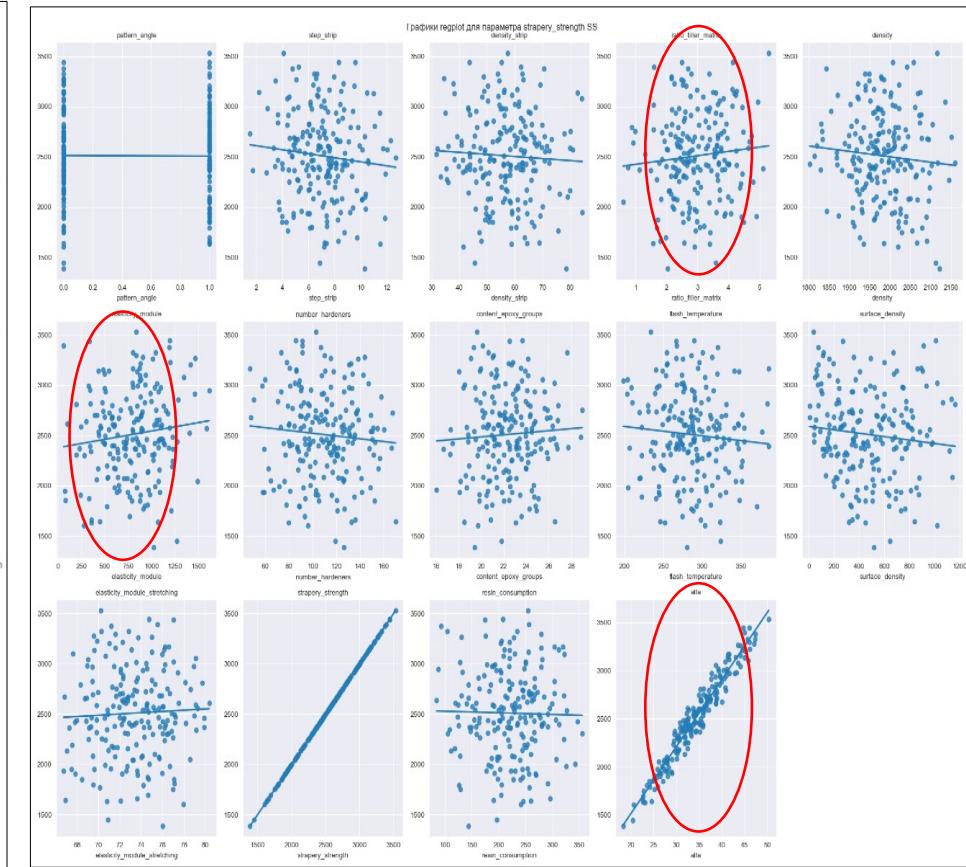
### Корреляционный анализ целевых переменных:

- Провели корр. анализ для 2-х целевых переменных  
'elasticity\_module\_stretching\_EMS'  
и 'strapery\_strength\_SS'
- На основе значений в корр. матрицах и при графическом анализе регрессионной прямой на точечных графиках осуществили выбор фитч) для целевых переменных для анализа в регрессионных моделях

'elasticity\_module\_stretching\_EMS'



'strapery\_strength\_SS'



Вероятно, можно проводить регрессивный анализ по параметрам

'elasticity\_module\_stretching\_EMS'

'elasticity\_module',  
'content\_epoxy\_groups'  
'alfa'

'strapery\_strength\_SS'

'ratio\_filler\_matrix',  
'elasticity\_module'  
'alfa'.



## БЛОК № 3

### Предобработка данных:

#### ✓ Нормализация данных:

- Нормализуем данные `MinMaxScaler()`;
- Построим график плотности ядра;
- Проверим результат `MinMaxScaler()`;
- Построим графики `MinMaxScaler()`;

#### ✓ Собираем датасеты для моделей

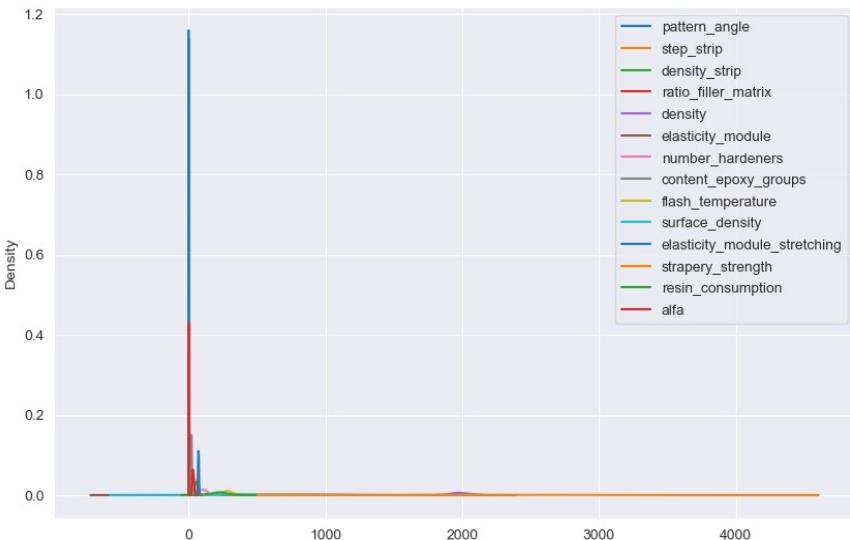
- Датасет для Train из  $N = 200$  строк
- Датасет для Туле из  $N = 100$  строк;



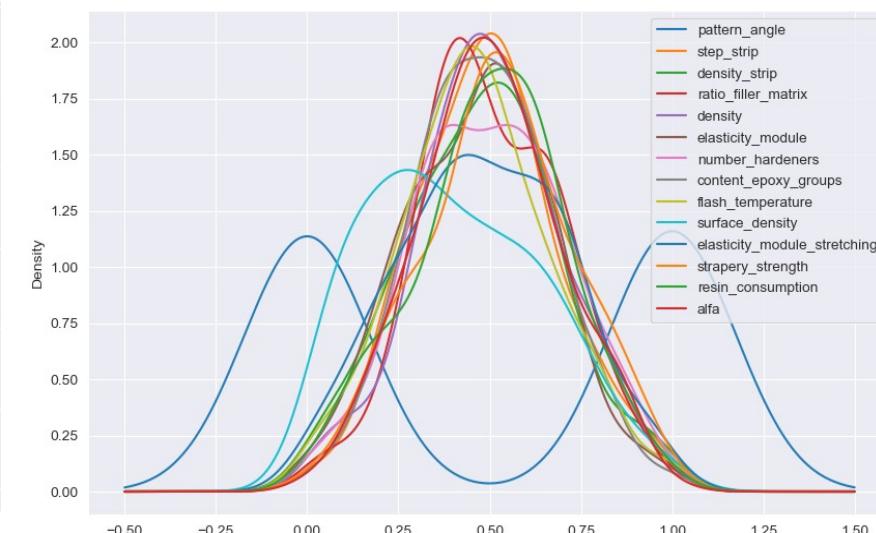
ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

### Нормализация данных MinMaxskaler (на примере `data_Train_EMS`)

Распределение плотности параметров `data_Train_ML`



Распределение плотности параметров `data_Train_norm` после нормализации



Собираем dataset для регрессий ( $y$  и  $X$ ) для `data_Train_EMS` и `data_Test_EMS`  
(оставляем только отобранные в Блоке корреляции фичи)

```
# Train для _EMS
```

```
data_Train_EMS = data_Train_norm.copy()
```

```
y_Train_EMS = data_Train_EMS["elasticity_module_stretching"]
```

```
X_Train_EMS = data_Train_EMS.drop(columns=['pattern_angle', 'step_strip', 'density_strip',
                                             'ratio_filler_matrix', 'density', 'number_hardeners',
                                             'flash_temperature', 'surface_density',
                                             'elasticity_modul_stretching', 'strapery_strength', 'resin_consumption'])
```

```
X_Train_EMS.shape
```

```
(200, 3)
```

```
# Test - для _EMS
```

```
data_Test_EMS = data_Test_norm.copy()
```

```
y_Test_EMS = data_Test_EMS["elasticity_module_stretching"]
```

```
X_Test_EMS = data_Test_EMS.drop(columns=['pattern_angle', 'step_strip', 'density_strip',
                                             'ratio_filler_matrix', 'density', 'number_hardeners',
                                             'flash_temperature', 'surface_density',
                                             'elasticity_modul_stretching', 'strapery_strength', 'resin_consumpti
```

```
X_Test_EMS.shape
```

```
(100, 3)
```

## Регрессионный анализ:

### ✓ Используемые модели и показатели:

- В данной работе рассмотрены 6 различных моделей регрессии;
- Теория по моделям рассмотрена в Пояснительной записке к ВКР;
- Выбраны в качестве показателей оценки работы моделей 7 различных показателей;
- Созданы датасеты для 2-х целевых переменных для записи и последующего анализа оценочных параметрами;
- Создана внешняя функция для расчета 7 показателей для 5 моделей;
- Модель sm.OLS – статистическая наименьших квадратов рассмотрена отдельно;



```
1. lr    = LinearRegression() - линейная регрессия
2. knr   = KNeighborsRegressor() - метод ближайших соседей
3. rfr   = RandomForestRegressor() - случайный лес
4. svr   = SVR() - метод опорных векторов
5. mlpr  = MLPRegressor() - многослойный перцептрон - обучение с 'учителем'
6. smlr  = sm.OLS() - statsmodels.regression.linear_model.OLS() статистических квадратов
```

В итоге сравним полученные результаты по расчетным параметрам:

- R2	- коэффициент детерминации
- MAE	- mean_absolute_error() Средняя абсолютная ошибка
- MSE	- mean_squared_error() Среднеквадратическая ошибка
- MaxER	- max_error() Максимальная ошибка
- RMSE	- среднеквадратическая ошибка
- MCVS	- Среднее значение оценки кросс-валидации
- StdDS	- Статистическая ошибка оценки кросс-валидации

```
# Создаем df для записи результатов работы моделей регрессии для целевых параметров
df_error_calc_EMS = pd.DataFrame(columns=['Model', 'R2', 'MAE', 'MSE', 'MaxER', 'RMSE', 'MCVS', 'StdDS'])
df_error_calc_SS = pd.DataFrame(columns=['Model', 'R2', 'MAE', 'MSE', 'MaxER', 'RMSE', 'MCVS', 'StdDS'])
```

```
# Инициализируем модели регрессии.
# Модель sm.OLS() - statsmodels.regression.linear_model.OLS() будет рассмотрена ПОСЛЕ этих типов
```

```
model_lr = LinearRegression()
model_knr = KNeighborsRegressor()
model_rfr = RandomForestRegressor()
model_svr = SVR()
model_mlpr = MLPRegressor()
```

```
# Переменные для расчетов
index_err_EMS = 0 # счетчик для 'elasticity_module_stretching'
index_err_SS = 0 # счетчик для 'strapery_strength'
cv_err = 5 # количество интервалов при расчете скросс-валидации
param_EMS = 'elasticity_module_stretching'
param_SS = 'strapery_strength'
```

## БЛОК № 3

### Регрессионный анализ:

#### ✓ Числовые показатели:

- Проведены исследования для двух целевых переменных 'elasticity\_module\_stretching\_EMS' и 'strapery\_strength\_SS' по 5 различным моделям;
- Для целевой переменной 'elasticity\_module\_stretching\_EMS' полученные значения говорят о полном отсутствии возможности прогнозирования этой переменной на основе использованных моделей;
- Для целевой переменной 'strapery\_strength\_SS' полученные значения говорят о ВОЗМОЖНОМ прогнозировании этой переменной на основе использованных моделей;



### Результаты тестирования моделей регрессии для df\_error\_calc\_EMS и df\_error\_calc\_SS

df_error_calc_EMS								
	Model	R2	MAE	MSE	MaxER	RMSE	MCVS	StdDS
0	model_lr	0.037741	0.176637	0.049992	0.633008	0.223588	-0.096812	0.168265
1	model_knr	0.026493	0.185602	0.050576	0.622712	0.224891	-0.317342	0.275991
2	model_rfr	0.114498	0.178494	0.046004	0.507401	0.247454	-0.385	0.213267
3	model_svr	-0.075523	0.192128	0.055876	0.612608	0.269565	-0.188618	0.274704
4	model_mlpr	-0.383212	0.217781	0.071861	0.626519	0.259056	-0.486948	0.443502

df_error_calc_SS								
	Model	R2	MAE	MSE	MaxER	RMSE	MCVS	StdDS
0	model_lr	0.887107	0.055714	0.005047	0.183987	0.071044	0.932461	0.038684
1	model_knr	0.808478	0.072267	0.008563	0.258085	0.092534	0.868781	0.051964
2	model_rfr	0.887779	0.056953	0.005017	0.177833	0.317756	0.907745	0.050048
3	model_svr	0.843971	0.066948	0.006976	0.230442	0.292071	0.832279	0.059178
4	model_mlpr	0.841984	0.068706	0.007065	0.2047	0.281166	0.865347	0.095879

Лучшая\_EMS  
rfr R2= 0.114

Худшая\_EMS  
mlrp R2=-0.383

Лучшая\_SS  
rfr R2= 0.888

Худшая\_SS  
knr R2=0.808

## Регрессионный анализ Результаты тестирования моделей

### ✓ Графический анализ тестирования моделей:

- Построим графики для тестовых и прогнозных значений.
- Здесь приведены графики для двух моделей- Линейной и К0Ближайших соседей для двух целевых переменных.

### Результаты тестирования моделей регрессии (на примере LR и KNN) для df\_error\_calc\_EMS и df\_error\_calc\_SS

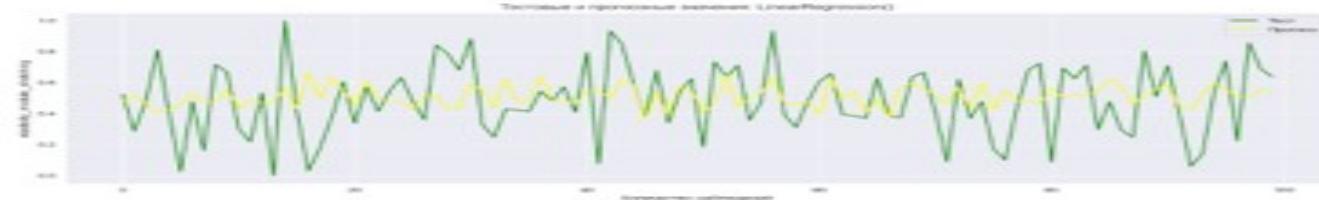


Рис. 60 Линейная модель для df\_error\_calc\_EMS

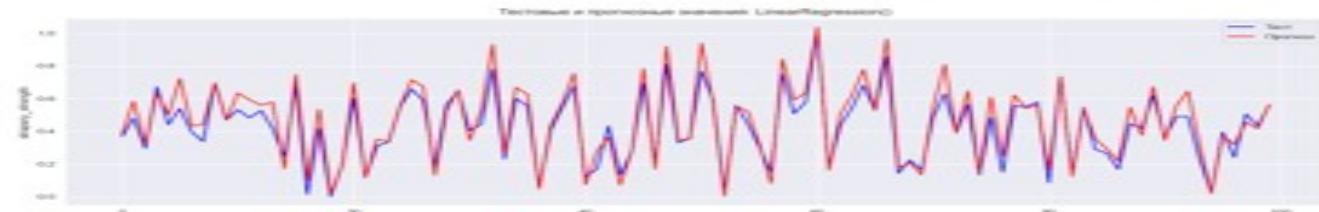


Рис.61 Линейная модель для df\_error\_calc\_SS

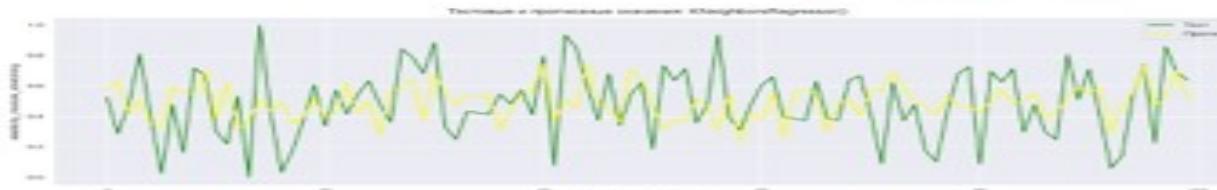


Рис.62 KNeighborsRegressor для df\_error\_calc\_EMS

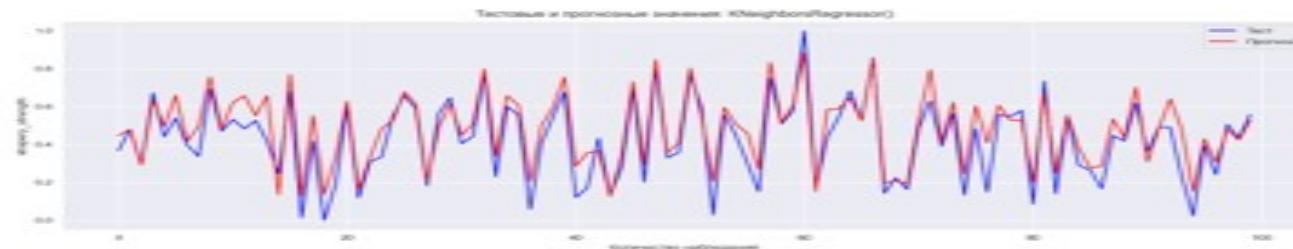


Рис.63 KNeighborsRegressor для df\_error\_calc\_SS



## БЛОК № 3

### Регрессионный анализ Результаты тестирования моделей

#### ✓ Графический анализ

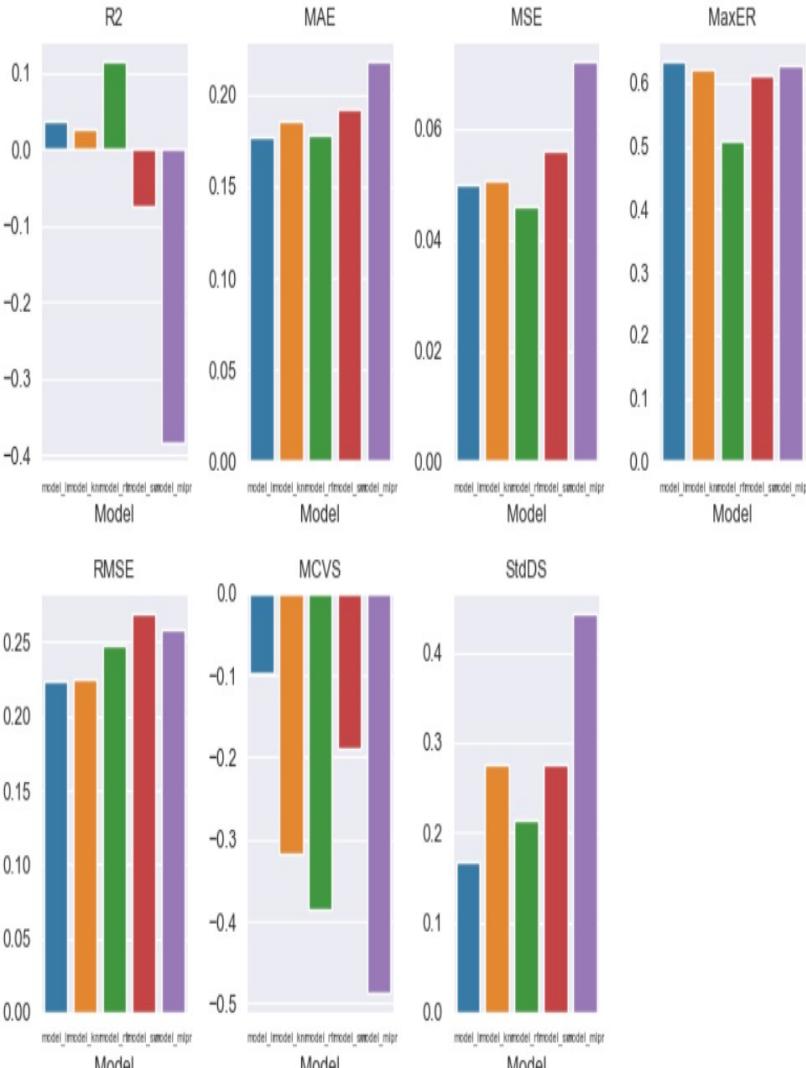
- Проведены исследования для двух целевых переменных 'elasticity\_module\_stretching\_EMS' и 'strapery\_strength\_SS' по 5 различным моделям;
- Для целевой переменной 'elasticity\_module\_stretching\_EMS' построенные графики говорят о полном отсутствии возможности прогнозирования этой переменной на основе использованных моделей – R2\_rfr = 0,11 – близок нулю!!!;
- Для целевой переменной 'strapery\_strength\_SS' полученные значения говорят о ВОЗМОЖНОМ прогнозировании этой переменной на основе использованных моделей, R2\_rfr = 0.888 и близок к 1!!!!



ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

### 'elasticity\_module\_stretching\_EMS'

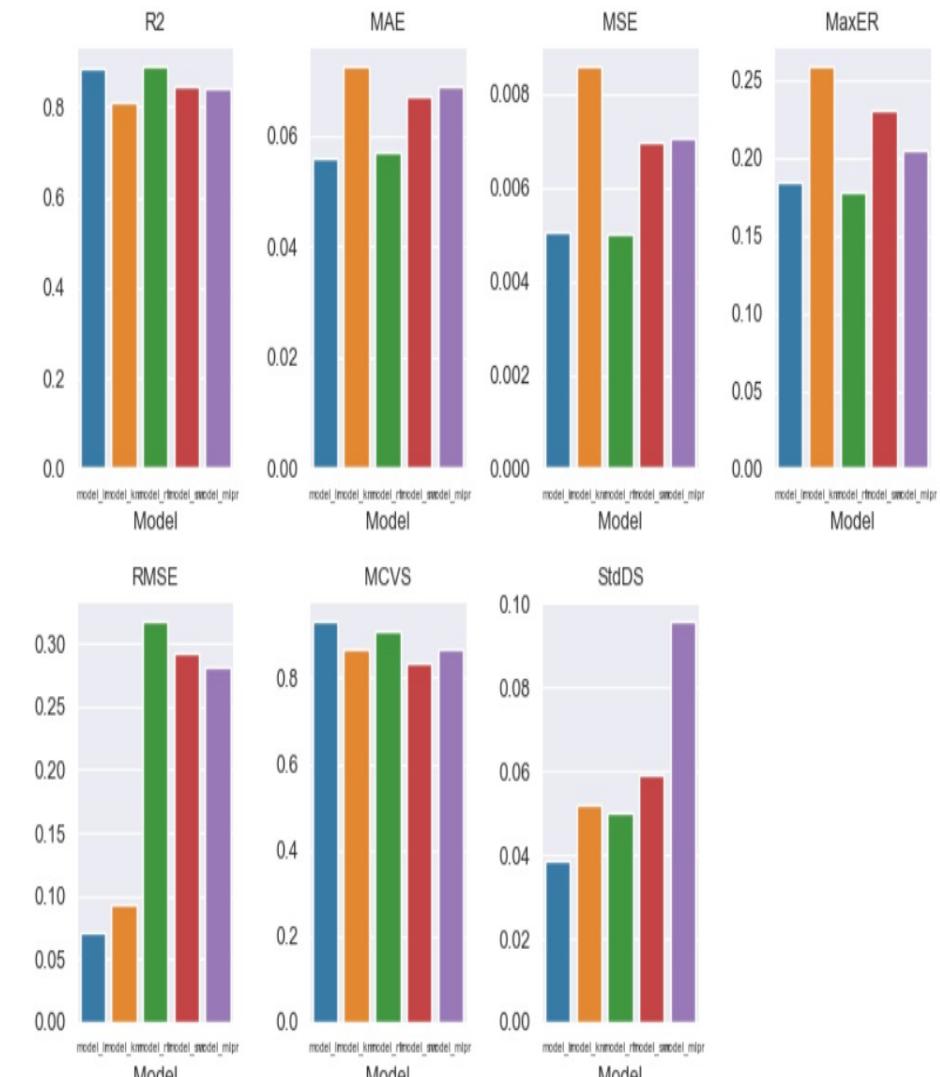
Графики изменения размера оценочного параметра EMS от типа модели регрессии



Лучшая rfr R2= 0.114

### 'strapery\_strength\_SS'

Графики изменения размера оценочного параметра SS от типа модели регрессии



Лучшая rfr R2= 0.888!!!

## БЛОК № 3

# Регрессионный анализ Результаты тестирования

### ✓ Тестирование модели sm.OLS:

- Проведены исследования для двух целевых переменных elasticity\_module\_stretching\_EMS' и 'strapery\_strength\_SS' по 5 различным моделям;
- Для целевой переменной 'elasticity\_module\_stretching\_EMS' построенные графики говорят о полном отсутствии возможности прогнозирования этой переменной на основе использованных моделей – R2\_rfr = 0,058 – близок нулю!!!;
- Для целевой переменной 'strapery\_strength\_SS' полученные значения говорят о ВОЗМОЖНОМ прогнозировании этой переменной на основе использованных моделей, R2\_rfr = 0.944 и близок к 1!!!!



ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

## 'elasticity\_module\_stretching\_EMS'

```
# OLS - ordinary least squares - метод наименьших квадратов
model_sm = sm.OLS(y_Train_EMS, X_train_EMS_add).fit()
```

```
model_sm.params
```

```
const          0.548250
elasticity_module -0.068939
content_epoxy_groups 0.159847
alfa           -0.219414
dtype: float64
```

```
model_sm.summary()
```

OLS Regression Results						
Dep. Variable:	elasticity_module_stretching	R-squared:	0.058			
Model:	OLS	Adj. R-squared:	0.044			
Method:	Least Squares	F-statistic:	4.046			
Date:	Thu, 20 Apr 2023	Prob (F-statistic):	0.00808			
Time:	19:07:51	Log-Likelihood:	17.550			
No. Observations:	200	AIC:	-27.10			
Df Residuals:	196	BIC:	-13.91			
Df Model:	3					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	0.5482	0.070	7.850	0.000	0.411	0.686
elasticity_module	-0.0689	0.081	-0.856	0.393	-0.228	0.090
content_epoxy_groups	0.1598	0.086	1.866	0.064	-0.009	0.329
alfa	-0.2194	0.083	-2.635	0.009	-0.384	-0.055
Omnibus:	1.904	Durbin-Watson:	1.884			
Prob(Omnibus):	0.386	Jarque-Bera (JB):	1.608			
Skew:	-0.072	Prob(JB):	0.448			
Kurtosis:	2.585	Cond. No.	8.91			

Худшая sm.OLS\_EMS R2=0.058

## 'strapery\_strength\_SS'

```
# OLS - ordinary least squares - метод наименьших квадратов
model_sm = sm.OLS(y_Train_SS, X_train_SS_add).fit()
```

```
model_sm.params
```

```
const          0.011492
ratio_filler_matrix 0.001450
elasticity_module -0.021967
alfa           1.043117
dtype: float64
```

```
model_sm.summary()
```

OLS Regression Results						
Dep. Variable:	strapery_strength	R-squared:	0.944			
Model:	OLS	Adj. R-squared:	0.943			
Method:	Least Squares	F-statistic:	1106.			
Date:	Thu, 20 Apr 2023	Prob (F-statistic):	1.54e-122			
Time:	19:08:55	Log-Likelihood:	321.08			
No. Observations:	200	AIC:	-634.2			
Df Residuals:	196	BIC:	-621.0			
Df Model:	3					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	0.0115	0.014	0.804	0.422	-0.017	0.040
ratio_filler_matrix	0.0014	0.018	0.080	0.936	-0.034	0.037
elasticity_module	-0.0220	0.018	-1.249	0.213	-0.057	0.013
alfa	1.0431	0.018	56.989	0.000	1.007	1.079
Omnibus:	2.866	Durbin-Watson:	1.841			
Prob(Omnibus):	0.239	Jarque-Bera (JB):	2.034			
Skew:	0.025	Prob(JB):	0.362			
Kurtosis:	2.508	Cond. No.	8.22			

Лучшая sm.OLS\_SS R2= 0.944!!!

## БЛОК № 4

# Создание нейронной сети

### ✓ Выбор количества слоев и нейронов:

- Загрузили очищенный датасет из Блока 1 из 13 параметров
- Обосновали выбор количества слоев и провели расчет количества нейронов в каждом слое;

## БЛОК № 1

Датасет 'data\_main\_clean'

## БЛОК № 4

```
# У нас 1 имеется целевая переменная - 'ratio_filler_matrix' RFM - Соотношение матрица-наполнитель  
# Следовательно, у нас будет 1 нейрон на выходе  
# На входе DF у нас остаются 12 признаков.  
# Следовательно у нас на входе нейронной сети будет 12 нейронов.
```

```
# Проведем расчет количества нейронов в 2 - х скрытых слоях по правилу геометрической пирамиды:  
# k - # число нейронов в скрытом слое  
n = 12 # число нейронов во входном слое  
m = 1 # число нейронов в выходном слое  
r = (n/m)**(1.0/3.0)  
k1 = m * r**2 # число нейронов в 1-м скрытом слое  
k2 = m * r # число нейронов во 2-м скрытом слое
```

```
print('число нейронов во входном слое - ', n)  
print('число нейронов в выходном слое - ', m)  
print('число нейронов в 1-м скрытом слое - ', math.ceil(k1)) # округляем в большую сторону  
print('число нейронов во 2-м скрытом слое - ', math.ceil(k2)) # округляем в большую сторону
```

число нейронов во входном слое - 12

число нейронов в выходном слое - 1

число нейронов в 1-м скрытом слое - 6

число нейронов во 2-м скрытом слое - 3



# Создание нейронной сети

## ✓ Создание архитектуры нейронной сети:

- Выбрали активационную функцию SELU;
- Выбрали оптимизатор ADAM;
- Выбрали оценочный параметр MAE;
- Создали индикатор переобучения модели;

```
# Создаем нейросеть из 4 слоев:
# 1-й - Входной слой размерностью n = 12 - по количеству параметров в нашем df
# 2-й - Скрытый слой размерностью k1 = 6 - получен из расчета по правилу геометрического прогрессии
# 3-й - Скрытый слой размерностью k2 = 3 - получен из расчета по правилу геометрического прогрессии
# 4-й - Выходной слой размерностью m = 1 - по количеству целевых параметров

model_RFМ.add(Dense(12)) # входной полносвязный слой
model_RFМ.add(Dense(6, kernel_initializer='lecun_normal', activation='selu'))
model_RFМ.add(AlphaDropout(0.25))
model_RFМ.add(Dense(3, kernel_initializer='lecun_normal', activation='selu'))
model_RFМ.add(AlphaDropout(0.25))
model_RFМ.add(Dense(1, activation= 'linear')) # выходной слой
```

• [30]: # stop\_RFМ - индикатор для остановки обучения нейросети, когда она перестает улучшаться в течении 15 эпох

```
stop_RFМ = EarlyStopping(monitor='val_loss', min_delta=0, patience=15, verbose=1, mode='auto')
```

[31]: # Компиляция модели. Вызывать будем оптимизатор по имени Adam.

# Learning\_rate=0.02 - Шаг обучения 0.02

# LossM -функция ошибки - "средняя абсолютная ошибка"

```
model_RFМ.compile(optimizer = tf.optimizers.Adam(learning_rate=0.02), loss = 'mean_absolute_error')
```



# Создание нейронной сети

## ✓ Обучение модели:

- Обучим нейросеть;
- Модель прервала работу по условию индикатора переобучения на 35 эпохе;
- Посмотрим на потери модели;
- Лучшие показатели достигнуты на 9 эпохе;
- МАЕ модели составило 0.1634.



```
[32]: %%time
history_RFM = model_RFM.fit(
    X_train_RFM, # входящая выборка
    y_train_RFM, # целевая выборка
    batch_size = 32, # вычисляем градиенты каждые 32 наблюдений
    epochs=100, # max 100 эпох, stop_RFM у нас создан
    verbose=1, # индикатор выполнения
    validation_split = 0.2, # доля обучающих данных, для проверки
    callbacks = [stop_RFM] # условие остановки при прекращении обучения в течении 10 эпох
)

17/17 [=====] - 0s 3ms/step - loss: 0.1506 - val_loss: 0.1467
Epoch 29/100
17/17 [=====] - 0s 3ms/step - loss: 0.1532 - val_loss: 0.1436
Epoch 30/100
17/17 [=====] - 0s 3ms/step - loss: 0.1514 - val_loss: 0.1736
Epoch 31/100
17/17 [=====] - 0s 3ms/step - loss: 0.1569 - val_loss: 0.1443
Epoch 32/100
17/17 [=====] - 0s 3ms/step - loss: 0.1516 - val_loss: 0.1401
Epoch 33/100
17/17 [=====] - 0s 3ms/step - loss: 0.1514 - val_loss: 0.1588
Epoch 34/100
17/17 [=====] - 0s 3ms/step - loss: 0.1535 - val_loss: 0.1412
Epoch 35/100
17/17 [=====] - 0s 3ms/step - loss: 0.1502 - val_loss: 0.1411
Epoch 35: early stopping
Wall time: 2.81 s
```

---

```
[33]: # Оценка модели по лучшему расчетному показателю MAE
evaluation_model = model_RFM.evaluate(x=X_test_RFM,
                                         y=y_test_RFM)
print("Оценка модели = ", evaluation_model)

9/9 [=====] - 0s 1ms/step - loss: 0.1634
Оценка модели = 0.1633787751197815
```

---

```
[ ]: # Лучший показатель MAE был достигнут во время 9 эпохи и составил MAE(9) = 0.1633
```

## БЛОК № 4

# Создание нейронной сети

### ✓ Итоги работы модели:

- Построим график потерь на тренировочной и тестовой выборках.
- Построим график результата работы модели графики «чистого» датасета

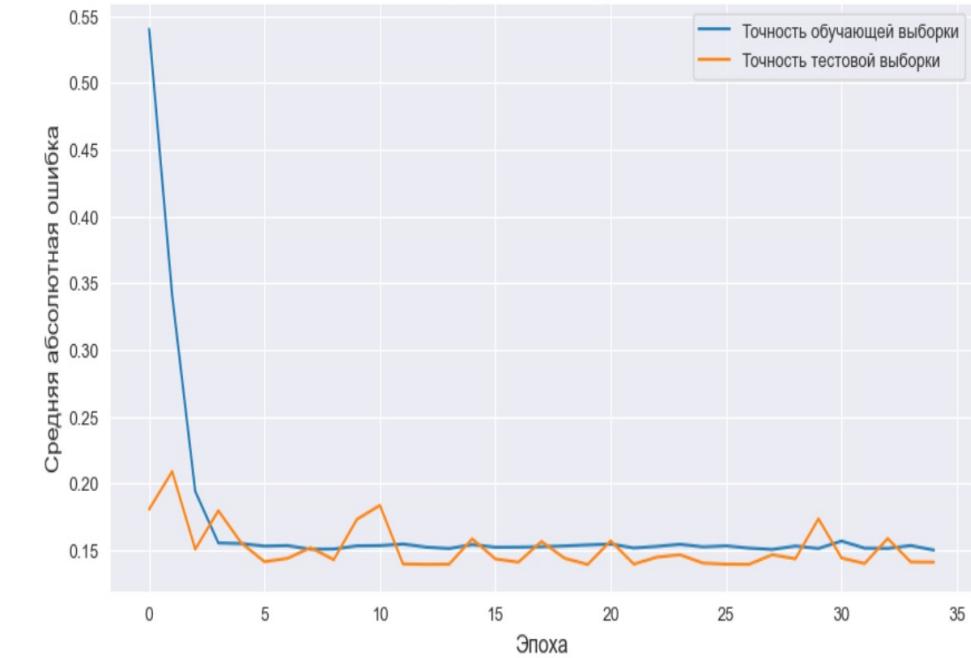
```
[34]: model_RFМ.summary()
```

Model: "sequential"

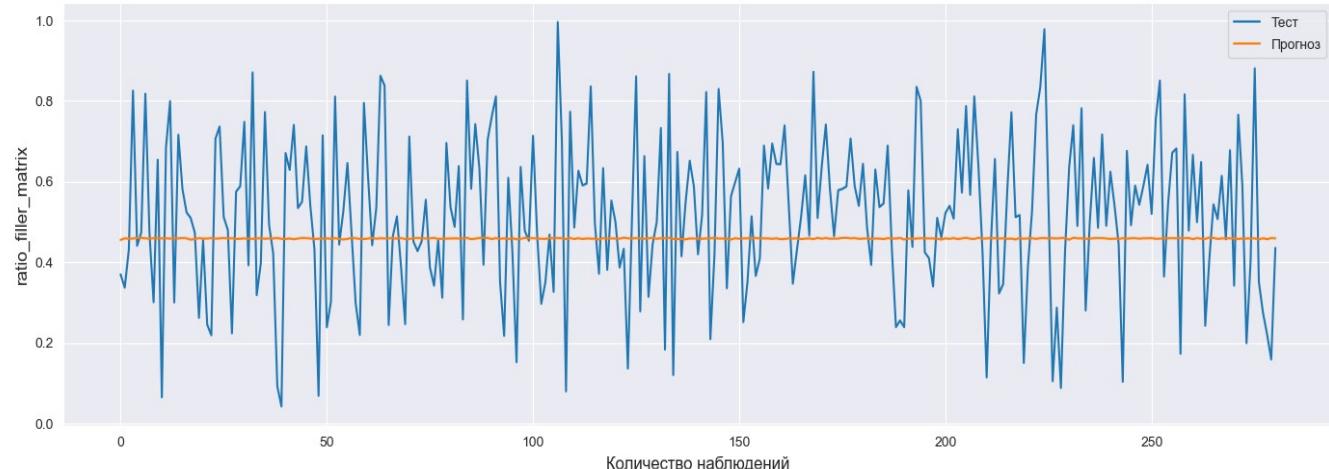
Layer (type)	Output Shape	Param #
<hr/>		
dense (Dense)	(None, 12)	156
dense_1 (Dense)	(None, 6)	78
alpha_dropout (AlphaDropout)	(None, 6)	0
dense_2 (Dense)	(None, 3)	21
alpha_dropout_1 (AlphaDropout)	(None, 3)	0
dense_3 (Dense)	(None, 1)	4
<hr/>		
Total params:	259	
Trainable params:	259	
Non-trainable params:	0	

```
[80]: # Построение графика ошибки MAE - mean_absolute_error функцией model_loss_plot  
model_loss_plot(history_RFМ, graf_name_path = r'save_fig\fig_block4_neuro\model_loss_MAE_flask.png')
```

График потерь модели



Динамика тестовых и прогнозных значений:



## Создание Приложения

### ✓ Приложение APP.PY:

- Создан шаблон main.html;
- Написан код Приложения по прогнозированию параметра “Соотношение матрица-наполнитель”;
- В коде Приложения использован модуль инверсирования предсказанного параметра в нормальный формат данных;



```

Файл Правка Выделение Вид Переход Выполнить Терминал Справка • app.py - Di
dj main.html app.py info.py 9+
app.py > ...
1 # Импортируем необходимые библиотеки для нашего приложения
2 import numpy as np
3 import pandas as pd
4 from sklearn.preprocessing import MinMaxScaler
5 import os
6 import pickle
7 import tensorflow as tf
8 from tensorflow import keras
9 import flask
10 from flask import Flask, request, render_template
11
12 app = flask.Flask(__name__, template_folder = 'templates')
13 @app.route('/', methods=['POST', 'GET'])
14 @app.route('/index', methods=['GET', 'POST'])
15 def main():
16     if flask.request.method == 'GET':
17         return render_template('main.html')
18     if request.method == 'POST':
19         X_pred = []
20         result = 1.0
21         # загрузка нашей модели model_RFm_flask
22         model = keras.models.load_model(r'./data_block5_flask/model_RFm_flask/')
23         # загрузка наших нормализаторов scaler_y.pkl и scaler_X.pkl
24         with open(r'./data_block5_flask/model_RFm_flask/scaler_X.pkl', 'rb') as f:
25             scaler_X = pickle.load(f)
26         with open(r'./data_block5_flask/model_RFm_flask/scaler_y.pkl', 'rb') as f:
27             scaler_y = pickle.load(f)
28         # получим данные из наших форм main.html и заполняем список X_pred из 12 параметров
29         for i in range(1,13,1):
30             exp = float(flask.request.form[f'param{i}'])
31             X_pred.append(exp)
32         # нормализуем полученные от пользователя данные с помощью загруженного scaler_X
33         X_norm = scaler_X.transform([X_pred])
34         # Отправляем нормализованные данные в модель и получаем нормализованную y_pred_norm
35         y_pred_norm = model.predict([[X_norm]])
36         # Инвертируем рассчитанную y_pred_norm в выходное значение
37         y_pred = scaler_y.inverse_transform(y_pred_norm)
38         #result = y_pred[0][0]
39         # указываем шаблон и прототип сайта для вывода
40         return render_template('main.html', result = y_pred)
41     # Запускаем приложен
42     if __name__ == '__main__':
43         app.run() # с отладкой debug = True

```

## БЛОК № 5

### ✓ Пользовательское приложение

- Сохранил модель нейронной сети для разработки веб-приложения для прогнозирования соотношения "матрица-наполнитель" в фреймворке Flask;
- При запуске приложения, пользователь переходит на: <http://127.0.0.1:5000/>;
- В открывшемся окне пользователю необходимо ввести в соответствующие ячейки значения в соответствии с рекомендованными и нажать на кнопку «Рассчитать значение».
- В случае ошибки –нажать кнопку “Очистить форму”;
- На выходе пользователь получает результат прогноза для значения параметра «Соотношение «матрица – наполнитель»».
- Приложение успешно работает

### ✓ Репозиторий на [github.com](https://github.com)

- [https://github.com/Grain1963/MHTS\\_Diploma](https://github.com/Grain1963/MHTS_Diploma)



ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР МГТУ им. Н. Э. Баумана

MHTS\_Diploma/translate\_list\_c + Дзен Diploma\_bloc... - JupyterLab 127.0.0.1:5000

Программа расчета  
прогнозного значения параметра  
«Соотношение матрица – наполнитель»

Заполните ячейки и нажмите "РАССЧИТАТЬ ЗНАЧЕНИЕ" или "ОЧИСТИТЬ ФОРМУ"

Название параметра	Рекомендованное значение от до	Значение параметра
Угол наливки, град	0.0 или 1.0	Заполните значение
Шаг наливки	1.6 - 13.0	Заполните значение
Плотность наливки	30.0 - 80.0	Заполните значение
Плотность, кг/м <sup>3</sup>	1800.0 - 2150.0	Заполните значение
Модуль упругости, ГПа	55.0 - 155.0	Заполните значение
Количество отвердителя, м. %	40.0 - 180.0	Заполните значение
Содержание эпоксидных групп, %	17.0 - 28.0	Заполните значение
Температура вспышки, С_2	190.0 - 380.0	Заполните значение
Поверхностная плотность, г/м <sup>2</sup>	30.0 - 120.0	Заполните значение
Модуль упругости при растяжении, ГПа	66.0 - 81.0	Заполните значение
Прочность при растяжении, МПа	1330.0 - 3600.0	Заполните значение
Потребление смолы, г/м <sup>2</sup>	80.0 - 350.0	Заполните значение

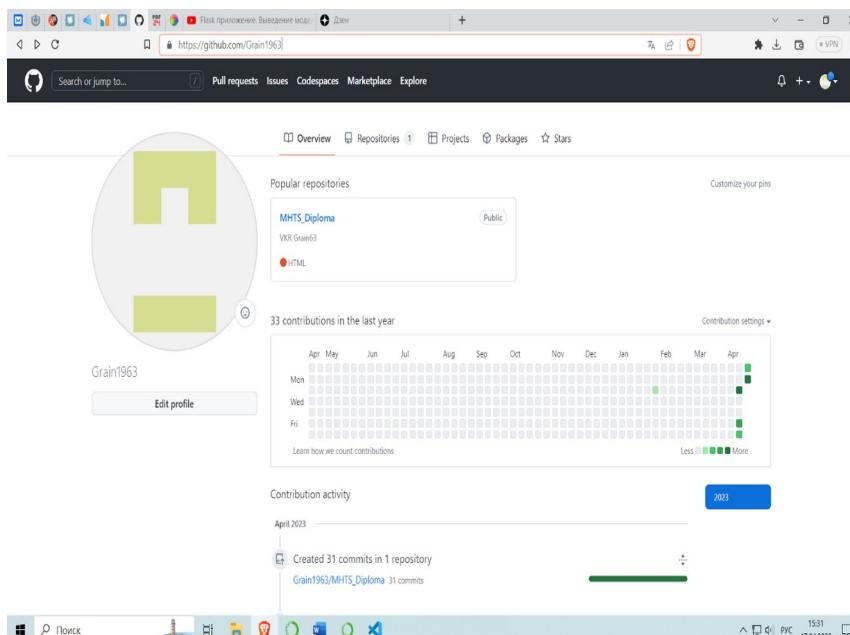
РАССЧИТАТЬ ЗНАЧЕНИЕ

ОЧИСТИТЬ ФОРМУ

Результат прогноза:

[12.741078]

личного центра МГТУ им. Н.Э. Баумана ©, апрель 2023 г.



Файл Правка Выделение Вид Переход Выполнить ... info.py - Diploma\_block5\_flask - Visual Studio Code

dj main.html app.py info.py 9+ X

info.py > ...

```
1
2 PS C:\Users\grain\Work_folder\MGTU Diplom\Diploma block5 flask> &
3 C:/Users/grain/anaconda3/envs/MyENV_JupiterPS
4 C:/Users/grain/Work_folder/MGTU Diplom\Diploma block5 flask> &
5 C:/Users/grain/anaconda3/envs/MyENV_JupiterLab/python.exe
6 c:/Users/grain/Work_folder/MGTU Diplom/Diploma_block5_flask/app.py
7 * Serving Flask app 'app'
8 * Debug mode: off
9 WARNING: This is a development server. Do not use it in a production deployment
10 Use a production WSGI server instead
11 * Running on http://127.0.0.1:5000
12 Press CTRL+C to quit
13 127.0.0.1 - - [17/Apr/2023 14:41:32] "GET / HTTP/1.1" 200 ]
```

https://github.com/Grain1963/MHTS\_Diploma

Grain1963/MHTS\_Diploma Public

Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Grain1963 Add files via upload 53871ca 36 minutes ago 31 commits

Diploma\_bloc5\_flask Add files via upload 40 minutes ago

Diploma\_naive\_old Add files via upload 2 days ago

data\_storage Add files via upload 36 minutes ago

modules\_def Add files via upload last week

profile\_report Add files via upload last week

save\_fig Add files via upload 42 minutes ago

Diploma\_block1\_clean.ipynb Add files via upload 5 hours ago

Diploma\_block2\_corr.ipynb Add files via upload 5 hours ago

Diploma\_block3\_regress.ipynb Add files via upload 5 hours ago

Diploma\_block4\_neuro.flask.ipynb Add files via upload 5 hours ago

README.md Update README.md 39 minutes ago

README.md