

**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»**

Слушатель

Кулабухов Константин Витальевич

Москва, 2022

Содержание

Содержание.....	2
Введение	3
1. Аналитическая часть.....	4
1.1. Загрузка и разведочный анализ данных	4
1.2. Предобработка данных	13
2. Практическая часть.....	14
2.1 Подготовка и обучение моделей.....	14
3. Создание нейронной сети	16
4 Создание удалённого репозитория и загрузка.....	25
5 Заключение	26
3.1. Список используемой литературы и веб ресурсы.	27

Введение

Композиционные материалы — это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т. е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита - железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

На входе имеются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Актуальность: Созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

1. Аналитическая часть

1.1. Загрузка и разведочный анализ данных

Для анализа данных были предоставлены два датасета со свойствами композитов в формате Excel: X_br.xlsx и X_nur.xlsx.

```
# загрузка датасета 1
data1 = pd.read_excel('X_br.xlsx')
data1
```

	Unnamed: 0	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2
0	0.0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000
1	1.0	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000
2	2.0	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000
3	3.0	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000
4	4.0	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
...
1018	1018.0	2.271346	1952.067902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669
1019	1019.0	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099
1020	1020.0	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764
1021	1021.0	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067
1022	1022.0	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342

1023 rows x 11 columns

Рисунок 1 X_br.xlsx

Согласно инфо содержит 1023 строки, и 11 колонок.

```
[3] data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1023 entries, 0 to 1022
Data columns (total 11 columns):
#   Column                                                                 Non-Null Count  Dtype
---  -
0   Unnamed: 0                                                            1023 non-null   float64
1   Соотношение матрица-наполнитель                                     1023 non-null   float64
2   Плотность, кг/м3                                                    1023 non-null   float64
3   модуль упругости, ГПа                                              1023 non-null   float64
4   Количество отвердителя, м.%                                         1023 non-null   float64
5   Содержание эпоксидных групп,%_2                                    1023 non-null   float64
6   Температура вспышки, С_2                                           1023 non-null   float64
7   Поверхностная плотность, г/м2                                      1023 non-null   float64
8   Модуль упругости при растяжении, ГПа                               1023 non-null   float64
9   Прочность при растяжении, МПа                                       1023 non-null   float64
10  Потребление смолы, г/м2                                             1023 non-null   float64
dtypes: float64(11)
memory usage: 88.0 KB
```

Рисунок 2 Инфо первого датасета

```
# загрузка датасета 2
data2 = pd.read_excel('X_nup.xlsx')
data2
```

	Unnamed: 0	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0.0	0.0	4.000000	57.000000
1	1.0	0.0	4.000000	60.000000
2	2.0	0.0	4.000000	70.000000
3	3.0	0.0	5.000000	47.000000
4	4.0	0.0	5.000000	57.000000
...
1035	1035.0	90.0	8.088111	47.759177
1036	1036.0	90.0	7.619138	66.931932
1037	1037.0	90.0	9.800926	72.858286
1038	1038.0	90.0	10.079859	65.519479
1039	1039.0	90.0	9.021043	66.920143

1040 rows × 4 columns

Рисунок 3 X_nup.xlsx

Согласно инфо содержит 1040 строк и 4 колонки.

```
data2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1040 entries, 0 to 1039
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Unnamed: 0            1040 non-null   float64
1   Угол нашивки, град    1040 non-null   float64
2   Шаг нашивки           1040 non-null   float64
3   Плотность нашивки     1040 non-null   float64
dtypes: float64(4)
memory usage: 32.6 KB
```

Рисунок 4 Инфо второго датасета

Необходимо объединить файлы. Объединение делать по индексу, тип объединения INNER.

```
# объединение датасетов по индексу, тип объединения inner
df = pd.merge(data1, data2, how = 'inner', left_index = True, right_index = True)
df.head()
```

Unnamed: 0_x	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
0	0.0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	3000.0
1	1.0	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	3000.0
2	2.0	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	3000.0
3	3.0	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	3000.0
4	4.0	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	3000.0

Рисунок 5 объединение датасетов

В результате объединения видим колонки `Unnamed: 0_x` и `Unnamed: 0_y`

Их необходимо удалить, так как они не содержат в себе ценной информации и являются аналогом индексов.

```
# дропаем их
df1 = df.drop(df.columns[[0,11]], axis=1)
df1.head()
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0

Рисунок 6, удаляем не нужные колонки

На следующем этапе проведем разведочный анализ данных – проверим пропуски, дубли, а также создадим гистограммы распределения каждой из переменной, диаграммы «ящика с усами» и попарные графики рассеяния точек.

```
# проверяем пропуски
df1.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 7, поиск пропусков в датасете

Пропуски не обнаружены.

```
# проверяем дубли
df.duplicated().sum()

0
```

Рисунок 8, поиск дублей в датасете

Дубли не обнаружены.

Для каждой колонки получим среднее, медианное значение, минимальное и максимальное значения, значения верхнего и нижнего квартилей, а так же стандартное отклонение.

```
# смотрим описание (для каждой колонки получаем среднее, медианное значение и т.д.)
df1.describe()
```

	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
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	482.731833	73.328571	2466.922843
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3.118983	485.628006
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64.054061	1036.856605
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71.245018	2135.850448
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73.268805	2459.524526
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75.356612	2767.193119
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82.682051	3848.436732

Рисунок 8, получаем описание датасета.

Из описания датасета видно, что данные имеют сильный разброс и в дальнейшем потребуется нормализовать их.

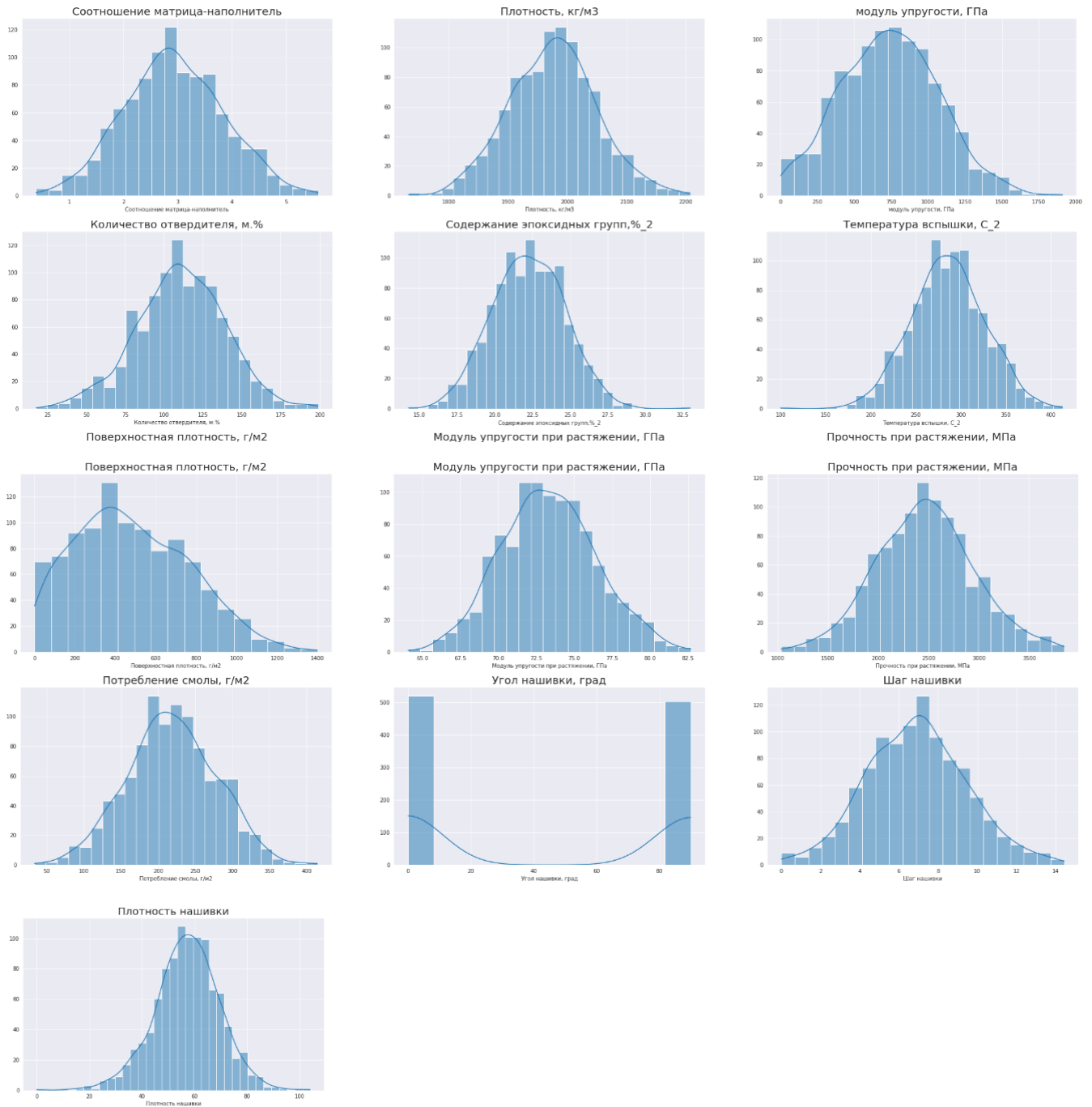


Рисунок 9 гистограммы распределения каждой переменной

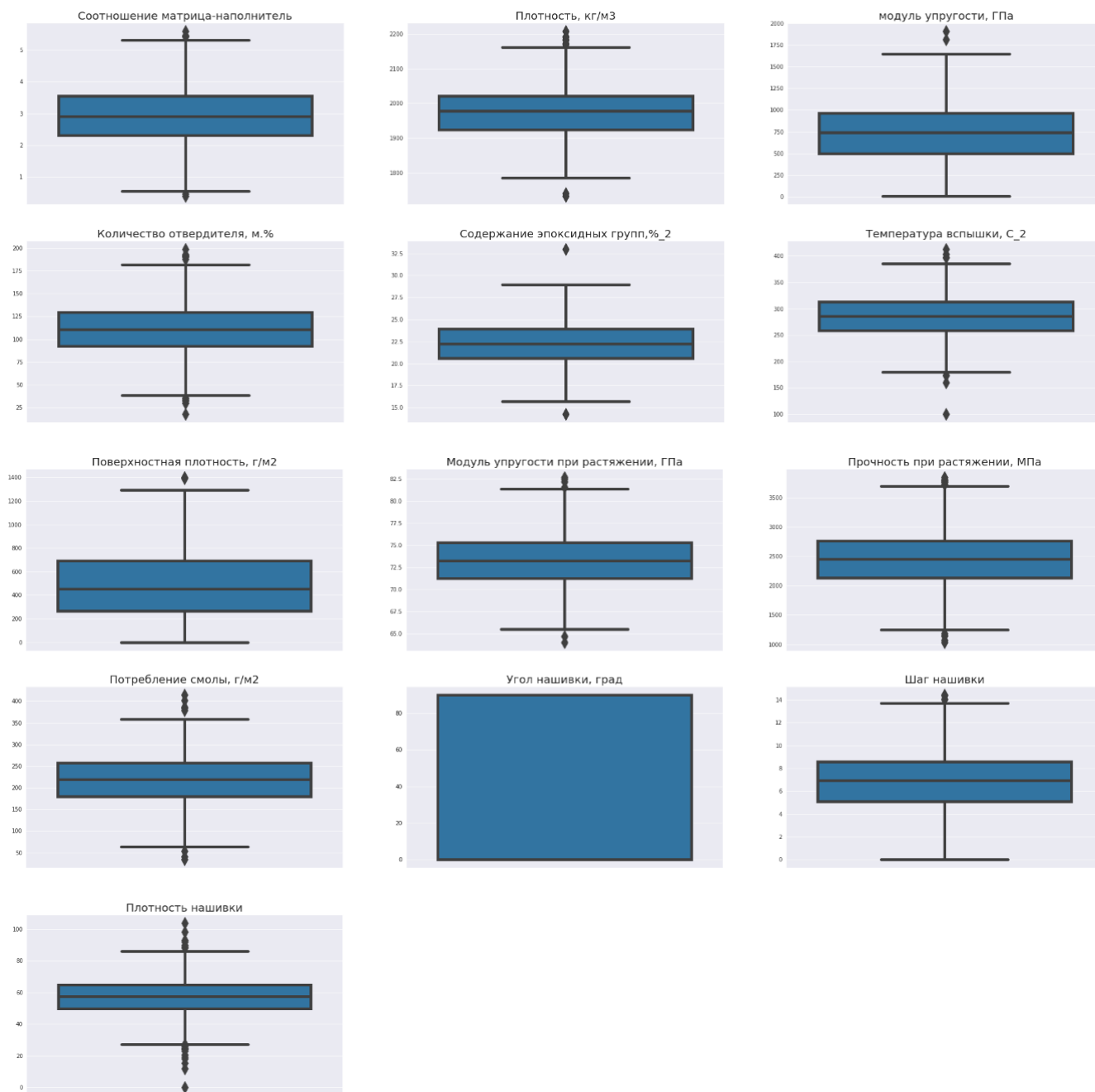


Рисунок 10 строим «ящики с усами».

По рисунку 10 можно сделать вывод, что в данных присутствуют выбросы, которые в дальнейшем потребуется обработать.

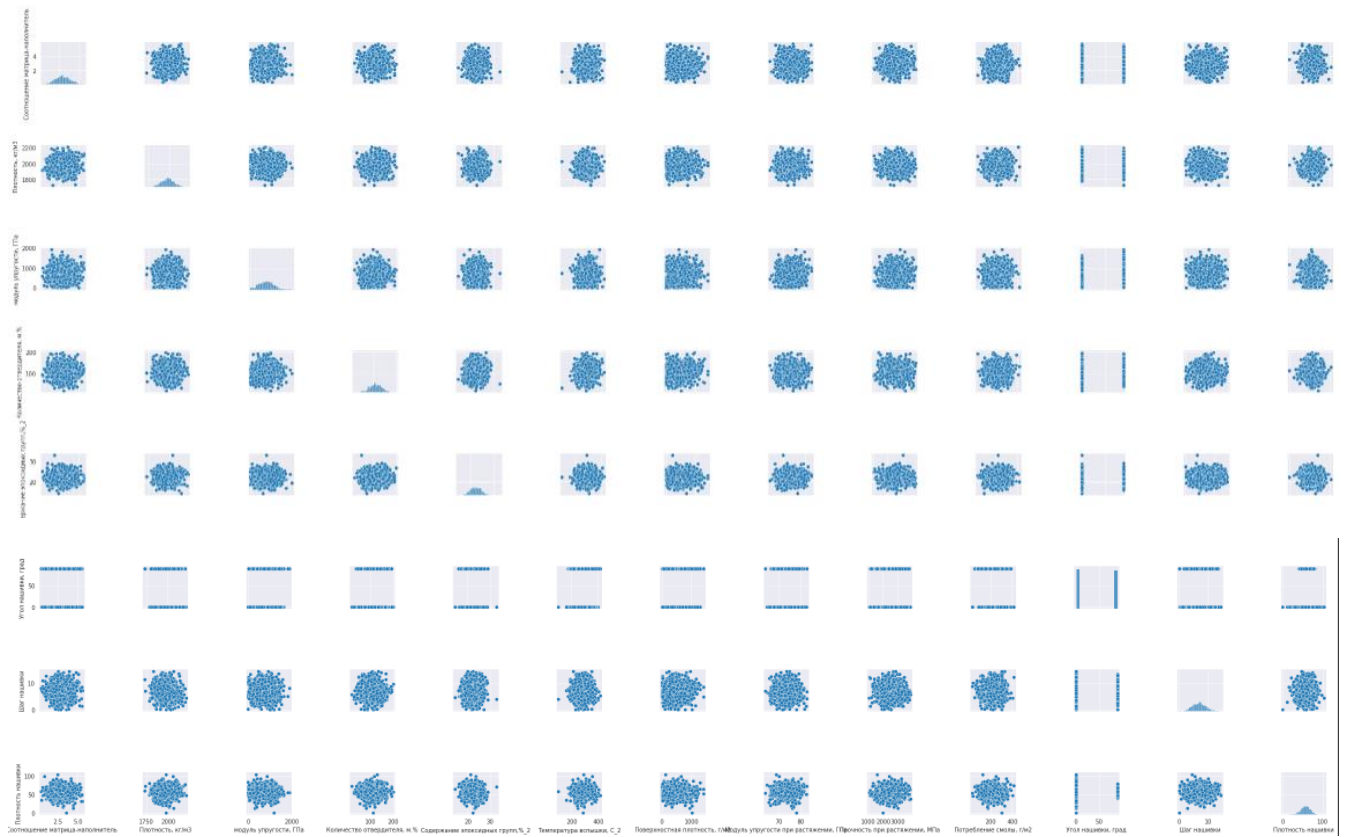


Рисунок 11 попарные графики рассеяния точек

Посмотрим также корреляцию между переменными в нашем датасете.

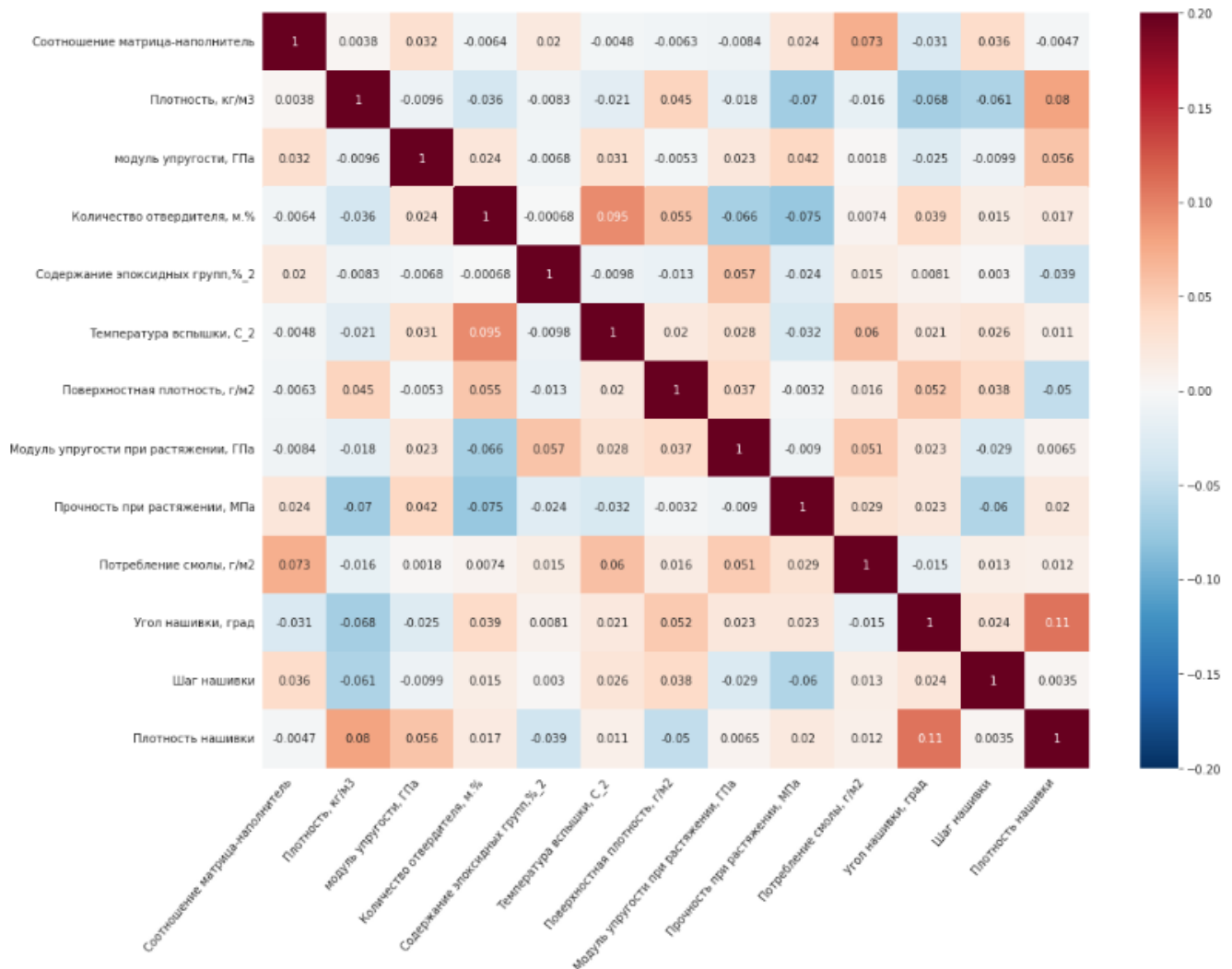


Рисунок 12 корреляция данных в датасете

Из данных корреляции можно сделать вывод, что корреляция между переменными почти не наблюдается. Максимальные показатели равны 0.11 и наблюдаются между углом и плотностью нашивки.

1.2. Предобработка данных

Так как в процессе анализа наблюдались выбросы, необходимо определить сколько их в нашем датасете. Для этого используем расчёт межквартильного диапазона и все данные за пределами «усов» превратим в нули, а далее посчитаем их сумму.

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	

Рисунок 13 количество выбросов в датасете.

Как видно из рисунка выше, количество выбросов не является существенным, их можно удалить без искажения результатов обработки.

А так же необходимо нормализовать наши данные

```
# нормализация данных
mms = MinMaxScaler()
df2 = pd.DataFrame(mms.fit_transform(df1), columns = df1.columns, index=df1.index)
df2.describe()
```

	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
count	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000
mean	0.498933	0.502695	0.446764	0.504664	0.491216	0.516059	0.373733	0.488647	0.495706	0.521141	0.511752
std	0.187489	0.187779	0.199583	0.188865	0.180620	0.190624	0.217078	0.191466	0.188915	0.195781	0.500129
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.372274	0.368517	0.301243	0.376190	0.367716	0.386128	0.205619	0.359024	0.365149	0.392067	0.000000
50%	0.494538	0.511229	0.447061	0.506040	0.489382	0.515980	0.354161	0.485754	0.491825	0.523766	1.000000
75%	0.629204	0.624999	0.580446	0.637978	0.623410	0.646450	0.538683	0.615077	0.612874	0.652447	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Рисунок 14 нормализация данных и их описание.

2. Практическая часть

2.1 Подготовка и обучение моделей

1) Задача: обучить несколько 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.

Для выполнения данной задачи нам потребуется использовать модели регрессии, обучаемые с учителем. В данной работе были использованы следующие модели:

- линейная регрессия (Linear regression);
- метод ближайших соседей (kNN - k Nearest Neighbours);
- случайный лес (RandomForest);

Каждый из методов применяется дважды. Для прогноза модуля упругости при растяжении и прочности при растяжении.

Так как при построении моделей нам потребуется провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10, импортируем библиотеку GridSearchCV и RandomizedSearchCV.

GridSearchCV исчерпывающе рассматриваются все комбинации параметров, а RandomizedSearchCV можно выбрать заданное количество кандидатов из пространства параметров с указанным распределением.

В качестве метрики для оценки моделей мы будем использовать MAPE (средняя абсолютная процентная ошибка), а также R2 (коэффициент детерминации).

По результатам обучения моделей

	Model	MAPE	R2
1	LR_uprugost	0.453354	-0.005
2	KN_uprugost	0.449671	-0.003
3	RF_uprugost	0.450596	-0.006

Рисунок 15. Оценка моделей. Модуль упругости при растяжении, ГПа.

	Model	MAPE	R2
1	LR_prochnost	0.473412	-0.047
2	KN_prochnost	0.472405	-0.028
3	RF_prochnost	0.475677	-0.034

Рисунок 15. Оценка моделей. Прочность при растяжении, МПа.

Можно сделать следующий вывод.

Средняя абсолютная процентная ошибка между значениями, предсказанными моделью, и фактическими значениями составляет 45-47%. А коэффициент детерминации ~ 0 , что говорит о том, что данные, прогнозируемые моделями равны усреднённым значениям. Такие низкие показатели работы моделей обусловлены слабой корреляцией данных.

3. Создание нейронной сети

Задача: написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель.

Для этого необходимо исключить столбец «Соотношение матрица-наполнитель» и создать обучающую и тестовую выборку.

```
# создаем обучающую и тестовую выборку
x_mn = df1.drop(['Соотношение матрица-наполнитель'], axis=1)
y_mn = df1[['Соотношение матрица-наполнитель']]

X_train_mn, X_test_mn, y_train_mn, y_test_mn = train_test_split(x_mn, y_mn, test_size=0.3, random_state=42)
```

Рисунок 16 подготовка обучающей и тестовой выборки

Выборки делим в соотношении 30% на 70%, в которых 30% оставляем на тестирование модели. Генерация чисел, а также ответ на главный вопрос жизни, вселенной и всего такого – 42.

Далее необходимо сформировать слои нашей нейронной сети

```
# формируем слои нейросети
model_mn = Sequential(X_train_mn_norm) #

model_mn.add(Dense(128)) # добавляем полно
model_mn.add(BatchNormalization()) # норма
model_mn.add(LeakyReLU()) # расширенный ак
model_mn.add(Dense(128, activation='selu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(64, activation='selu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(32, activation='selu'))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dense(16, activation='selu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(1))
model_mn.add(Activation('selu'))
```

Рисунок 17 добавление слоёв.

Из рисунка 17 видно, что мы создаем слои последовательно, начиная с плотности 128, эти слои самые быстрые, однако не слишком точные. Также мы нормализуем входные данные. И будем использовать активатор `relu`. Что позволит нейронной сети при умножении или добавлении компонентов считаться гауссовой. Так как вся сеть и ее выход на последнем уровне также нормализуются.

`LeakyReLU` используем как слой, для создания небольшого градиента, когда блок не активен.

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

Рисунок 18. Ранняя остановка работы сети.

Для того, чтобы наша нейронная сеть не работала в «холостую», мы используем раннюю остановку. Она будет отслеживать ошибки. Если нейронная сеть перестает улучшаться в течение 10 эпох, мы останавливаем её обучение.

```
model_mn.compile(  
    optimizer=tf.optimizers.SGD(learning_rate=0.02, momentum=0.5),  
    loss='mean_absolute_error')
```

Рисунок 19 компилятор

Для компиляции нашей модели, мы будем использовать стохастический оптимизатор градиентного спуска, с шагом обучения 0.02. А также зададим некоторую инерцию по методу импульсов 0.5, для того чтобы не модель не застревала на минимальных показателях.

Функция ошибки – средняя абсолютная ошибка.

```
itogo_mn = model_mn.fit(  
    X_train_mn,      # входя  
    y_train_mn,  
    batch_size = 64,  # в  
    epochs=100, # 100 эпох  
    verbose=1, # индикатор  
    validation_split = 0.2,  
    callbacks = [early_mn]  
)
```

Рисунок 20 параметры

Далее призываем нашу скомпилированную модель с оптимизатором, подключаем входящую и целевую выборки. Устанавливаем градиент для каждой 64 наблюдений. Так как у нас есть ранний стоппер, мы можем поставить любое число эпох, например 100, подключим индикатор выполнения. Долю обучающих данных которые будут использоваться в качестве данных для проверки установим на 0.2. Колбэк – в соответствии с нашим стопом.

```
Epoch 1/100  
9/9 [=====] - 2s 31ms/step - loss: 2.4620 - val_loss: 4.6872  
Epoch 2/100  
9/9 [=====] - 0s 7ms/step - loss: 1.9428 - val_loss: 4.6707  
Epoch 3/100  
9/9 [=====] - 0s 10ms/step - loss: 1.7368 - val_loss: 4.7023  
Epoch 4/100  
9/9 [=====] - 0s 6ms/step - loss: 1.6229 - val_loss: 4.6982  
Epoch 5/100  
9/9 [=====] - 0s 5ms/step - loss: 1.5067 - val_loss: 4.5422  
Epoch 6/100  
9/9 [=====] - 0s 6ms/step - loss: 1.3514 - val_loss: 3.0965  
Epoch 7/100  
9/9 [=====] - 0s 6ms/step - loss: 1.2251 - val_loss: 0.8193  
Epoch 8/100  
9/9 [=====] - 0s 5ms/step - loss: 0.9629 - val_loss: 0.8287  
Epoch 9/100  
9/9 [=====] - 0s 6ms/step - loss: 0.7757 - val_loss: 0.7748  
Epoch 10/100  
9/9 [=====] - 0s 7ms/step - loss: 0.7504 - val_loss: 0.8480  
Epoch 11/100  
9/9 [=====] - 0s 7ms/step - loss: 0.7479 - val_loss: 0.9385  
Epoch 12/100  
9/9 [=====] - 0s 17ms/step - loss: 0.7288 - val_loss: 1.0796
```

```
Epoch 19/100
9/9 [=====] - 0s 5ms/step - loss: 0.7223 - val_loss: 0.7473
Epoch 20/100
9/9 [=====] - 0s 6ms/step - loss: 0.6894 - val_loss: 0.8947
Epoch 21/100
9/9 [=====] - 0s 5ms/step - loss: 0.7046 - val_loss: 0.7508
Epoch 22/100
9/9 [=====] - 0s 9ms/step - loss: 0.6962 - val_loss: 0.8455
Epoch 23/100
9/9 [=====] - 0s 7ms/step - loss: 0.7048 - val_loss: 0.7671
Epoch 24/100
9/9 [=====] - 0s 8ms/step - loss: 0.6899 - val_loss: 0.7940
Epoch 25/100
9/9 [=====] - 0s 6ms/step - loss: 0.6786 - val_loss: 0.7647
Epoch 26/100
9/9 [=====] - 0s 9ms/step - loss: 0.6653 - val_loss: 0.9090
Epoch 27/100
9/9 [=====] - 0s 6ms/step - loss: 0.6902 - val_loss: 0.7992
Epoch 28/100
9/9 [=====] - 0s 7ms/step - loss: 0.6892 - val_loss: 0.7803
Epoch 29/100
9/9 [=====] - 0s 6ms/step - loss: 0.6778 - val_loss: 0.7954
Epoch 29: early stopping
```

Рисунок 21 работа нейронной сети

Из рисунка 21 видно, что на 29 эпохе обучение останавливается благодаря ранней остановке.

```
model_mn.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 128)	1664
batch_normalization_10 (Batch Normalization)	(None, 128)	512
leaky_re_lu_4 (LeakyReLU)	(None, 128)	0
dense_13 (Dense)	(None, 128)	16512
batch_normalization_11 (Batch Normalization)	(None, 128)	512
dense_14 (Dense)	(None, 64)	8256
batch_normalization_12 (Batch Normalization)	(None, 64)	256

```

dense_15 (Dense)                (None, 32)                2080
batch_normalization_13 (Bat    (None, 32)                128
chNormalization)

leaky_re_lu_5 (LeakyReLU)       (None, 32)                0
dense_16 (Dense)                (None, 16)                528
batch_normalization_14 (Bat    (None, 16)                64
chNormalization)

dense_17 (Dense)                (None, 1)                 17
activation_2 (Activation)       (None, 1)                 0

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

Рисунок 22 результаты обучения нашей нейронной сети

Далее визуализируем потери нашей модели.

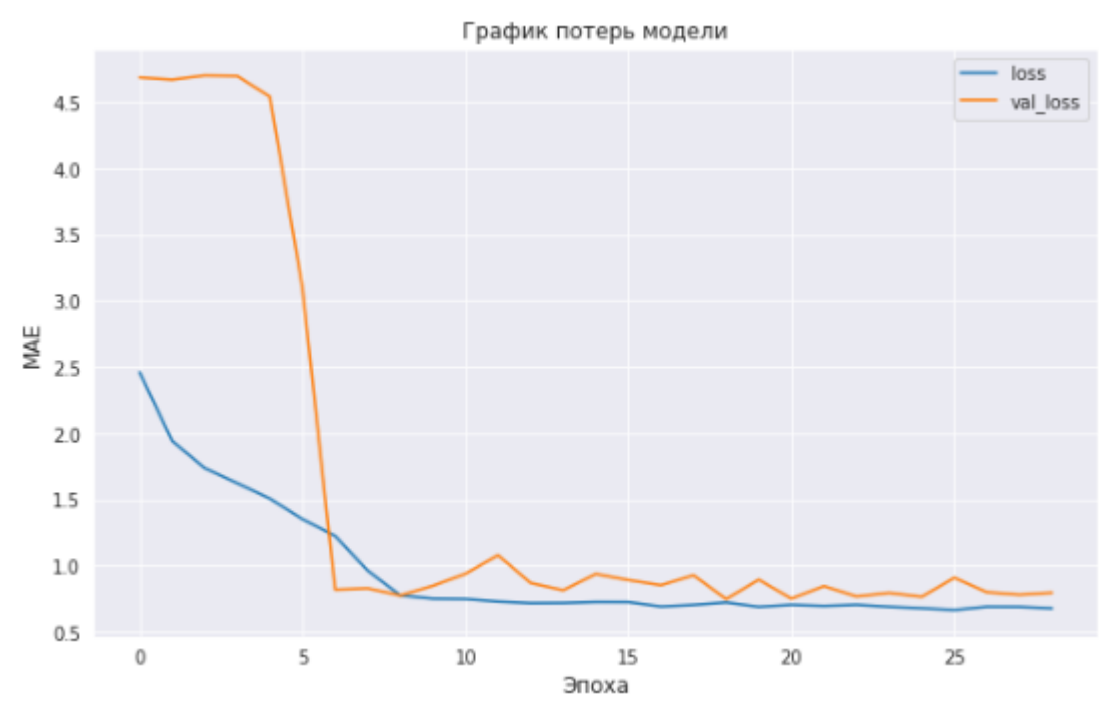


Рисунок 23 график потерь нейронной сети по эпохам

Делаем предсказание

```
pred_mn = model_mn.predict(np.array((X_test_mn)))  
original_mn = y_test_mn.values  
predicted_mn = pred_mn
```

Рисунок 24. предикт



Рисунок 25 тестовые и прогнозные значения нейронной сети

Визуализируем оригинальные и предсказанные значения Y



Рисунок 26 тестовые и прогнозные значения нейронной сети (скатерплот)

Разработка приложения

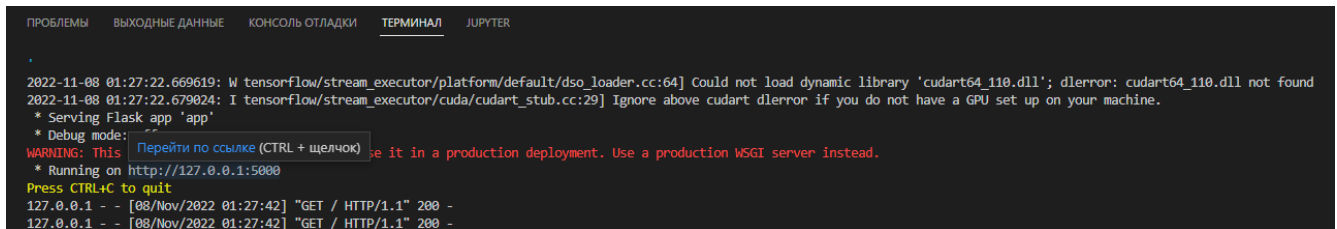
Приложение успешно работает и показывает результат прогноза для соотношения «матрица – наполнитель».

Данное приложение — это основной файл Flask, папка templates, с шаблоном html - страницы, папка model_vkr2 с сохранённой моделью для данных.

```
app.py > main
1  import flask
2  from flask import render_template
3  import tensorflow as tf
4  from tensorflow import keras
5  import sklearn
6  import keras
7
8  app = flask.Flask(__name__, template_folder='templates')
9
10 @app.route('/', methods=['GET', 'POST'])
11 @app.route('/index', methods=['GET', 'POST'])
12
13 def main():
14     temp = 1
15     param_lst = []
16
17     if flask.request.method == 'GET':
18         return render_template('main0.html' )
19
20     if flask.request.method == 'POST':
21         loaded_model = keras.models.load_model("model_vkr1")
22         for i in range(1,13,1):
23             experience = float(flask.request.form.get(f'experience{i}'))
24             param_lst.append(experience)
25
26         temp = loaded_model.predict([param_lst])
27         return render_template('main0.html', result = temp)
28
29 if __name__ == '__main__':
30     app.run()
```

Рисунок 27 - Код приложения

При запуске приложения, пользователь переходит на: <http://127.0.0.1:5000/>.



```

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  JUPYTER

2022-11-08 01:27:22.669619: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-11-08 01:27:22.679024: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [08/Nov/2022 01:27:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2022 01:27:42] "GET / HTTP/1.1" 200 -
  
```

Рисунок 28 - Ссылка для открытия html файла

В открывшемся окне пользователю необходимо ввести в соответствующие ячейки требуемые значения и нажать на кнопку «Готово».

На выходе пользователь получает результат прогноза для значения параметра «Соотношение «матрица – наполнитель»».

Введите значения:

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:

Угол нашивки, град:

Шаг нашивки:

Плотность нашивки:

Отправить

Результат прогноза:

[[-1.7580487]]

Рисунок 29 Приложение Flask

4 Создание удалённого репозитория и загрузка

Репозиторий создан на github.com по адресу:

https://github.com/KulabukhovKV/VKR_MGTU

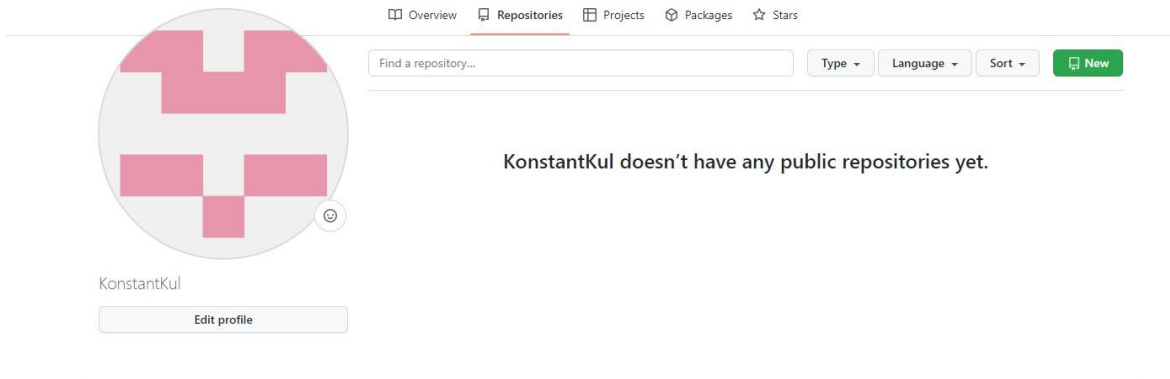


Рисунок 29, аккаунт на гитхаб

5 Заключение

Исследовательская работа, которая была проведена в рамках текущего проекта позволяет сделать некоторые выводы по теме. Распределение данных в объединённом датасете близко к нормальному, однако коэффициенты корреляции между парами признаков имеют околонулевые значения. В результате этого, построенные модели не дали каких-либо ценных результатов. Метрики показывают, что модели выдают лишь усреднённые показатели.

По результату работы нейронной сети качественное соотношение «матрица – наполнитель» определить не представляется возможным. Данный факт не указывает на то, что прогнозирование характеристик композитных материалов на основании предоставленного набора данных невозможно, но может указывать на недостатки базы данных, подходов, использованных при прогнозе, необходимости пересмотра инструментов для прогнозирования.

Требуются дополнительные вводные данные, для получения более корректных результатов и проведения более качественного исследования.

Подводя итог можно сделать вывод, что прогнозирование конечных свойств/характеристик композитных материалов без изучения материаловедения, погружения в вопрос экспериментального анализа характеристик композитных материалов не демонстрирует сколько-нибудь удовлетворительных результатов. Проработка моделей и построение прогнозов требует внедрения в процесс производных от имеющихся показателей для выявления иного уровня взаимосвязей.

3.1. Список используемой литературы и веб ресурсы.

1. Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – Режим доступа: <https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19>. (дата обращения: 07.06.2022)
2. Andre Ye. 5 алгоритмов регрессии в машинном обучении, о которых вам следует знать: – Режим доступа: <https://habr.com/ru/company/vk/blog/513842/> (дата обращения: 01.06.2022).
3. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
4. Абросимов Н.А.: Методика построения разрешающей системы уравнений динамического деформирования композитных элементов конструкций (Учебно-методическое пособие), ННГУ, 2010
5. Абу-Хасан Махмуд, Масленникова Л. Л.: Прогнозирование свойств композиционных материалов с учётом наноразмера частиц и акцепторных свойств катионов твёрдых фаз, статья 2006 год
6. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.
7. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Издательство Казанского университета, 2018. – 121 с.
8. Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
9. Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>. (дата обращения: 08.06.2022).
10. Документация по библиотеке matplotlib: – Режим досту-

па: <https://matplotlib.org/stable/users/index.html>. (дата обращения: 10.06.2022)

11. Документация по библиотеке numpy: – Режим досту-

па: <https://numpy.org/doc/1.22/user/index.html#user>. (дата обращения: 03.06.2022).

12. Документация по библиотеке pandas: – Режим досту-

па: https://pandas.pydata.org/docs/user_guide/index.html#user-guide. (дата обращения: 04.06.2022).

13. Документация по библиотеке scikit-learn: – Режим досту-

па: https://scikit-learn.org/stable/user_guide.html. (дата обращения: 05.06.2022).

14. Документация по библиотеке seaborn: – Режим досту-

па: <https://seaborn.pydata.org/tutorial.html>. (дата обращения: 06.06.2022).

15. Документация по библиотеке Tensorflow: – Режим доступа:

<https://www.tensorflow.org/overview> (дата обращения: 10.06.2022).

16. Документация по языку программирования python: – Режим досту-

па: <https://docs.python.org/3.8/index.html>. (дата обращения: 02.06.2022).

17. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.

18. Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: <https://habr.com/ru/post/428503/> (дата обращения 07.06.2022)

19. Ларин А. А., Способы оценки работоспособности изделий из композиционных материалов методом компьютерной томографии, Москва, 2013, 148 с.

20. Материалы конференции: V Всероссийская научно-техническая конференция «Полимерные композиционные материалы и производственные технологии нового поколения», 19 ноября 2021 г.

21. Миронов А.А. Машинное обучение часть I ст.9 – Режим доступа: <http://is.ifmo.ru/verification/machine-learning-mironov.pdf>. (дата обращения 08.06.2022)

22. Плас Дж. Вандер, Python для сложных задач: наука о данных и ма-

шинное обучение. Санкт-Петербург: Питер, 2018, 576 с.

23. Реутов Ю.А.: Прогнозирование свойств полимерных композиционных материалов и оценка надёжности изделий из них, Диссертация на соискание учёной степени кандидата физико-математических наук, Томск 2016.

24. Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.: Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.

25. Руководство по быстрому старту в flask: – Режим доступа: <https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html>. (дата обращения: 09.06.2022)

26. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.

27. Скиена, Стивен С. С42 Наука о данных: учебный курс.: Пер. с англ. - СПб.: ООО "Диалектика", 2020. - 544 с. : ил.

28. Справочник по композиционным материалам: в 2 - х кн. Кн. 2 / Под ред. Дж. Любина; Пер. с англ. Ф. Б. Геллера, М. М. Гельмонта; Под ред. Б. Э. Геллера - М.: Машиностроение, 1988. - 488 с. : ил;

29. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.

30. Чун-Те Чен и Грейс Х. Гу. Машинное обучение для композитных материалов (март 2019г.) – Режим доступа: <https://www.cambridge.org/core/journals/mrs-communications/article/machine-learning-for-composite-materials/F54F60AC0048291BA47E0B671733ED15>. (дата обращения 02.06.2022)