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 Лещинский Александр Викторович

Москва, 2022

Содержание

Введение

Пояснительная записка подготовлена в рамках выпускной квалификационной работы по курсу «Data Science» на тему «Прогнозирование конечных свойств новых материалов (композиционных материалов)».

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).

Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

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

В результате работы формируется комплект документов, в частности - код в Jupiter notebook, графическая визуализация, презентация для защиты, данная пояснительная записка и другие материалы.

1. Аналитическая часть

1.1 Постановка задачи

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.

Датасет со свойствами композитов расположен по ссылке https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw_SOi-vyNf2/view?usp=sharing и представлен в двух файлах формата excel:

- Файл X_nup – Накладки углепластика;
- Файл X_bp – Базальтопластик.

В файлах набор данных, состоящий из характеристик компонентов композитных материалов и характеристик производственного процесса. Общий объем dataset - 1040 измерений (строк) для каждой из 13 переменных:

- 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.

1.2 Описание используемых методов

Необходимо разработать несколько моделей для прогнозирования следующих переменных:

- Модуль упругости при растяжении, ГПа;
- Прочность при растяжении, МПа.

Перед началом решения задачи важно вспомнить теорию.

Модуль Юнга E показывает отношение нормальных напряжений к относительным деформациям в пределах пропорциональности

Модуль Юнга также определяется опытным путем при испытании стандартных образцов на растяжение. Так как нормальные напряжения в материале равны силе, деленной на начальную площадь сечения:

$$\sigma = P/F_o \quad (318.3.1), (317.2)$$

а относительное удлинение ε - отношению абсолютной деформации к начальной длине

$$\varepsilon_{пр} = \Delta l/l_o \quad (318.3.2)$$

то модуль Юнга согласно закону Гука можно выразить так

$$E = \sigma/\varepsilon_{пр} = Pl_o/F_o\Delta l = tg\alpha \quad (318.3.3)$$

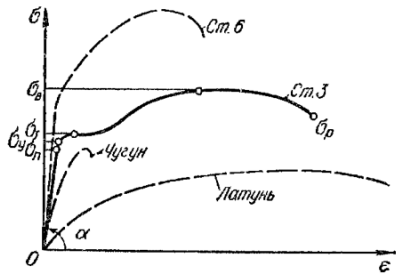
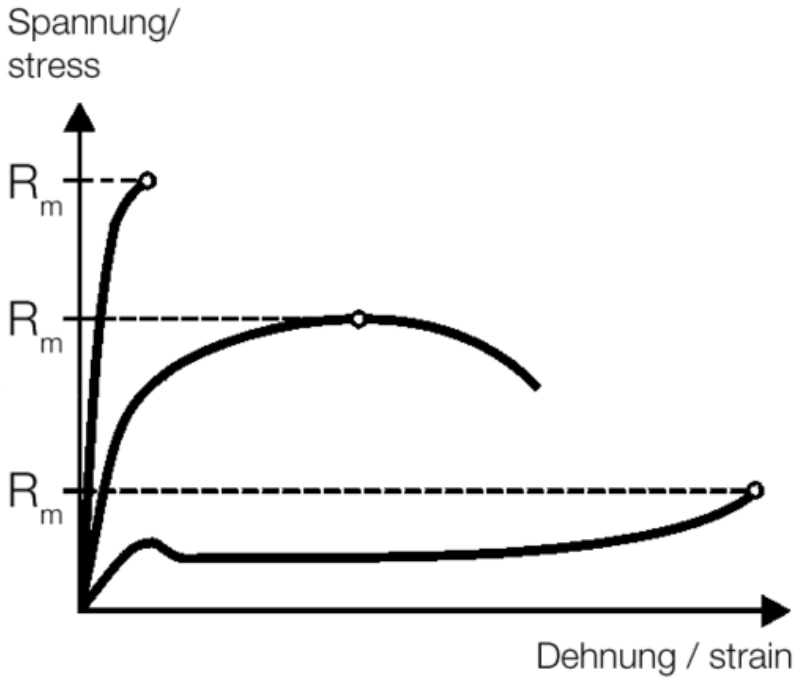


Рисунок 318.2. Диаграммы напряжений некоторых сплавов металлов

В основном, модуль упругости зависит от материала.

Прочность при растяжении Rm (такж разрывная прочность) представляет собой характеристику материала для оценки прочностных свойств. Прочность при растяжении (англ.: tensile strength) обозначает максимальное механическое растягивающее напряжение, с которым можно нагружать образец. При превышении прочности при растяжении материал разрушается: приложение усилия снижается, пока образец, наконец, не порвется. Разумеется, на образце возникает пластичная (т.е. остаточная) деформация еще до достижения прочности при растяжении.



и мы видим, что одному значению x - прочность при растяжении, могут соответствовать множество материалов в различных экспериментах

Для переменной «Соотношение матрица-наполнитель» необходимо разработать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель с использованием приложения.

Для реализации прогноза конечных свойств композитных материалов необходимо решить **задачу регрессии.**

На первом этапе за базовую модель принимается **линейная регрессия** - tf.keras.Sequential.

На следующем этапе используется м**ногослойный персептрон** с последующими экспериментами для выявления оптимальных настроек - keras.Sequential.

Линейная регрессия.

Линейная регрессия - способ выбрать из семейства функций ту, которая которая минимизирует функцию потерь. Представляет собой линейную комбинацию наперед заданных базисных функций

Регрессия — способ выбрать из семейства функций ту, которая минимизирует функцию потерь. Последняя характеризует насколько сильно пробная функция отклоняется от значений в заданных точках. Если точки получены в эксперименте, они неизбежно содержат ошибку измерений, шум, поэтому разумнее требовать, чтобы функция передавала общую тенденцию, а не точно проходила через все точки. В каком-то смысле регрессия — это «интерполирующая аппроксимация»: мы хотим провести кривую как можно ближе к точкам и при этом сохранить ее максимально простой чтобы уловить общую тенденцию. За баланс между этими противоречивыми желаниями как-раз отвечает функция потерь (в английской литературе «loss function» или «cost function»).

Цель регрессии — найти коэффициенты этой линейной комбинации, и тем самым определить регрессионную функцию (которую также называют *моделью*). Отметим, что линейную регрессию называют линейной именно из-за линейной комбинации базисных функций — это не связано с самыми базисными функциями (они могут быть линейными или нет).

Регрессия с нами уже давно: впервые метод опубликовал Лежандр в 1805 году, хотя Гаусс пришел к нему раньше и успешно использовал для предсказания орбиты «кометы» (на самом деле карликовой планеты) Цереры.

Существует множество вариантов и обобщений линейной регрессии: LAD, метод наименьших квадратов, Ridge регрессия, Lasso регрессия, ElasticNet и многие другие.

Многослойный перцептрон (нейронные сети)

Перцептрон принимает обратную логит (логистическую) функцию от wx и не использует вероятностные предположения ни для модели, ни для ее параметра. Онлайновое обучение даст вам точно такие же оценки для весов / параметров модели, но вы не сможете интерпретировать их в причинно-следственной связи из-за отсутствия p-значений, доверительных интервалов и, следовательно, базовой вероятностной модели.

Однослойный персептрон (англ. *Single-layer perceptron*) — перцептрон, каждый S-элемент которого однозначно соответствует одному A-элементу, S-A связи всегда имеют вес *1*, а порог любого A-элемента равен *1*. Часть однослойного персептрона соответствует модели искусственного нейрона.

Его уникальность состоит в том, что каждый S-элемент однозначно соответствует одному A-элементу, все S-A связи имеют вес, равный *+1*, а порог A элементов равен *1*. Часть однослойного перцептрона, не содержащая входы, соответствует искусственному нейрону, как показано на картинке. Таким образом, однослойный перцептрон — это искусственный нейрон, который на вход принимает только *0* и *1*.

Однослойный персептрон также может быть и элементарным персептроном, у которого только по одному слою S,A,R-элементов.

Задача обучения перцептрона — подобрать такие $w_0, w_1, w_2, \ldots, w_n$, чтобы $sign(\sigma(w_0 + w_1x_{1+}w_2x_{2+\ldots+}w_nx_n))$ как можно чаще совпадал с *y(x)*— значением в обучающей выборке (здесь σ — функция активации). Для удобства, чтобы не тащить за собой свободный член w_0, добавим в вектор x лишнюю «виртуальную размерность» и будем считать, что $x = (1, x_1, x_2, \ldots, x_n)$. Тогда $w_0 + w_1x_{1+}w_2x_{2+\ldots+}w_nx_n$ можно заменить на $w^T x$.

Чтобы обучить эту функцию, сначала выбирается функция ошибки, которую затем оптимизируется градиентным спуском. Число неверно

классифицированных примеров здесь использовать неуместно, потому что эта функция кусочно-гладкая, с массой разрывов: она будет принимать только целые значения и резко меняться при переходе от одного числа неверно классифицированных примеров к другому. Поэтому следует использовать другую функцию, так называемый *критерий перцептрона*:

$$E_p(w) = -\sum_{x \in M} y(x)(\sigma(w^T x))$$

где M — множество примеров, которые перцептрон с весами w классифицирует неправильно.

Иначе говоря, надо минимизировать суммарное отклонение наших ответов от правильных, но только в неправильную сторону; верный ответ ничего не вносит в функцию ошибки. Умножение на $y(x)$ здесь нужно для того, чтобы знак произведения всегда получался отрицательным: если правильный ответ -1, значит, перцептрон выдал положительное число (иначе бы ответ был верным), и наоборот. В результате получается кусочно-линейная функция, дифференцируемая почти везде, а этого вполне достаточно.

Теперь $E_p(w)$ можно оптимизировать градиентным спуском. На очередном шаге получаем:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla_w E_p(w)$$

Алгоритм такой — мы последовательно проходим примеры $x_1, x_2, \ldots$ из обучающего множества, и для каждого x_n:

- если он классифицирован правильно, не меняем ничего;
- а если неправильно, прибавляем $\eta \nabla_w E_p(w)$.

Ошибка на примере x_n при этом, очевидно, уменьшается, но, конечно, совершенно никто не гарантирует, что вместе с тем не увеличится ошибка от других примеров. Это правило обновления весов так и называется — правило обучения перцептрона, и это было основной математической идеей работы Розенблатта.

1.3 Разведочный анализ данных

Для первоначального анализа используются методы разведочного анализа данных.

df.info()

#проверка на пропуски

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1040 entries, 0 to 1039
Data columns (total 13 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Угол нашивки, град                     1040 non-null   int64
 1   Шаг нашивки                            1040 non-null   float64
 2   Плотность нашивки                      1040 non-null   float64
 3   Соотношение матрица-наполнитель        1023 non-null   float64
 4   Плотность, кг/м3                       1023 non-null   float64
 5   модуль упругости, ГПа                  1023 non-null   float64
 6   Количество отвердителя, м.%            1023 non-null   float64
 7   Содержание эпоксидных групп,%_2        1023 non-null   float64
 8   Температура вспышки, С_2               1023 non-null   float64
 9   Поверхностная плотность, г/м2          1023 non-null   float64
 10  Модуль упругости при растяжении, ГПа   1023 non-null   float64
 11  Прочность при растяжении, МПа          1023 non-null   float64
 12  Потребление смолы, г/м2                1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 105.8 KB
```

df.duplicated().sum()

#проверка на дубликаты

df.describe()

 # описательная статистика

for col in df.columns:

plt.title("Распределение")

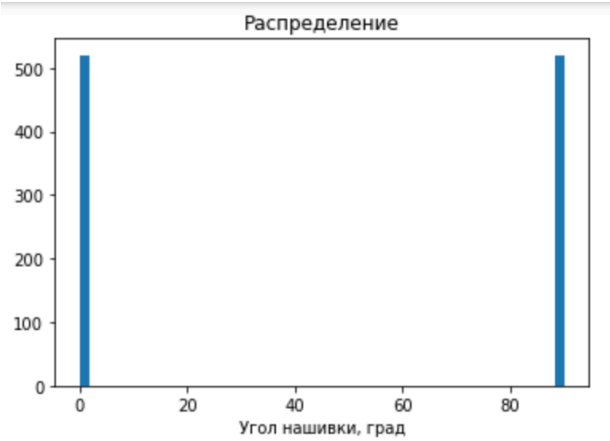
plt.xlabel(col)

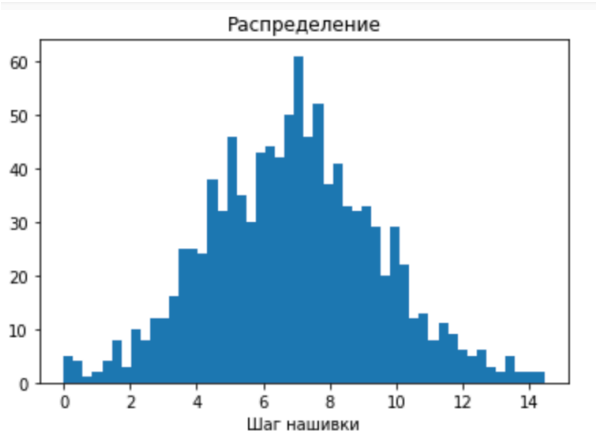
plt.ylabel("")

plt.hist(df[col], bins = 50)

plt.show()

#гистограммы распределения





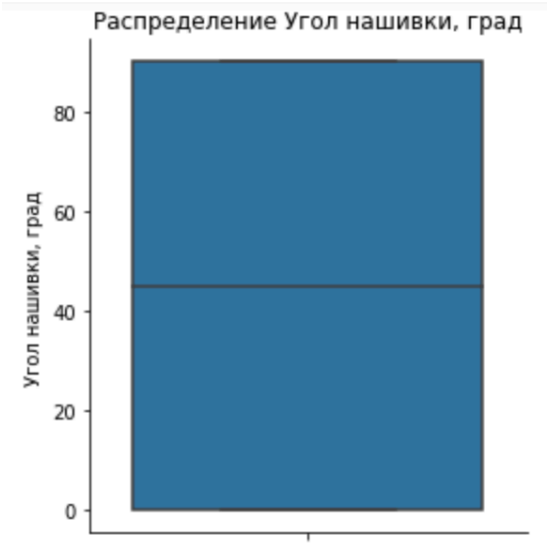
В целом по всем параметрам данные распределены нормально (как в примере с Шагом нашивки), только с Углом нашивки наблюдается анормальная картина.

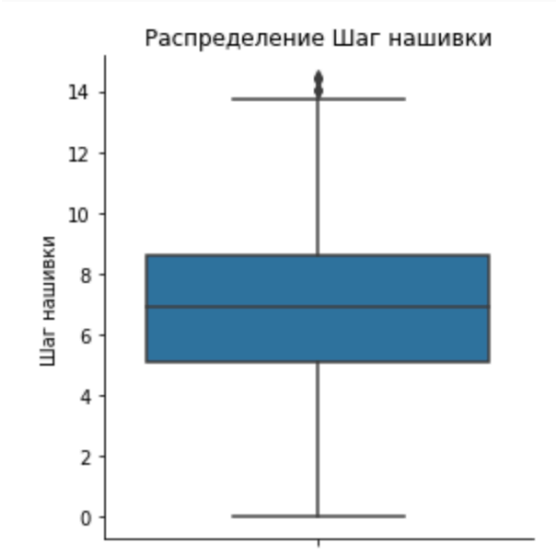
for col in df.columns:

sns.catplot(y = col, data = df, kind = 'box', height=4)

plt.title("Распределение " + str(col))

#диаграммы ящика с усами





В целом по всем параметрам данные распределены нормально (как в примере с Шагом нашивки), только с Углом нашивки наблюдается анормальная картина.

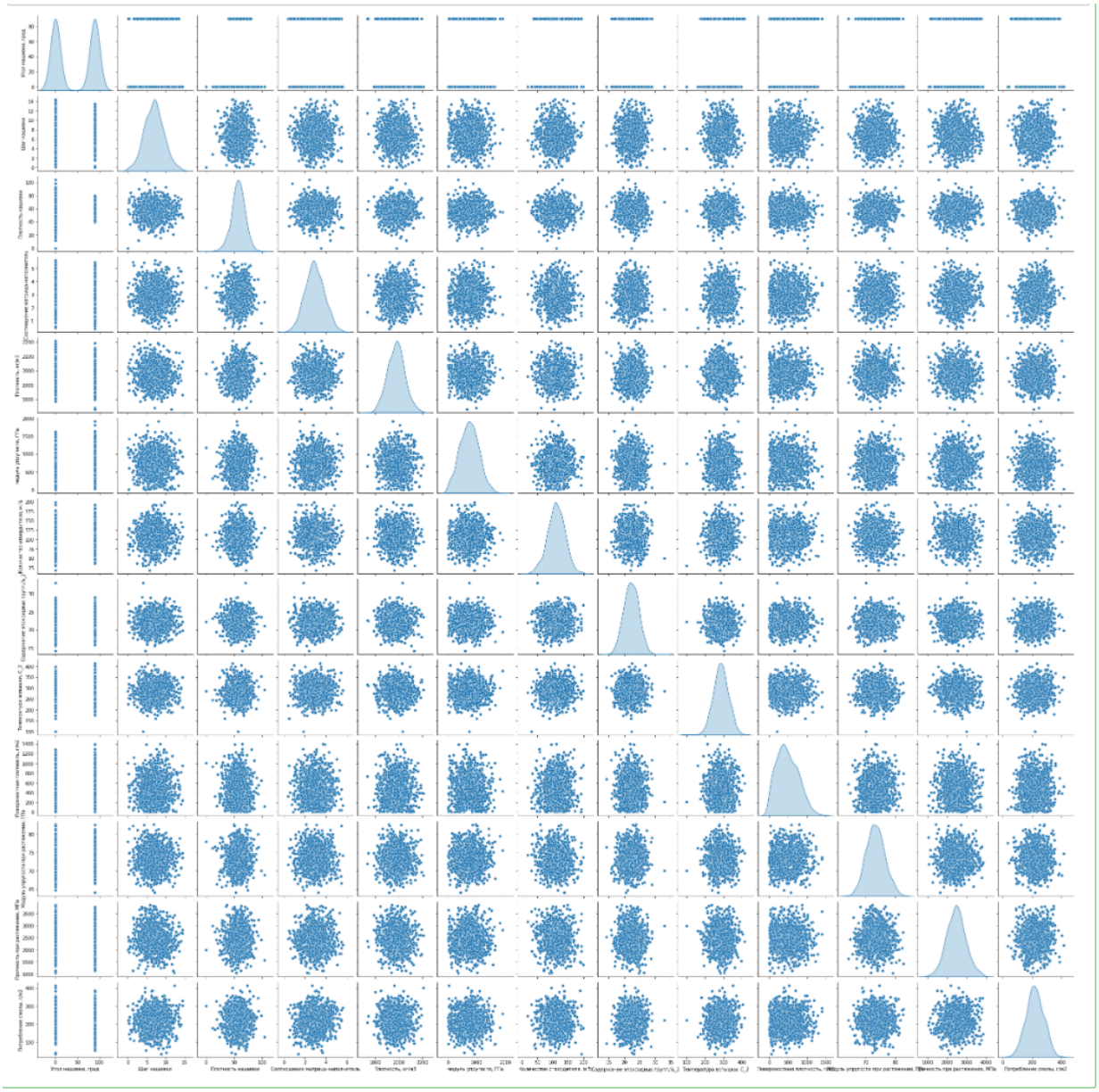
df.describe().transpose()[['mean', '50%']]

 # среднее медианное значение

Выбросов нет. все нормально

```
sns.pairplot(df[df.columns],
    diag_kind='kde');
```

 # попарные графики



Получилось не наглядно.

```
corr = df.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
# Создаем полотно для отображения графика
f, ax = plt.subplots(figsize = (12, 10))
# Создаем цветовую политру
cmap = sns.diverging_palette(170, 200, as_cmap=True)
```

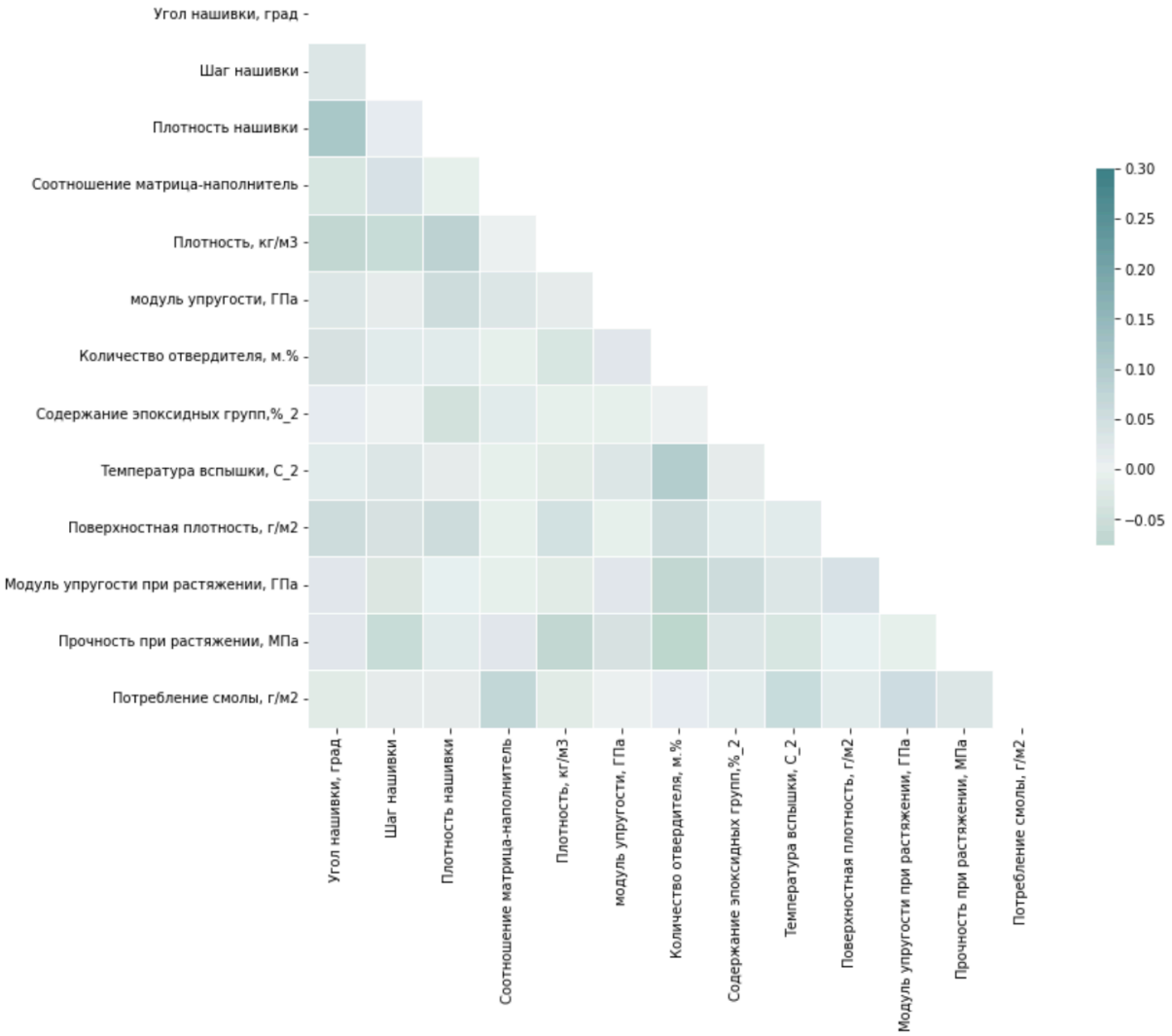
 # Визуализируем данные корреляции

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar_kws={"shrin

Модель	dnn_model2	dnn_model	linear_model
	keras.Sequential	keras.Sequential	tf.keras.Sequential
loss	MAE	MAE	MAE
activation	sigmoid	relu	
layers.Dense	200/20/10/2	40/40/2	2
optimizer	RMSprop	Adam	
layers.Dropout	0.6		

*k": .5})****#проверим корреляции признаков***



Можно отметить пары количество отвердителя - температура вспышки, плотность нашивки - угол нашивки.

2. Практическая часть

```
# сделаем копию, чтобы изменения не влияли на оригинальный dataset
df2 = copy.copy(df)
df2.head()
```

2.1 Предобработка данных

В процессе предобработки данных производится их подготовка к анализу, в результате которой они приводятся в соответствие с требованиями, определяемыми спецификой решаемой задачи.

Предобработка является важнейшим этапом Data Mining, и если она не будет выполнена, то дальнейший анализ в большинстве случаев невозможен из-за того, что аналитические алгоритмы просто не смогут работать или результаты их работы будут некорректными. Иными словами, реализуется принцип GIGO — garbage in, garbage out (мусор на входе, мусор на выходе).

Предобработка данных включает три направления: удаление, нормализацию и стандартизацию.

df.isnull().sum()

#проверим какое количество выбросов по каждому столбцу

```
Угол нашивки, град                    0
Шаг нашивки                           0
Плотность нашивки                     0
Соотношение матрица-наполнитель      17
Плотность, кг/м3                     17
модуль упругости, ГПа                17
Количество отвердителя, м.%          17
Содержание эпоксидных групп,%_2      17
Температура вспышки, С_2             17
Поверхностная плотность, г/м2        17
Модуль упругости при растяжении, ГПа 17
Прочность при растяжении, МПа        17
Потребление смолы, г/м2              17
dtype: int64
```

Учитывая, что количество выбросов довольно мало, можно просто удалить эти строки.

Удаление пропусков произведено с использованием

df = df.dropna()

#удаление пропусков

df2.isna().sum()

 # теперь пропусков нет

```
Угол нашивки, град                        0
Шаг нашивки                               0
Плотность нашивки                         0
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
dtype: int64
```

min_max_scaler = preprocessing.MinMaxScaler()

norm_df = pd.DataFrame(min_max_scaler.fit_transform(df2),

columns = df2.columns,

index = df2.index)

norm_df

#нормализуем значения

norm_df.describe()

	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	Моду упругос п растяжени П
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.0000
mean	0.491691	0.477768	0.549616	0.488427	0.512533	0.386301	0.512273	0.426215	0.593354	0.344638	0.4978
std	0.500175	0.177519	0.118772	0.175541	0.154890	0.172978	0.156147	0.128370	0.130695	0.201092	0.1674
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	0.351790	0.478890	0.370696	0.404175	0.260652	0.412240	0.338919	0.507756	0.190296	0.3860
50%	0.000000	0.478940	0.551423	0.483912	0.516497	0.386165	0.512240	0.425487	0.593401	0.322574	0.4946
75%	1.000000	0.594597	0.624537	0.608045	0.608413	0.502528	0.618003	0.517842	0.679924	0.495105	0.6067
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

#нормализация выполнена

```
s_scaler = preprocessing.StandardScaler()
standart_df = pd.DataFrame(s_scaler.fit_transform(df2),
   columns = df2.columns,
   index = df2.index)
standart_df
```

#стандартизируем значения

	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	Пр расті
0	-0.983518	-1.131530	-0.012469	-1.175779	0.736365	-0.003594	-2.848828	0.009757	-4.542215	-0.969964	-1.067720	1
1	-0.983518	-1.131530	0.230546	-1.175779	0.736365	-0.003594	-2.141666	0.626001	-0.030955	-0.969964	-1.067720	1
2	-0.983518	-1.131530	1.040595	-1.175779	0.736365	-0.003594	-2.145202	4.471955	-0.030955	-0.969964	-1.067720	1
3	-0.983518	-0.741243	-0.822518	-1.175779	0.736365	-0.003594	0.651622	-0.413446	0.344984	-0.969964	-1.067720	1
4	-0.983518	-0.741243	-0.012469	-0.174232	0.736365	0.039618	0.045585	0.009757	-0.030955	-0.969964	-1.067720	1
...	...	...	...	...	...	...	...	...	...	...	...	
1018	1.016758	0.849718	-0.820917	-0.721995	-0.320884	0.523926	-0.833694	-0.881925	0.950375	-0.972814	-0.076219	-(
1019	1.016758	1.430947	-0.275671	0.562741	1.008971	-0.894327	1.252072	-1.099577	-0.773808	-0.469707	-0.130794	-(
1020	1.016758	-1.068634	0.848588	0.383707	-0.045622	-0.978843	-0.001319	0.712276	-0.915350	0.915476	0.450935	(
1021	1.016758	-0.228717	0.089684	0.849043	1.235730	0.004703	1.089990	-1.246274	-0.246860	0.564542	0.229077	-(
1022	1.016758	-0.320161	1.642823	0.961522	-1.157793	-0.977389	0.658107	2.174679	0.368264	0.981645	0.314722	(

1023 rows × 13 columns

2.2. Разработка и обучение модели

В данной части приводится список моделей, которые будут использоваться для прогноза модуля упругости при растяжении и прочности при растяжении.

Для прогноза модуля упругости при растяжении и прочности при растяжении данные были разделены в соотношении:

- 70% обучение модели;
- 30% тестирование модели.

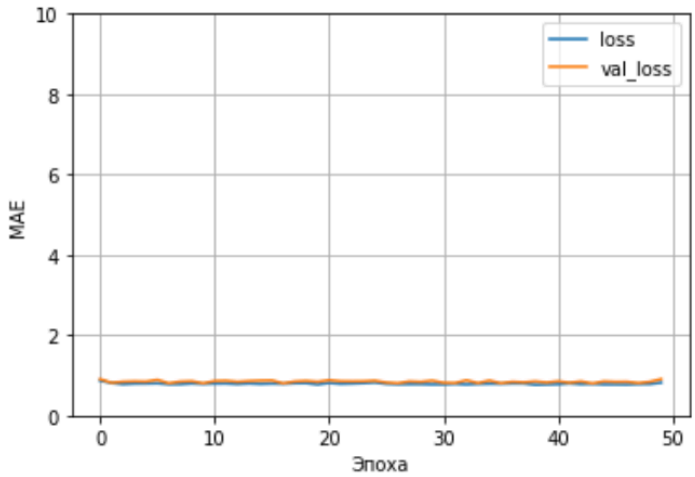
Были использованы следующие модели:

- Модель 1 - линейная регрессия leaner_model;
- Модель 2- многослойный персептрон dnn_model;
- Модель 3 – проведены эксперименты с многослойным персептроном dnn_model2.

Модель 1 - линейная регрессия leaner_model

```
plot_loss(history)
linear_model.evaluate(X_test, y_test, verbose=0)
```

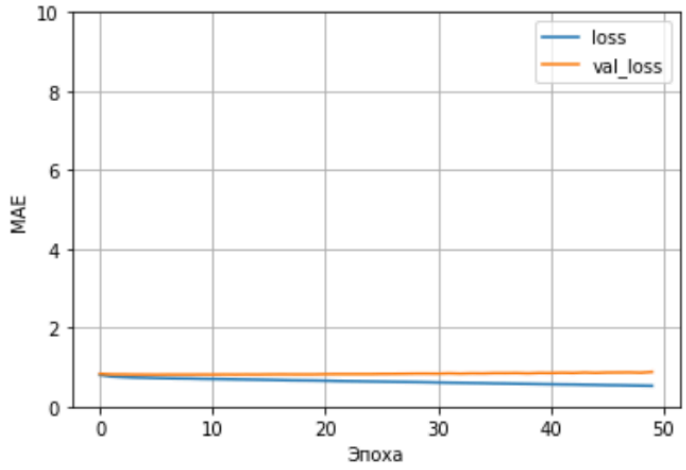
```
0.941592812538147
```



Модель 2 - многослойный персептрон 2 dnn_model

```
%%time
history = dnn_model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    verbose=0, epochs=50)
plot_loss(history)
dnn_model.evaluate(X_test, y_test, verbose=0)
```

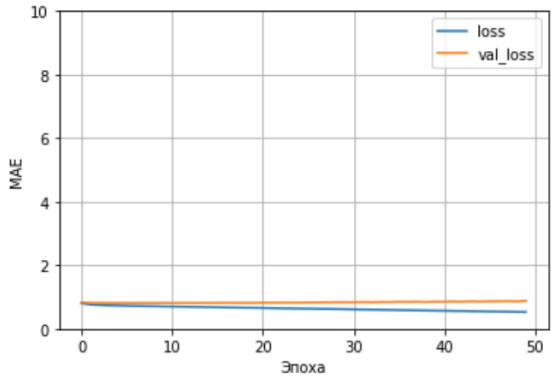
```
CPU times: user 3.98 s, sys: 696 ms, total: 4.67 s
Wall time: 4.33 s

0.9466024041175842
```



Модель 3 – измененный многослойный персептрон dnn_model2.

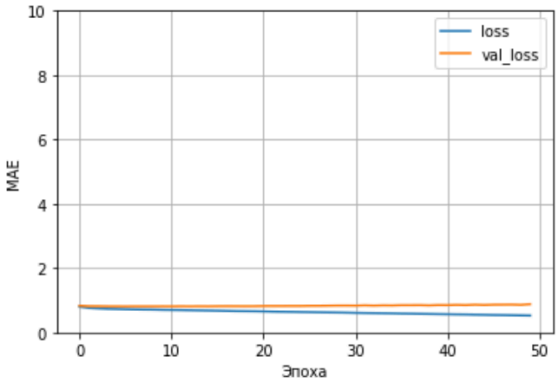
```
%%time
history2 = dnn_model2.fit(
    X_train2,
    y_train2,
    validation_split = 0.1,#изменили на 0.1
    verbose=1, epochs=70)#добавили 20 эпох
plot_loss(history)
dnn_model2.evaluate(X_test2, y_test2, verbose=0)
```

```
Wall time: 9.88 s

0.15084879100322723
```




```
%%time
history2 = dnn_model2.fit(
    X_train2,
    y_train2,
    validation_split = 0.3,
    verbose=1, epochs=50)
plot_loss(history)
dnn_model2.evaluate(X_test2, y_test2, verbose=0)
```

```
Wall time: 7.94 s

0.15041205286979675
```



2.3. Тестирование модели

Для каждой модели были определены средние абсолютные ошибки на тренировочной и тестирующей части выборке. Тестовая часть одинаково преобразована для всех моделей (удалены пропуски и проведена нормализация).

```
pd.DataFrame(test_results, index=['Mean absolute error']).T
```

	Mean absolute error
linear_model	0.941593
dnn_model	0.946602
dnn_model2	0.150412

Для модели dnn_model2 выявлена наименьшая абсолютная средняя ошибка.

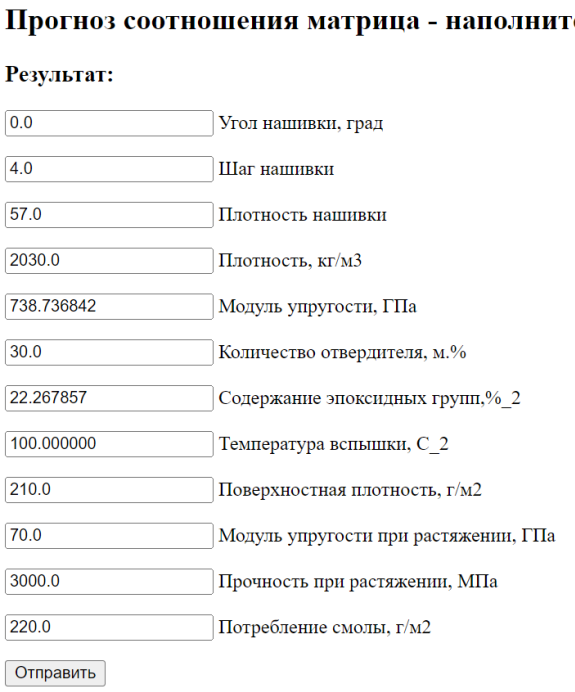
2.4. Написать нейронную сеть, которая будет рекомендовать соотношение матрица.

Для прогнозирования соотношения матрица – наполнитель была выбрана модель dnn_model2. Активационная функция sigmoid, оптимизатор RMSprop, функция потерь MAE, 3 скрытых слоя определены экспериментальным путем, на выходе 1 нейрон.

2.5. Разработка приложения

Для отображения результата прогнозирования пользователю необходимо ввести известные характеристики (тип float) и нажать кнопку «Отправить», ответ будет выведен в строке «Результат».

Работа приложения показана на рисунках ниже.



2.6.Создание удаленного репозитория и загрузка результатов работы на него.

Создан репозиторий в GitHub, где размещен код исследования, оформлен файл README.

Страница: https://github.com/AlexLeshchinskiy/GraduationWork

Репозиторий: GraduationWork

Коммиты в репозитории:

1. Презентация Выпускная квалификационная работа по курсу «Data Science» (файлы VKR_presentation.pdf и .pptx)

2. Jupiter ноутбук в котором проводилось исследование задачи (Файл VKR_Notebook.ipynb)

3. Выпускная квалификационная работа по курсу «Data Science» (Файл VKR_LeshchinskiyAleksander.docx и.pdf)

4. Модель (Файл «VKR_keras_model.h5»)

5. Jupiter ноутбук в котором запускалось Приложение на Flask (Файл «VKR_apps.ipynb»)

3. Заключение

В результате работы было проведено исследования с использованием методов, изученных на курсе «Data Science». и построены несколько прогнозных моделей. Результаты проекта сформированы.

Описанные выше подходы возможно использовать только как обучающие, для промышленного использования не подходят.

4. Библиографический список

1. Шакирьянов Э. Д. Ш17 Компьютерное зрение на Python. Первые шаги / Э. Д. Шакирьянов. -- Электрон. изд. М.: Лаборатория знаний, 2021. 163 с. (Школа юного инженера). Систем. требования: Adobe Reader XI; экран 10". Загл. с титул. экрана. Текст: электронный.
2. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.
3. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
4. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.

5. Уилан, Чарльз. Голая статистика. Самая интересная книга о самой скучной науке / Чарльз Уилан; пер. с англ. И. Веригина; [науч. ред. А. Минько]. — М.: Манн, Иванов и Фербер, 2016. — 352 с.
6. Джоэл Грас. – Data Sience. Наука о данных с нуля: Пер. с англ. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2022 – 416с.: ил.
7. Д.А. Иванов А.И., Ситников, С.Д Шляпин – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.
8. А.А. Миронов. Машинное обучение часть I ст.9
9. 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.
10. Язык программирования Python- Режим доступа: https://www.python.org/.
11. Среда разработки Jupyter Notebook- Режим доступа: https://jupyter.org/.
12. Библиотека Tensorflow: Режим доступа: https://www.tensorflow.org/.
13. Документация по библиотеке numpy: – Режим доступа: https://numpy.org/doc/1.22/user/index.html#user
14. Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html#user-guide
15. Документация по библиотеке matplotlib: – Режим доступа: https://matplotlib.org/stable/users/index.html
16. Документация по библиотеке seaborn: – Режим доступа: https://seaborn.pydata.org/tutorial.html
17. Документация по библиотеке sklearn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html
18. Документация по библиотеке keras: – Режим доступа: https://keras.io/api/
19. Руководство по быстрому старту в flask: – Режим доступа: https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html.

5. Приложения

1. Презентация Выпускная квалификационная работа по курсу «Data Science» (файлы VKR_presentation.pdf и .pptx)

2. Jupiter ноутбук в котором проводилось исследование задачи (Файл VKR_Notebook.ipynb)

3. Выпускная квалификационная работа по курсу «Data Science» (Файл VKR_LeshchinskiyAleksander.docx и.pdf)

4. Модель (Файл «VKR_keras_model.h5»)

5. Jupiter ноутбук в котором запускалось Приложение на Flask (Файл «VKR_apps.ipynb»)