

**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

Тулупов Роман Иванович (ФИО)

Москва, 2023

Содержание

Содержание	2
Введение	3
Основная часть.....	4
1. Аналитическая часть	4
1.1. Постановка задачи.....	4
1.2. Описание используемых методов.....	5
1.3. Разведочный анализ данных	10
2. Практическая часть	14
2.1. Предобработка данных	14
2.2. Разработка и обучение модели	17
2.3. Тестирование модели.....	19
2.4. Написать нейронную сеть, которая будет рекомендовать соотношение матрица - наполнитель.	19
2.5. Разработка приложения	20
2.6. Создание удаленного репозитория и загрузка результатов работы на него.	22
Заключение.....	23
Список литературы и веб ресурсы.....	24

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.

Основная часть

1. Аналитическая часть

1.1. Постановка задачи.

Для сокращения количества физических испытаний, а также для пополнения базы данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов, требуется разработать приложение для предсказания свойств материалов на основе предложенного датасета. Имеется два файла X_br.xlsx - с информацией о параметрах, состоящий из 1023 строк и 11 колонок и X_nup.xlsx - с информацией о нашивке, состоящий из 1040 строк и 4 колонок. Файл X_br.xlsx содержит колонку с индексами, колонки с начальными параметрами:

«Соотношение матрица-наполнитель»;

«Количество отвердителя, м.>%»;

«Содержание эпоксидных групп,%_2»;

«Температура вспышки, С_2»;

«Потребление смолы, г/м2».

и колонки с характеристиками полученного материала:

«Плотность, кг/м3»;

«модуль упругости, ГПа»;

«Поверхностная плотность, г/м2»;

«Модуль упругости при растяжении, ГПа»;

«Прочность при растяжении, МПа».

Файл X_nur.xlsx содержит колонку с индексами и колонки с информацией о нашивке:

«Угол нашивки, град»;

«Шаг нашивки»;

«Плотность нашивки».

Файлы объединяются по индексу в датасет «full_df» тип объединения INNER. Объем полученной выборки составляет 1023 строки и 13 признаков, из них 8 являются входными переменными и 5 выходными переменными. В данном задании колонки «Соотношение матрица-наполнитель», «Модуль упругости при растяжении, ГПа», «Прочность при растяжении, МПа» являются целевыми.

```
full_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп, %_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, C_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Пропусков нет, разброс данных по всем колонкам достигает 5 порядков, отрицательных значений нет. «Шаг нашивки» принимает всего два значения 0 и 90. Медиана и средняя близки по значению для всех колонок кроме поверхностной плотности, где распределение, видимо, немного смещено.

1.2. Описание используемых методов

Данная задача является задачей регрессии, буду использовать библиотеку sklearn и следующие модели:

- Линейная регрессия `LinearRegression()`;
- Метод опорных векторов `SVR()`;
- Случайный лес `RandomForestRegressor()`;
- Градиентный бустинг `GradientBoostingRegressor()`;
- Лассо регрессор `Lasso()`.

Линейная регрессия – один из самых простых инструментов статистического моделирования, но именно в его простоте и заключается его эффективность. Прогнозируемое значение целевой переменной выражается в виде суммы постоянного смещения и взвешенной суммы исходных переменных.

Достоинства метода: быстр и прост в реализации; легко интерпретируем; имеет меньшую сложность по сравнению с другими алгоритмами.

Недостатки метода: моделирует только прямые линейные зависимости; требует прямую связь между зависимыми и независимыми переменными; выбросы оказывают огромное влияние, а границы линейны.

В качестве метрики прежде всего используется R^2 или коэффициент детерминации, который позволяет измерить, насколько модель может объяснить дисперсию данных. Если R-квадрат равен 1, это значит, что модель описывает все данные. Если же R-квадрат равен 0.5, модель объясняет лишь 50% дисперсии данных. Оставшиеся отклонения не имеют объяснения. Чем ближе R^2 к единице, тем лучше.

Метод опорных векторов – это бинарный линейный классификатор. Хорошо работает на небольших датасетах. Алгоритм помечает каждый объект, как принадлежащий к одной из двух категорий, строит модель, которая определяет новые наблюдения в одну из категорий. Модель метода опорных векторов – отображение данных точками в пространстве, так что между наблюдениями отдельных категорий имеется разрыв. Каждый объект данных представляется

как вектор (точка) в p -мерном пространстве. Он создаёт линию или гиперплоскость, которая разделяет данные на классы.

Достоинства метода: для классификации достаточно небольшого набора данных; при правильной работе модели, построенной на тестовом множестве, вполне возможно применение данного метода на реальных данных; эффективен при большом количестве гиперпараметров; способен обрабатывать случаи, когда гиперпараметров больше, чем количество наблюдений; существует возможность гибко настраивать разделяющую функцию; алгоритм максимизирует разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации.

Недостатки метода: неустойчивость к шуму; для больших наборов данных требуется долгое время обучения; параметры модели сложно интерпретировать.

Случайный лес — универсальный алгоритм машинного обучения, суть которого состоит в использовании ансамбля решающих деревьев. Само по себе решающее дерево предоставляет крайне невысокое качество классификации, но из-за большого их количества результат значительно улучшается. Также это один из немногих алгоритмов, который можно использовать в абсолютном большинстве задач.

Достоинства метода: имеет высокую точность предсказания, которая сравнима с результатами градиентного бустинга; не требует тщательной настройки параметров, хорошо работает из коробки; практически не чувствителен к выбросам в данных из-за случайного семплирования (random sample); не чувствителен к масштабированию и к другим монотонным преобразованиям значений признаков; редко переобучается; способен эффективно обрабатывать данные с большим числом признаков и классов; хорошо работает с пропущенными данными; одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки.

Недостатки метода: для реализации алгоритма случайного дерева требуется значительный объем вычислительных ресурсов; большой размер моделей; построение случайного леса отнимает больше времени, чем деревья решений или линейные алгоритмы; алгоритм склонен к переобучению на зашумленных данных; нет формальных выводов, которые используются для оценки важности переменных; в отличие от более простых алгоритмов, результаты случайного леса сложнее интерпретировать; когда в выборке очень много разреженных признаков, алгоритм работает хуже, чем линейные методы; в отличие от линейной регрессии, Random Forest не обладает возможностью экстраполяции; это можно считать и плюсом, так как в случае выбросов не будет экстремальных значений; если данные содержат группы признаков с корреляцией, которые имеют схожую значимость для меток, то предпочтение отдается небольшим группам перед большими, что ведет к недообучению; процесс прогнозирования с использованием случайных лесов очень трудоемкий по сравнению с другими алгоритмами.

Градиентный бустинг — это ансамбль деревьев решений, обученный с использованием градиентного бустинга. В основе данного алгоритма лежит итеративное обучение деревьев решений с целью минимизировать функцию потерь. Основная идея градиентного бустинга: строятся последовательно несколько базовых классификаторов, каждый из которых как можно лучше компенсирует недостатки предыдущих. Финальный классификатор является линейной композицией этих базовых классификаторов.

Достоинства метода: новые алгоритмы учатся на ошибках предыдущих; требуется меньше итераций, чтобы приблизиться к фактическим прогнозам; наблюдения выбираются на основе ошибки; прост в настройке темпа обучения и применения; легко интерпретируем.

Недостатки метода: необходимо тщательно выбирать критерии остановки, иначе это может привести к переобучению; наблюдения с наибольшей ошибкой появляются чаще; слабее и менее гибок чем нейронные сети.

Лассо регрессор — это линейная модель, которая оценивает разреженные коэффициенты. Это простой метод, позволяющий уменьшить сложность модели и предотвратить переопределение, которое может возникнуть в результате простой линейной регрессии. Данный метод вводит дополнительное слагаемое регуляризации в оптимизацию модели. Это даёт более устойчивое решение. В регрессии лассо добавляется условие смещения в функцию оптимизации для того, чтобы уменьшить коллинеарность и, следовательно, дисперсию модели. Но вместо квадратичного смещения, используется смещение абсолютного значения. Лассо регрессия хорошо прогнозирует модели временных рядов на основе регрессии, таким как авторегрессии.

Достоинства метода: легко полностью избавляется от шумов в данных; быстро работает; не очень энергоёмко; способно полностью убрать признак из датасета; доступно обнуляет значения коэффициентов.

Недостатки метода: выбор модели не помогает и обычно вредит; часто страдает качество прогнозирования; выдаёт ложное срабатывание результата; случайным образом выбирает одну из коллинеарных переменных; не оценивает правильность формы взаимосвязи между независимой и зависимой переменными; не всегда лучше, чем пошаговая регрессия.

1.3. Разведочный анализ данных

Дублирующие записи не только искажают статистические показатели датасета, но и снижают качество обучения модели.

Удаляю дубликаты

```
[21] work_df.shape
```

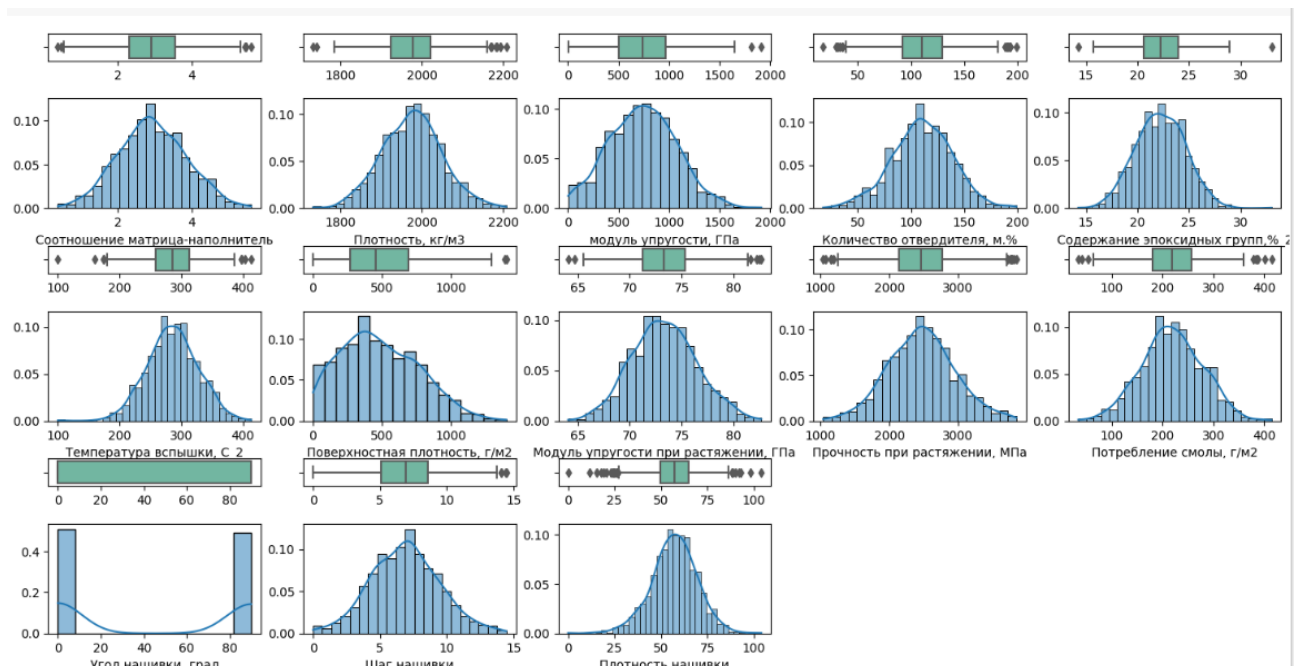
```
(1023, 13)
```

```
[22] work_df = work_df.drop_duplicates()  
work_df.shape
```

```
(1023, 13)
```

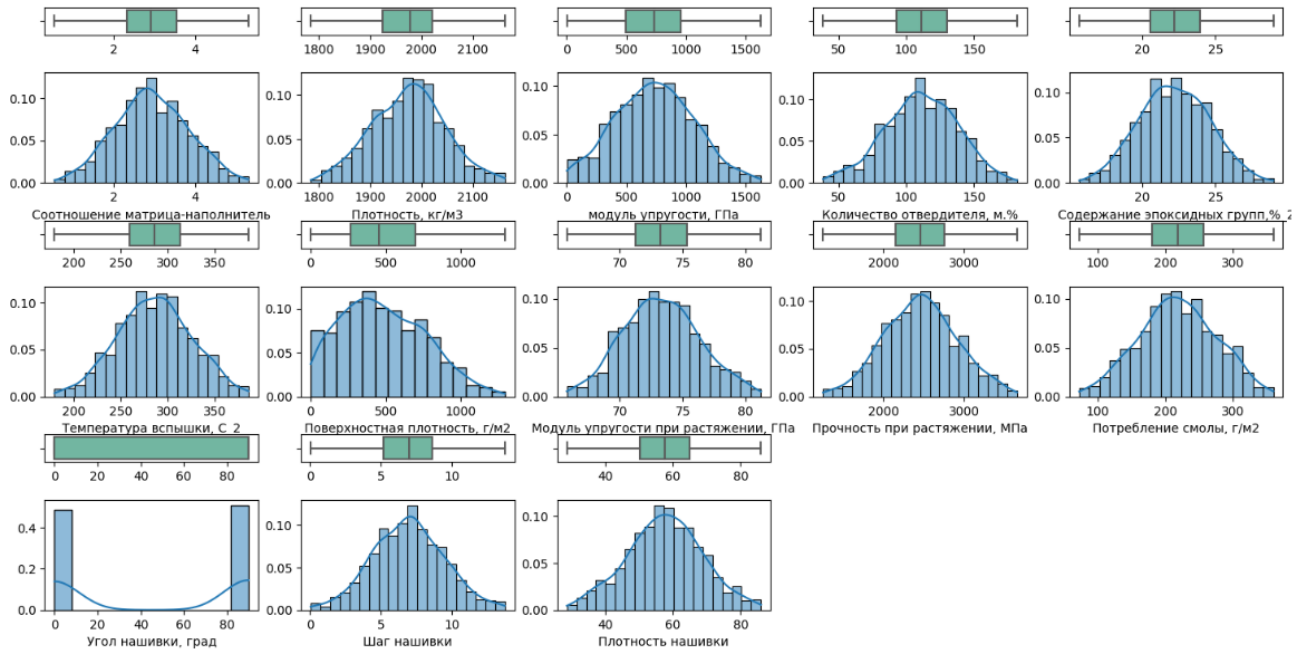
Дубликатов нет

Для большей наглядности объединил histplot и boxplot в один комбинированный график, и построил для каждого признака



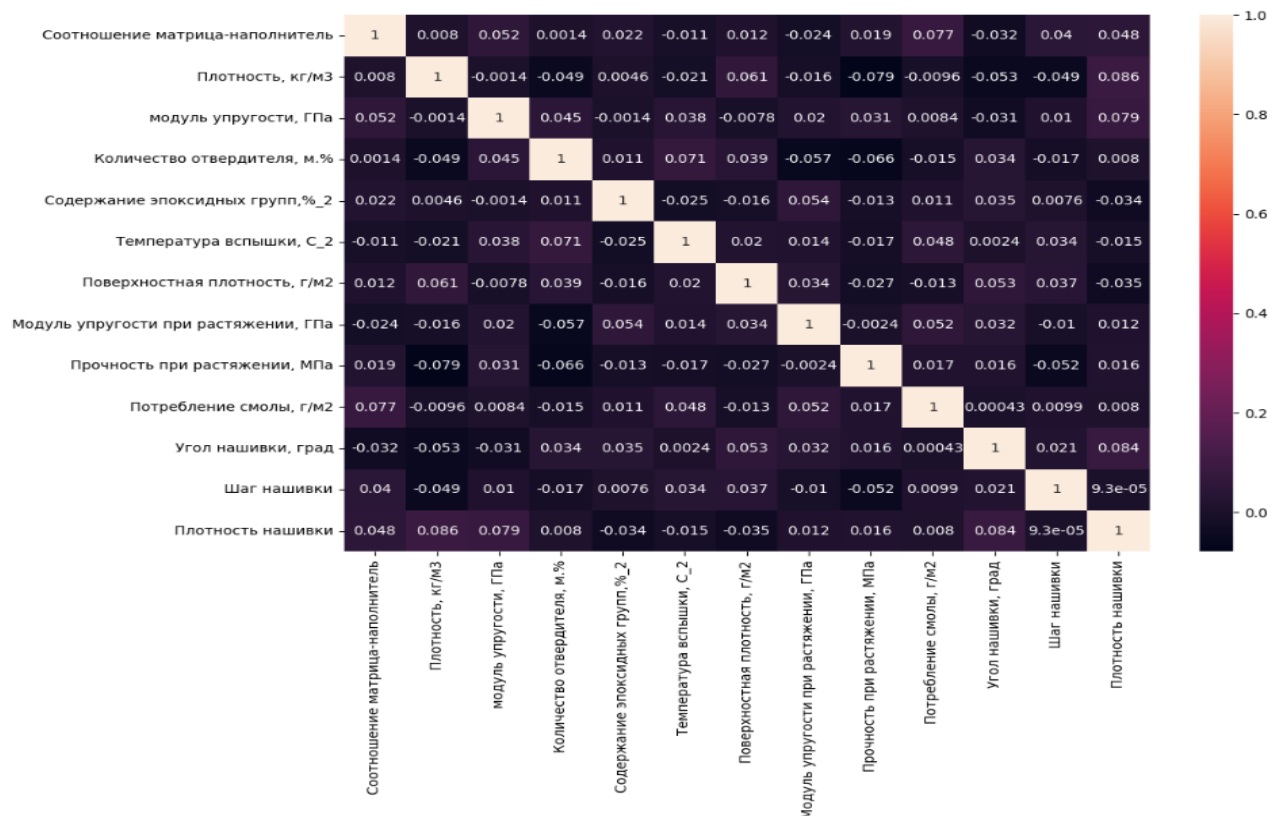
Распределение по большинству признаков близко к нормальному. Есть выбросы. Существует два распространенных способа удалить выбросы: метод межквартильного диапазона и метод трех сигм. Первый реализован в функции

method_iq(), второй – в method_3s(). После трехкратного применения method_iq() от выбросов удалось избавиться.

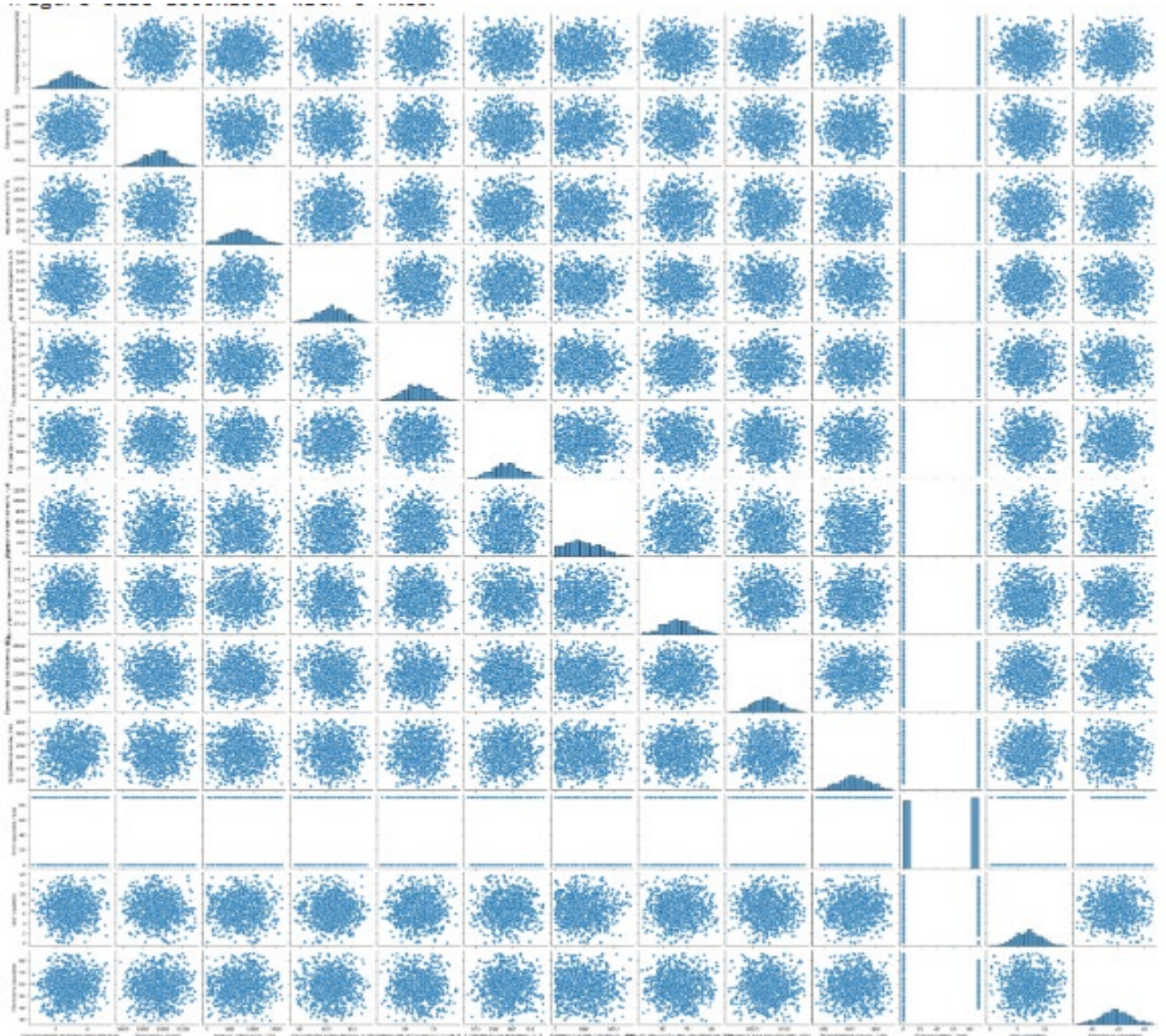


Для того, чтобы сравнить переменный друг с другом, построю тепловую карту корреляций и попарные графики рассеяния точек для всех признаков.

```
plt.figure(figsize=(12, 8))
sns.heatmap(clean_df.corr(), annot=True, )
plt.show()
```



Максимальная корреляция между плотностью нашивки и плотностью составляет 0.086, корреляция между всеми параметрами очень близка к 0, значит нет корреляционной связи между переменными



На попарных графиках распределения не видно корреляции между признаками.

Среднее, медианное значение для каждой колонки:

```
full_df.describe().loc[['mean', '50%']].T
```

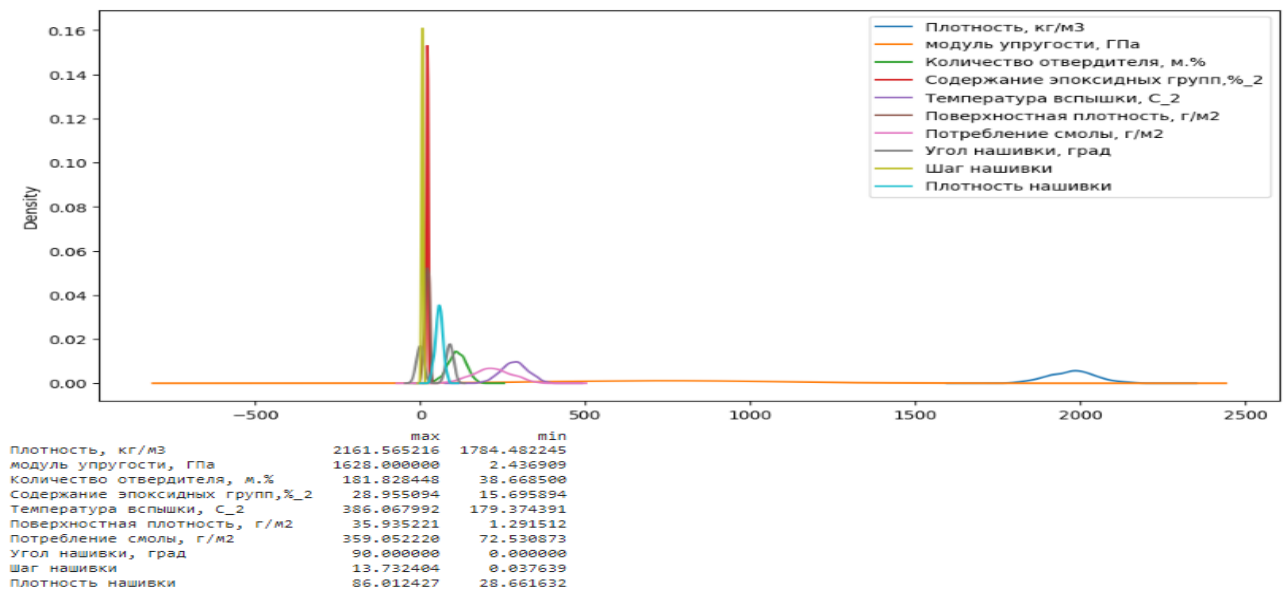
	mean	50%
Соотношение матрица-наполнитель	2.930366	2.906878
Плотность, кг/м3	1975.734888	1977.621657
модуль упругости, ГПа	739.923233	739.664328
Количество отвердителя, м.%	110.570769	110.564840
Содержание эпоксидных групп,%_2	22.244390	22.230744
Температура вспышки, С_2	285.882151	285.896812
Поверхностная плотность, г/м2	482.731833	451.864365
Модуль упругости при растяжении, ГПа	73.328571	73.268805
Прочность при растяжении, МПа	2466.922843	2459.524526
Потребление смолы, г/м2	218.423144	219.198882
Угол нашивки, град	44.252199	0.000000
Шаг нашивки	6.899222	6.916144
Плотность нашивки	57.153929	57.341920

Среднее и медианное значения по всем колонкам приблизительно равны за исключением поверхностной плотности. Здесь отклонение составляет 5.2%. Ранее на гистограмме можно было видеть на этом признаке смещение распределения влево. Для исправления ситуации можно извлечь корень.

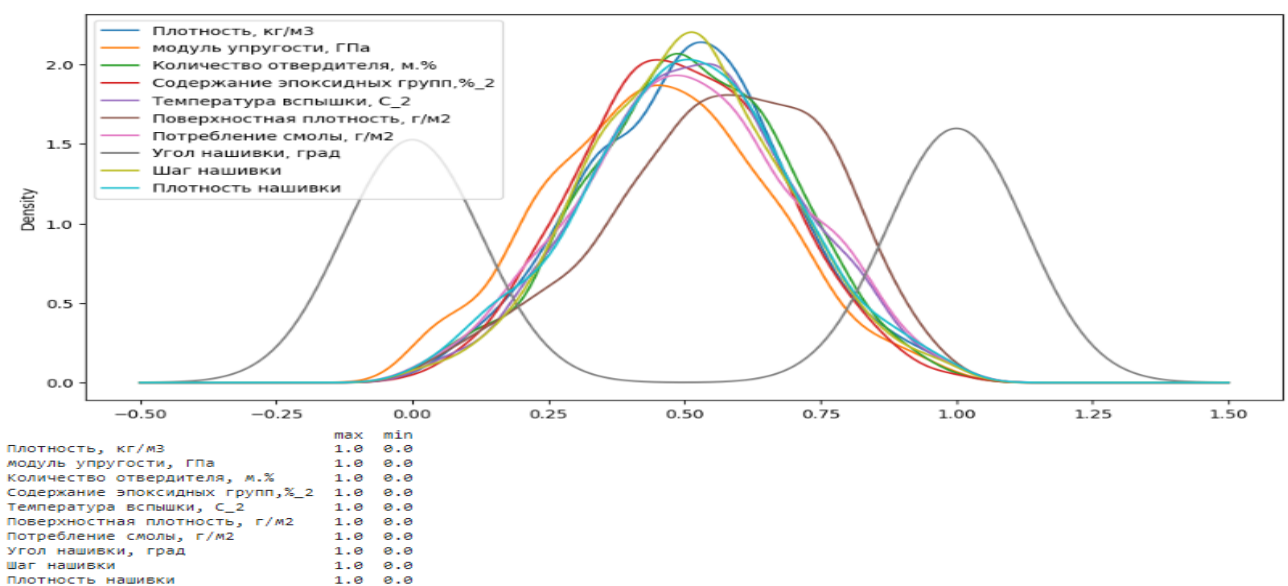
2. Практическая часть

2.1. Предобработка данных

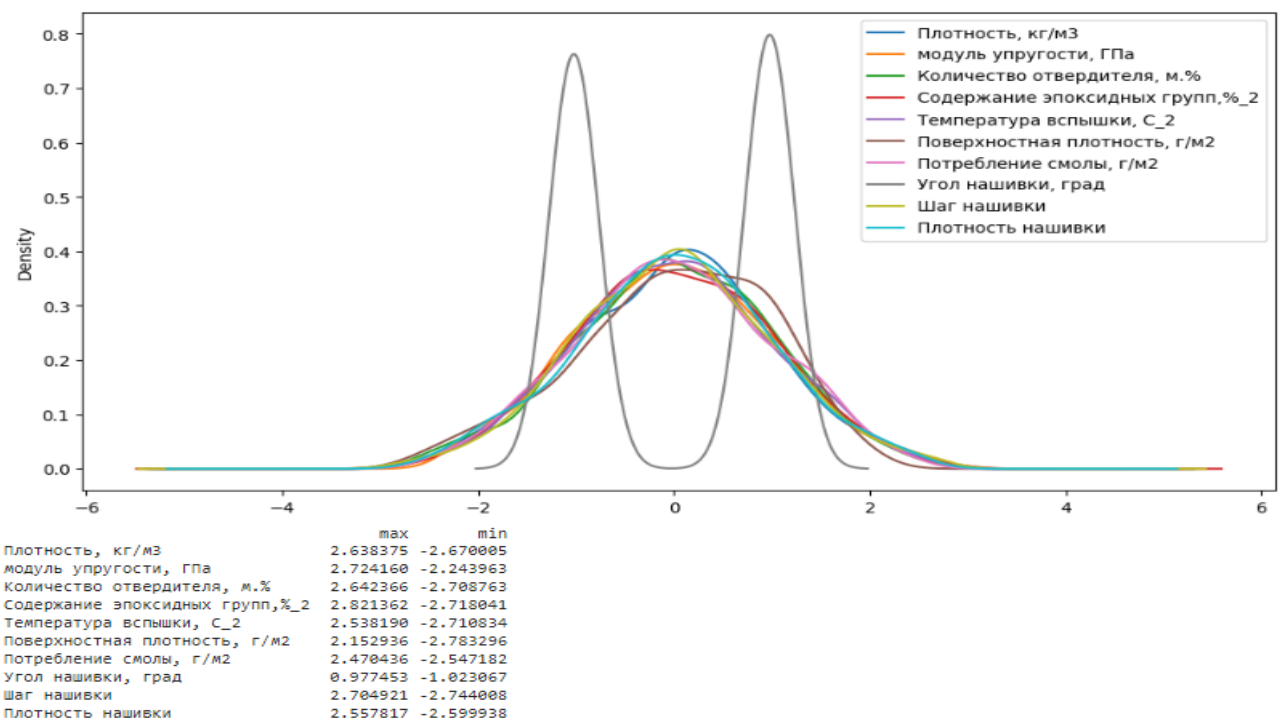
Как было замечено ранее данные имеют разный масштаб. Проведу нормализацию, построю графики распределения до и после нормализации, посмотрю максимальные и минимальные значения.



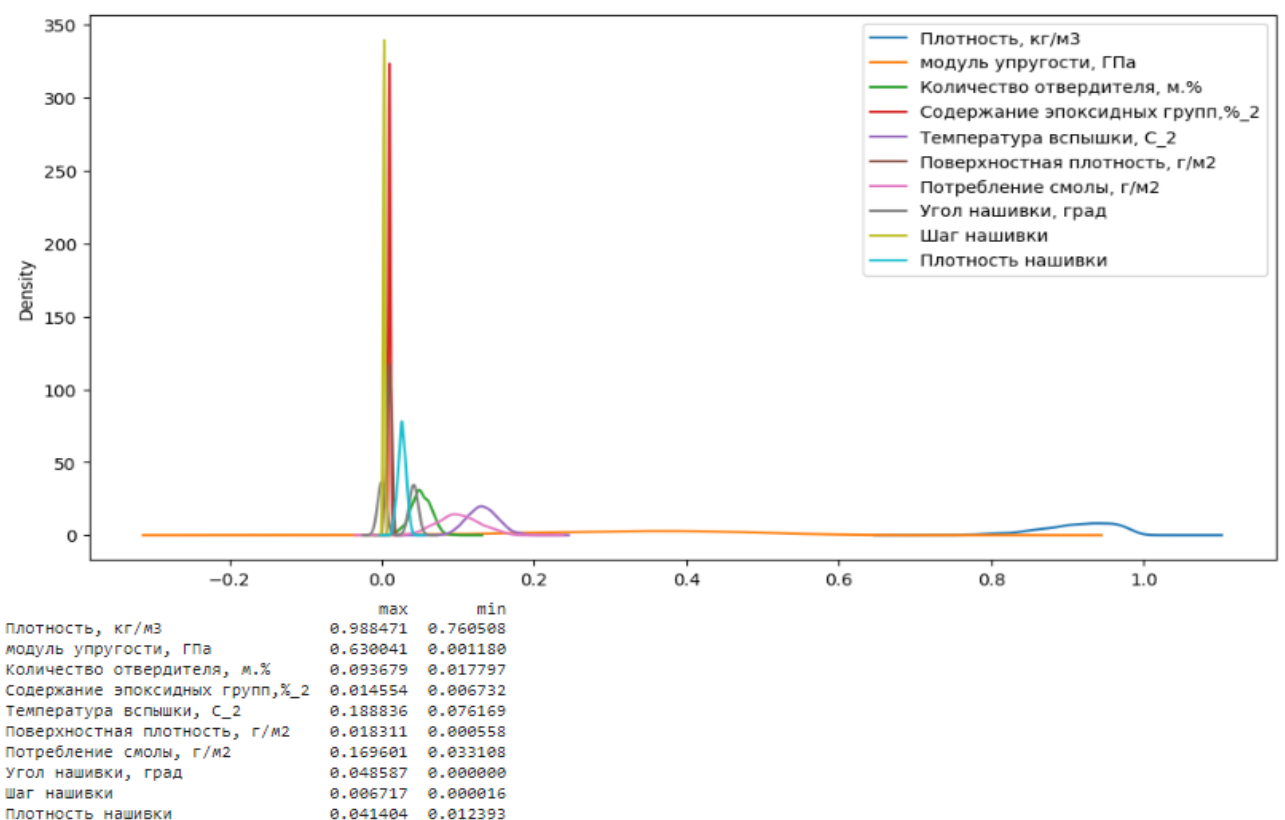
Так выглядят исходные данные. Наблюдается большой разброс.



После применения MinMaxScaler значения всех переменных укладываются в диапазон от 0 до 1



StandardScaler делает среднее значение наблюдаемых значений равным 0, а стандартное отклонение – 1



Normalizer приводит каждую строку к норме 1

Чтобы убить всех зайцев сразу, обошел в цикле датасеты (с выбросами и без), модели, препроцессоры. Все свел в одну таблицу, отсортировал по r^2

	note	r^2	MAE
1023 Lasso() MinMaxScaler()	0.009025	376.672787	
1023 Lasso() StandardScaler()	0.006825	377.205626	
1023 LinearRegression() StandardScaler()	0.005997	377.441884	
1023 LinearRegression() MinMaxScaler()	0.005997	377.441884	
1023 LinearRegression() None	0.005997	377.441884	
1023 Lasso() None	0.005851	377.474366	
921 SVR() StandardScaler()	0.000226	375.279351	
921 SVR() MinMaxScaler()	0.000040	375.336610	
921 SVR() None	-0.000219	375.320206	
921 SVR() Normalizer()	-0.000224	375.324378	
1023 SVR() None	-0.001578	379.341763	
1023 SVR() Normalizer()	-0.001587	379.331689	
921 Lasso() Normalizer()	-0.001608	376.115293	
921 Lasso() MinMaxScaler()	-0.001828	376.246037	
1023 SVR() MinMaxScaler()	-0.001960	379.546290	
1023 SVR() standardScaler()	-0.002011	379.507951	
1023 Lasso() Normalizer()	-0.002358	379.060952	
921 Lasso() StandardScaler()	-0.003493	377.016251	
921 LinearRegression() StandardScaler()	-0.004318	377.317329	
921 LinearRegression() MinMaxScaler()	-0.004318	377.317329	
921 LinearRegression() None	-0.004318	377.317329	
921 Lasso() None	-0.004390	377.304816	
1023 RandomForestRegressor() StandardScaler()	-0.013995	385.003996	
921 LinearRegression() Normalizer()	-0.014642	377.595439	
1023 RandomForestRegressor() MinMaxScaler()	-0.015935	383.559250	
1023 LinearRegression() Normalizer()	-0.025222	384.000973	
921 RandomForestRegressor() None	-0.025501	380.740480	
1023 RandomForestRegressor() None	-0.028470	383.908277	
921 RandomForestRegressor() Normalizer()	-0.033206	388.058996	
1023 GradientBoostingRegressor() MinMaxScaler()	-0.040548	391.176543	
921 RandomForestRegressor() standardScaler()	-0.042465	384.400437	
921 RandomForestRegressor() MinMaxScaler()	-0.042645	380.846385	
1023 GradientBoostingRegressor() StandardScaler()	-0.045566	391.544444	
1023 GradientBoostingRegressor() None	-0.048184	391.732341	
1023 RandomForestRegressor() Normalizer()	-0.048851	386.111038	
1023 GradientBoostingRegressor() Normalizer()	-0.061948	388.186795	
921 GradientBoostingRegressor() StandardScaler()	-0.066638	390.262835	
921 GradientBoostingRegressor() MinMaxScaler()	-0.069915	391.722409	
921 GradientBoostingRegressor() None	-0.073623	392.002986	
921 GradientBoostingRegressor() Normalizer()	-0.100567	398.295269	

Lasso MinMaxScaler на не очищенном датасете $r^2 = 0.009025$ - лучший результат. LinearRegression с выбросами $r^2 = 0.005997$ StandardScaler и MinMaxScaler на результат не повлияли он такой же, как и вообще без препроцессинга, Normalizer результат ухудшил.

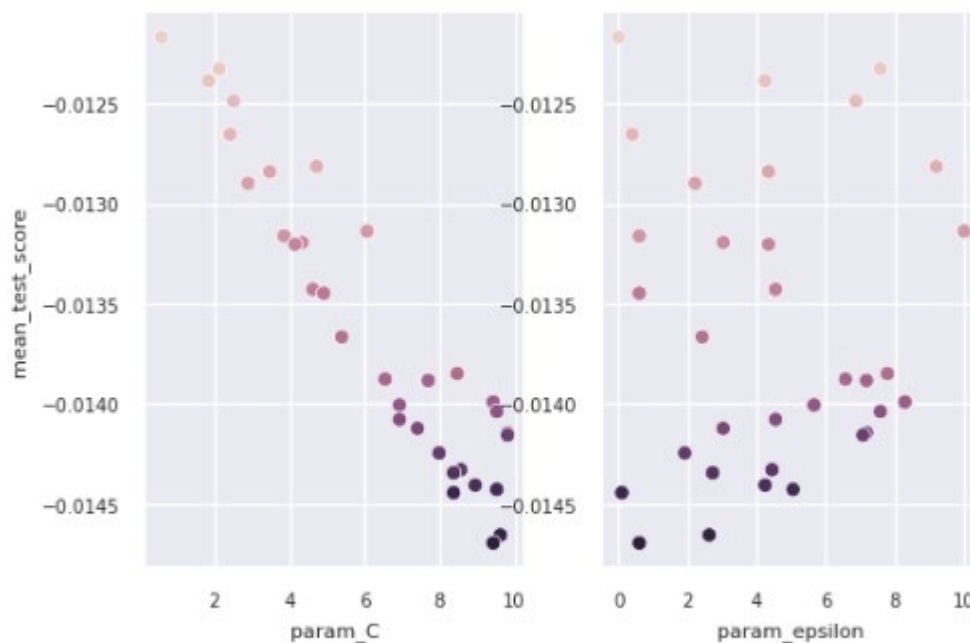
SVR лучше отработал на чистом датасете без выбросов в связке с StandardScaler $r^2 = 0.000226$

RandomForestRegressor StandardScaler на не очищенном датасете $r^2 = -0.013995$
GradientBoostingRegressor MinMaxScaler на не очищенном датасете $r^2 = -0.040548$

Продолжу работу с вышеперечисленными конфигурациями.

2.2. Разработка и обучение модели

Чтобы найти лучшую модель, воспользуюсь 2 методами RandomizedSearchCV для начального приближения параметров и GridSearchCV для уточнения параметров. Во избежание повторения кода случайный поиск гиперпараметров и визуализацию результатов поиска вынес в функцию `rs_viz`. Вот результат её работы на примере метода опорных векторов



```
StandardScaler() SVR(C=0.596, epsilon=0.001)
```

Видно, что оценка улучшается при приближении параметра «C» к нулю.

Во второй попытке можно выбрать диапазон значений для параметра «C» ближе к нулю и повторить процедуру. Следующим этапом подбираю параметры по сетке GridSearchCV, учитывая результаты предыдущего. Оценки помещаю в

таблицу `est_df`. Произведя эти действия для каждой модели, получаю для 'Прочность при растяжении, МПа' следующий результат:

	note	r2	MAE	MSE	params	transf
0	1023 MinMaxScaler() Lasso() default	0.009025	376.672787	221916.939602	Lasso()	MinMaxScaler()
1	1023 MinMaxScaler() Lasso(alpha=0.4, max_iter=2000, tol=...	0.008154	376.490015	222111.944194	Lasso(alpha=0.4, max_iter=2000, tol=7.68)	MinMaxScaler()
2	1023 StandardScaler() RandomForestRegressor(max_features=...	0.007651	376.313000	222224.678005	(DecisionTreeRegressor(max_features=0.1437142857142857, ...	StandardScaler()
3	1023 StandardScaler() Lasso() default	0.006825	377.205626	222409.541181	Lasso()	StandardScaler()
4	1023 StandardScaler() RandomForestRegressor(max_features=...	0.006409	377.966989	222502.714515	(DecisionTreeRegressor(max_features=0.021, min_samples_1...	StandardScaler()
5	1023 MinMaxScaler() LinearRegression() default	0.005997	377.441884	222595.007206	LinearRegression()	MinMaxScaler()
6	1023 StandardScaler() LinearRegression() default	0.005997	377.441884	222595.007206	LinearRegression()	StandardScaler()
7	1023 None LinearRegression() default	0.005997	377.441884	222595.007206	LinearRegression()	None
8	1023 None Lasso() default	0.005851	377.474366	222627.695909	Lasso()	None
9	1023 StandardScaler() GradientBoostingRegressor(learning...	0.001473	379.565664	223608.000736	(DecisionTreeRegressor(criterion='friedman_mse', max_de...	StandardScaler()
10	921 StandardScaler() SVR() default	0.000226	375.279351	220249.136403	SVR()	StandardScaler()
11	921 StandardScaler() SVR(C=0.596, epsilon=0.001) RndSearch	0.000068	375.285475	220283.966893	SVR(C=0.596, epsilon=0.001)	StandardScaler()
12	921 MinMaxScaler() SVR() default	0.000040	375.336610	220290.154695	SVR()	MinMaxScaler()
13	921 StandardScaler() SVR(C=0.25, epsilon=0.005) GrdSearch	-0.000115	375.295573	220324.336485	SVR(C=0.25, epsilon=0.005)	StandardScaler()
14	921 None SVR() default	-0.000219	375.320206	220347.261781	SVR()	None
15	921 Normalizer() SVR() default	-0.000224	375.324378	220348.262059	SVR()	Normalizer()
16	1023 MinMaxScaler() Lasso(alpha=9.9192, max_iter=273, pr...	-0.000283	378.983816	224001.308545	Lasso(alpha=9.9192, max_iter=273, precompute=True, tol=6...	MinMaxScaler()
17	1023 None SVR() default	-0.001578	379.341763	224291.244994	SVR()	None
18	1023 Normalizer() SVR() default	-0.001587	379.331689	224293.381910	SVR()	Normalizer()
19	921 Normalizer() Lasso() default	-0.001608	376.115293	220653.229489	Lasso()	Normalizer()
20	921 MinMaxScaler() Lasso() default	-0.001828	376.246037	220701.612915	Lasso()	MinMaxScaler()

а для 'Модуль упругости при растяжении, ГПа' такой:

	note	r2	MAE	MSE	params	transf
0	921 StandardScaler() SVR(C=0.25, epsilon=0.0001) GrdSearch	0.013171	2.434014	8.912916	SVR(C=0.25, epsilon=0.0001)	StandardScaler()
1	921 MinMaxScaler() SVR() default	0.001891	2.466912	9.014792	SVR()	MinMaxScaler()
2	921 Normalizer() SVR() default	-0.000669	2.453712	9.037911	SVR()	Normalizer()
3	921 None SVR() default	-0.001116	2.455252	9.041949	SVR()	None
4	921 StandardScaler() SVR() default	-0.002533	2.462676	9.054745	SVR()	StandardScaler()
5	921 None Lasso() default	-0.003707	2.460869	9.065354	Lasso()	None
6	921 Normalizer() Lasso() default	-0.006772	2.460820	9.093036	Lasso()	Normalizer()
7	921 MinMaxScaler() Lasso() default	-0.006772	2.460820	9.093036	Lasso()	MinMaxScaler()
8	921 StandardScaler() Lasso() default	-0.006772	2.460820	9.093036	Lasso()	StandardScaler()
9	921 MinMaxScaler() LinearRegression() default	-0.012262	2.471668	9.142619	LinearRegression()	MinMaxScaler()
10	921 StandardScaler() LinearRegression() default	-0.012262	2.471668	9.142619	LinearRegression()	StandardScaler()
11	921 None LinearRegression() default	-0.012262	2.471668	9.142619	LinearRegression()	None
12	921 Normalizer() LinearRegression() default	-0.013352	2.470731	9.152463	LinearRegression()	Normalizer()
13	921 StandardScaler() SVR(C=8.4646, epsilon=7.778) RndSearch	-0.014889	2.468936	9.166346	SVR(C=8.4646, epsilon=7.778)	StandardScaler()
14	1023 None Lasso() default	-0.015956	2.559376	10.092937	Lasso()	None
15	1023 Normalizer() Lasso() default	-0.016426	2.556417	10.097600	Lasso()	Normalizer()
16	1023 MinMaxScaler() Lasso() default	-0.016426	2.556417	10.097600	Lasso()	MinMaxScaler()
17	1023 StandardScaler() Lasso() default	-0.016426	2.556417	10.097600	Lasso()	StandardScaler()
18	1023 MinMaxScaler() Lasso(alpha=0.1, max_iter=2000, tol=...	-0.016426	2.556417	10.097600	Lasso(alpha=0.1, max_iter=2000, tol=0.00068)	MinMaxScaler()
19	1023 MinMaxScaler() Lasso(alpha=9.9192, max_iter=273, pr...	-0.016426	2.556417	10.097600	Lasso(alpha=9.9192, max_iter=273, precompute=True, tol=6...	MinMaxScaler()

Сохраняю две лучшие модели для дальнейшего использования в приложении.

2.3. Тестирование модели

Для модели 'Прочность при растяжении, МПа' ошибки на тренировочной и тестирующей части выборки составили:

r2 на тестовой: 0.008335652932277449

r2 на тренировочной: 0.02040802593721114

Для модели 'Модуль упругости при растяжении, ГПа' ошибки на тренировочной и тестирующей части выборки составили:

r2 на тестовой: -0.029439681645599514

r2 на тренировочной: 0.07053092036691377

По сути, ни одна из моделей не смогла обучиться.

Результат стабильно отвратительный, но не надо отчаиваться.

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

После экспериментов с построением сети вручную, решил воспользоваться инструментом `kerastuner`, который позволяет автоматически подбирать параметры нейросети, стремясь найти наилучшие. Для создания моделей реализована функция `build_model`. По итогам работы `kerastuner`, была выбрана модель со следующими результатами на тестовой и тренировочной выборках:

```
model.evaluate(X_test, y_test)
```

```
9/9 [=====] - 0s 2ms/step - loss: 0.8057 - mean_squared_error: 0.8057  
[0.8056550621986389, 0.8056550621986389]
```

```
model.evaluate(X_train, y_train)
```

```
21/21 [=====] - 0s 2ms/step - loss: 0.8017 - mean_squared_error: 0.8017  
[0.801701545715332, 0.801701545715332]
```

Вот её архитектура:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 576)	7488
dense_1 (Dense)	(None, 480)	276960
dense_2 (Dense)	(None, 1)	481
Total params: 284,929		
Trainable params: 284,929		
Non-trainable params: 0		

```
model.get_compile_config()
```

```
{'optimizer': 'adam',  
'loss': 'mean_squared_error',  
'metrics': ['mean_squared_error'],  
'loss_weights': None,  
'weighted_metrics': None,  
'run_eagerly': None,  
'steps_per_execution': None,  
'jit_compile': None}
```

Модель сохранена в файл 'Соотношение матрица-наполнитель' и может быть использована для дальнейшего обучения или при разработке приложения.

2.5. Разработка приложения

Для разработки приложения `composit.py` использовал библиотеку Flask. Flask — это небольшой и легкий веб-фреймворк, написанный на языке Python, предлагающий полезные инструменты и функции для облегчения процесса создания веб-приложений с использованием Python. Он обеспечивает гибкость и является более доступным фреймворком для новых разработчиков, так как позволяет создать веб-приложение быстро, используя только один файл Python. Flask — это расширяемая система, которая не обязывает использовать конкретную структуру директорий и не требует сложного шаблонного кода перед началом использования.

Приложение composit.py находится в папке app. Для запуска можно использовать сервер с поддержкой Flask или локальный компьютер с предварительно установленными зависимостями. Интерфейс приложения доступен в браузере по адресу hostname:5000/. Выглядит следующим образом:

Прогноз модуля упругости при растяжении, прочности при растяжении

Соотношение матрица-наполнитель	<input type="text" value="2.334566013"/>
Плотность, кг/м ³	<input type="text" value="2020.97"/>
модуль упругости, ГПа	<input type="text" value="1013.514"/>
Количество отвердителя, м. %	<input type="text" value="109.5051"/>
Содержание эпоксидных групп, %_2	<input type="text" value="24.3309"/>
Температура вспышки, C_2	<input type="text" value="266.2586"/>
Поверхностная плотность, г/м ²	<input type="text" value="719.87"/>
Потребление смолы, г/м ²	<input type="text" value="264.8356"/>
Угол нашивки, град	<input type="text" value="0"/>
Шаг нашивки	<input type="text" value="7.3636"/>
Плотность нашивки	<input type="text" value="69.17234"/>
<input type="button" value="рассчитать"/>	

Ожидаемый модуль упругости при растяжении, ГПа: 73.938

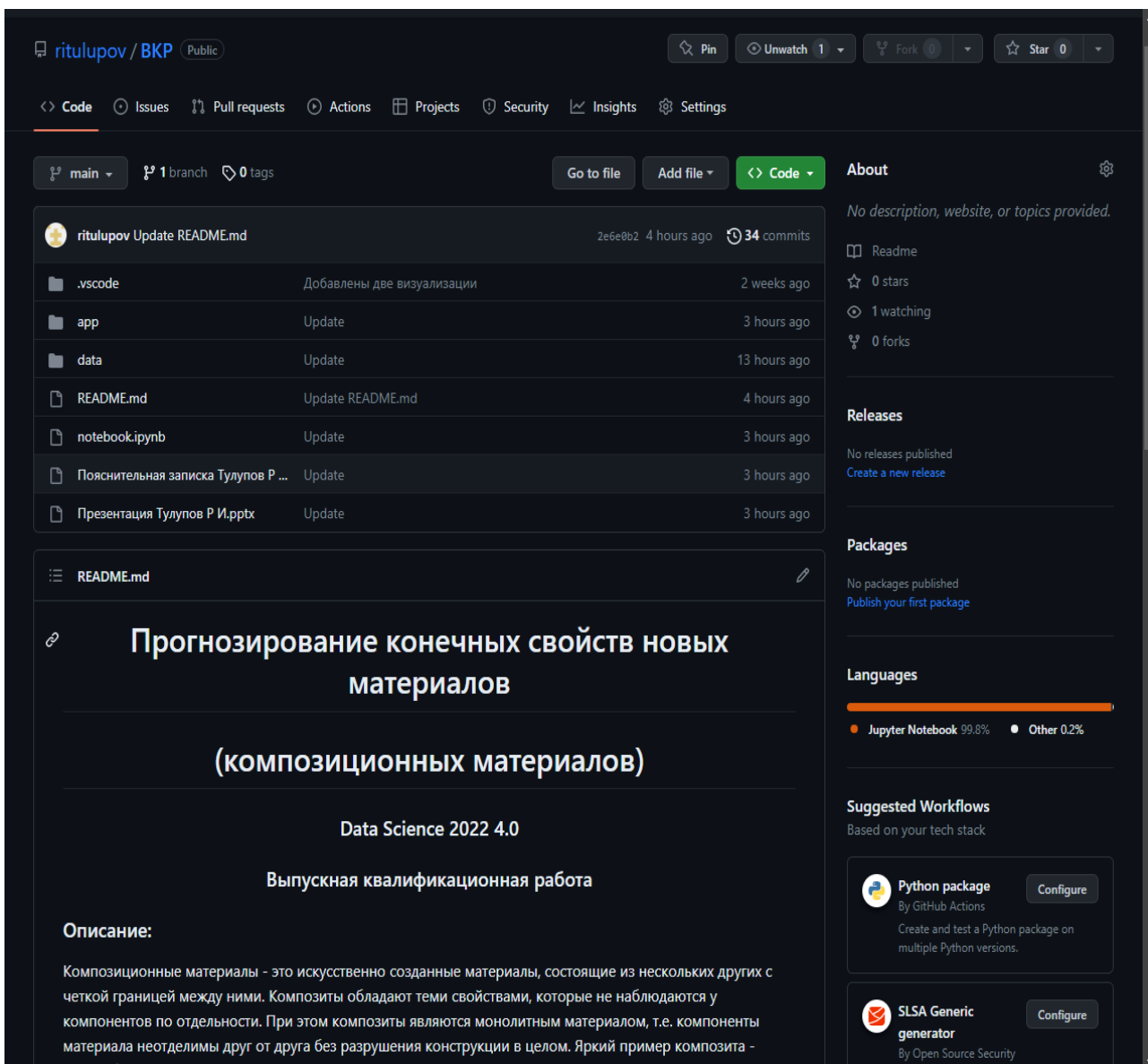
Ожидаемая прочность при растяжении, МПа: 2472.436

Для того, чтобы рассчитать модуль упругости при растяжении и прочность при растяжении необходимо заполнить все поля формы и нажать кнопку «рассчитать». Результат отобразится под кнопкой.

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

Репозиторий приложения composit.py расположен по ссылке

<https://github.com/ritulupov/BKP>



The screenshot displays the GitHub repository page for `ritulupov/BKP`. The repository is public and has 1 branch and 0 tags. The file list includes `.vscode`, `app`, `data`, `README.md`, `notebook.ipynb`, `Пояснительная записка Тулупов Р ...`, and `Презентация Тулупов Р И.pptx`. The README content is visible, showing the title "Прогнозирование конечных свойств новых материалов" and the subtitle "(композиционных материалов)". The repository is associated with "Data Science 2022 4.0" and is a "Выпускная квалификационная работа". The description states that compositional materials are artificially created materials consisting of several other materials with a clear boundary between them. The repository statistics show 34 commits, 0 stars, 1 watching, and 0 forks. The right sidebar shows the repository is not described, has no releases, and no packages published. The languages section shows 99.8% Jupyter Notebook and 0.2% Other. Suggested workflows for Python package and SLSA Generic generator are also visible.

[Commits on Apr 25, 2023](#)

Заключение

В ходе проделанной работы стало ясно, что предложенный для анализа набор данных – крепкий орешек. Изначально распределение данных было близко к нормальному, выбросы не большие, пропусков и дубликатов нет. В процессе подготовки данные были очищены от выбросов, нормализованы. Для каждой из пяти моделей, выбранных для обучения, был проведен автоматический подбор подходящего набора данных (очищенные и не очищенные), нормализатора (MinMaxScaler, StandardScaler, Normalizer). Хочется отметить, что на линейную регрессию не повлияло применение MinMaxScaler и StandardScaler – результат оказался такой же, как и без них. Затем для каждой модели был проведен поиск оптимальных гиперпараметров случайно и по сетке, но несмотря на все усилия результат оказался плачевным. Построение нейронной сети с автоматическим подбором гиперпараметров при помощи kerastuner, позволило выбрать лучшие параметры, но и с ними модель ничему не смогла научиться.

Судя по всему, в предоставленном для анализа датасете сокрыт какой-то секрет, который, к огромному сожалению, оказался мне не по силам. Но несмотря на это я получил хороший опыт и огромное удовольствие от проделанной работы.

Список литературы и веб ресурсы

1. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Издательство Казанского университета, 2018
2. <https://machinelearningmastery.ru/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8bf81991d0c5/>
3. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru
4. Andre Ye. 5 алгоритмов регрессии в машинном обучении, о которых вам сле-дует знать: <https://habr.com/ru/company/vk/blog/513842/>
5. Документация по библиотеке keras: <https://keras.io/api/>.
6. Документация по библиотеке matplotlib:
<https://matplotlib.org/stable/users/index.html>.
7. Документация по библиотеке numpy:
<https://numpy.org/doc/1.22/user/index.html#user>.
8. Документация по библиотеке pandas:
https://pandas.pydata.org/docs/user_guide/index.html#user-guide.
9. Документация по библиотеке scikit-learn: https://scikit-learn.org/stable/user_guide.html.
10. Документация по библиотеке seaborn:
<https://seaborn.pydata.org/tutorial.html>.

11. Документация по библиотеке Tensorflow:
<https://www.tensorflow.org/overview>
12. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы:
учебное пособие для вузов, 2019. 13 с.
13. Макс Ветков. Градиентный бустинг(AdaBoost) <https://your-scorpion.ru/gradient-boosting-adaboost>
14. Руководство по быстрому старту в flask: – Режим доступа:
<https://flaskrussian-docs.readthedocs.io/ru/latest/quickstart.html>
15. https://github.com/ugapanyuk/ml_course_2022/;
16. <https://wiki.loginom.ru/>;
17. <https://ai-news.ru/>;
18. https://keras.io/api/keras_tuner/tuners/bayesian/;