

**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»**

**Тема: «Прогнозирование конечных свойств новых материалов  
(композиционных материалов)»**

Слушатель

Шортанова Регина Исфандияровна

Москва, 2023

# Содержание

Содержание.....	2
Введение .....	2
1. Аналитическая часть.....	4
1.1. Постановка задачи .....	4
1.2. Описание используемых методов.....	7
1.3. Разведочный анализ данных.....	18
2. Практическая часть.....	23
2.1. Предобработка данных .....	23
2.2. Разработка и обучение модели.....	24
2.3. Тестирование модели .....	25
2.4. Написать нейронную сеть, которая будет рекомендовать .....	26
2.5. Список используемой литературы и веб ресурсы. ....	33
2.6. Приложение 1.....	36

## Введение

Композиционные материалы — это материалы, состоящие из двух или более компонентов, нерастворимых друг с другом, с чётко обозначенной границей раздела и сильным взаимодействием по всей зоне контакта. Одним из компонентов композитных материалов является непрерывная фаза, он называется матрица, в которой нерастворимые материалы помещаются в другую природу, называемую арматурой или наполнителем.

Внедрение композиционных материалов обусловлено стремлением использовать их преимущества по сравнению с традиционно используемыми металлами и сплавами. Примеры композита – железобетон (сочетание стали арматуры и камня бетона), древесноволокнистая плита ДВП (сочетание древесной основы – щепы и полимерного связующего).

Базальт - магматическая вулканическая порода. Это самая распространённая порода на поверхности Земли и на других планетах Солнечной системы. Базальты образуются путём затвердевания силикатного магматического расплава. Большая часть базальтов образуется на срединно-океанических хребтах и образует океаническую кору. Активно развивается использование композитных материалов на основе базальта.

Базальтопластик - современный композитный материал на основе базальтовых волокон и органического связующего вещества. В настоящее время базальтопластик успешно конкурирует с металлическими изделиями, превосходя их по коррозионной, щелочной, кислотоустойчивости и некоторым другим свойствам. Целью данной работы является прогнозирование конечных свойств новых материалов на основе базальтопластика (композиционных материалов).

Расширение разнообразия материалов, используемых при проектировании нового композиционного материала, увеличивает необходимость определения свойств нового композита при минимальных финансовых затратах. Для решения этой проблемы обычно используются два способа: физические тесты образцов материалов или оценка свойств, в том числе на основе физико-математических моделей. Традиционно разработка композитных материалов является долгосрочным процессом, так как из свойств отдельных компонентов невозможно рассчитать конечные свойства композита. Для достижения определенных характеристик требуется большое количество различных комбинированных тестов, что делает насущной задачу прогнозирования успешного решения, снижающего затраты на разработку новых материалов и затраты на рабочую силу. Суть прогнозирования

заключается в моделировании репрезентативного элемента композитного объёма на основе данных о свойствах входящих компонентов (связующего и армирующего компонента). В процессе исследовательской работы были разработаны несколько моделей, способные с высокой вероятностью прогнозировать модули упругости при растяжении и прочности при растяжении, а также были созданы 2 нейронных сети, которые предлагают соотношение «матрицы - наполнитель».

## 1. Аналитическая часть

### 1.1. Постановка задачи

Для исследовательской работы были даны 2 файла: X\_br.xlsx (с данными о параметрах базальтопластика, состоящий из 1023 строк и 11 столбцов) и X\_npr.xlsx (данными нашивок углепластика, состоящий из 1040 строк и 4 столбцов).

#### 1. Загрузить и проработать входящие датасеты

```
Ввод [2]: #1-Загружаем первый датасет (базальтопластик) и посмотрим на названия столбцов
df_bp = pd.read_excel(r"C:\Users\user\Desktop\МГТУ учеба\vr\datasets\X_br.xlsx")
df_bp.shape

Out[2]: (1023, 11)
```

#### 1.1 Удаляем первый неинформативный столбец

```
Ввод [3]: df_bp.drop(['Unnamed: 0'], axis=1, inplace=True)
#Посмотрим на первые 10 строк первого датасета и убедимся, что первый столбец удалился
df_bp.head(10)

Out[3]:
```

	Соотношение матрица-наполнитель	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
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0
5	2.767918	2000.0	748.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0
6	2.569620	1910.0	807.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0
7	2.561475	1900.0	535.000000	111.86	22.267857	284.615385	380.0	75.0	1800.0	120.0
8	3.557018	1930.0	889.000000	129.00	21.250000	300.000000	380.0	75.0	1800.0	120.0
9	3.532338	2100.0	1421.000000	129.00	21.250000	300.000000	1010.0	78.0	2000.0	300.0

Рисунок 1 – пример начала работы с файлом X\_br.xlsx

Цель работы разработать модели для прогноза модуля упругости при растяжении, прочности при растяжении и соотношения «матрица-наполнитель». Для

этого нужно объединить 2 файла. Часть информации (17 строк таблицы способов компоновки композитов) не имеют соответствующих строк в таблице соотношений и свойств используемых компонентов композитов, поэтому были удалены.

```
Ввод [5]: #Загружаем второй датасет (углепластик)
df_nup = pd.read_excel(r"C:\Users\user\Desktop\МГТУ учеба\ukr\datasets\X_nup.xlsx")
df_nup.shape

Out[5]: (1040, 4)
```

```
Ввод [6]: #Удаляем первый неинформативный столбец
df_nup.drop(['Unnamed: 0'], axis=1, inplace=True)
#Посмотрим на первые 10 строк второго датасета и убедимся, что и здесь не нужен первый столбец успешно удален
df_nup.head(10)

Out[6]:
```

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0	4.0	57.0
1	0	4.0	60.0
2	0	4.0	70.0
3	0	5.0	47.0
4	0	5.0	57.0
5	0	5.0	60.0
6	0	5.0	70.0
7	0	7.0	47.0
8	0	7.0	57.0
9	0	7.0	60.0

```
Ввод [7]: # Проверим размерность второго файла
df_nup.shape

Out[7]: (1040, 3)
```

Рисунок 2 - пример начала работы с файлом X\_nup.xlsx

Затем провести разведочный анализ данных, нарисовать гистограммы распределения каждой из переменной, диаграммы ящика с усами, попарные графики рассеяния точек.

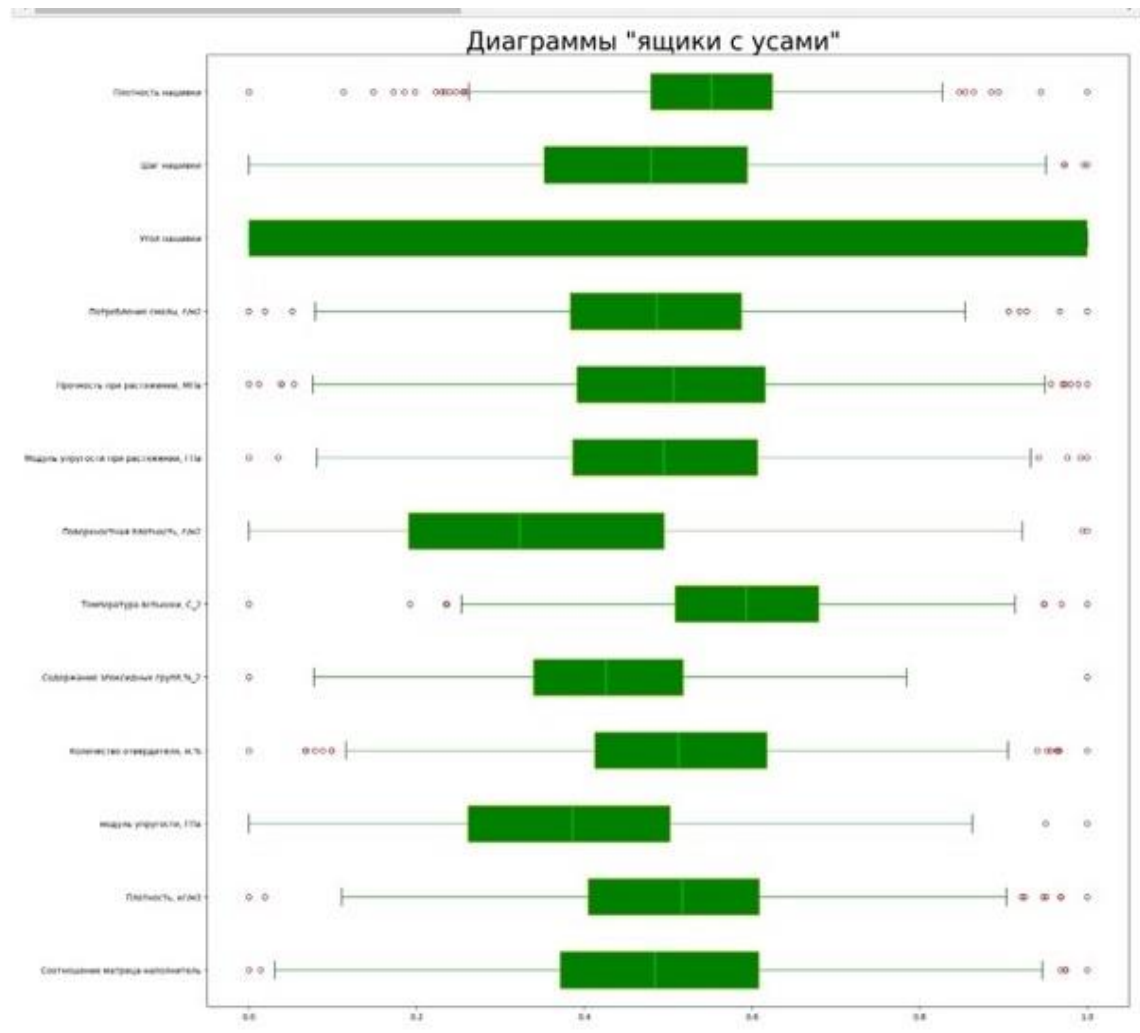


Рисунок 3 - диаграмма "ящик с усами" в объединённом датасете

Для каждой колонки получить среднее, медианное значение, провести анализ и исключить выбросы, проверить наличие пропусков; предобработать данные: удалить шумы и выбросы, сделать нормализацию и стандартизацию. Обучить несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении. Написать нейронную сеть, которая будет рекомендовать соотношение «матрица-наполнитель». Разработать приложение с графическим интерфейсом, которое будет выдавать прогноз соотношения «матрица-наполнитель». Оценить точность модели на тренировочном и тестовом датасете. Создать репозиторий в GitHub и разместить код исследования. Оформить файл README.

## 1.2. Описание используемых методов

Данная задача в рамках классификации категорий машинного обучения относится к машинному обучению с учителем и традиционно это задача регрессии. Цель любого алгоритма обучения с учителем — определить функцию потерь и минимизировать её, поэтому для наилучшего решения в процессе исследования были применены следующие методы:

- метод опорных векторов;
- случайный лес;
- линейная регрессия;
- градиентный бустинг;
- К-ближайших соседей;
- дерево решений;
- стохастический градиентный спуск;
- многослойный перцептрон;
- Лассо;

Метод опорных векторов (Support Vector Regression) – этот бинарный линейный классификатор был выбран, потому что он хорошо работает на небольших датасетах. Данный алгоритм – это алгоритм обучения с учителем, использующихся для задач классификации и регрессионного анализа, это контролируемое обучение моделей с использованием схожих алгоритмов для анализа данных и распознавания шаблонов. Учитывая обучающую выборку, где 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, и он максимален.

Каждый объект данных представляется как вектор (точка) в  $r$ -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, поэтому в работе была проведена тщательнейшая работа с выбросами, иначе в обучающих данных шумы становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости; для больших наборов данных требуется долгое время обучения; достаточно сложно подбирать полезные преобразования данных; параметры модели сложно интерпретировать, поэтому были рассмотрены и другие методы.



Рисунок 4 - график метода опорных векторов для прочности при растяжении, МПа



Случайный лес (RandomForest) — это множество решающих деревьев. Универсальный алгоритм машинного обучения с учителем, представитель ансамблевых методов. Если точность дерева решений оказалась недостаточной, мы можем множество моделей собрать в коллектив.

Достоинства метода: не переобучается; не требует предобработки входных данных; эффективно обрабатывает пропущенные данные, данные с большим числом классов и признаков; имеет высокую точность предсказания и внутреннюю оценку обобщающей способности модели, а также высокую параллелизуемость и масштабируемость.

Недостатки метода: построение занимает много времени; сложно интерпретируемый; не обладает возможностью экстраполяции; может недо обучаться; трудоёмко прогнозируемый; иногда работает хуже, чем линейные методы.

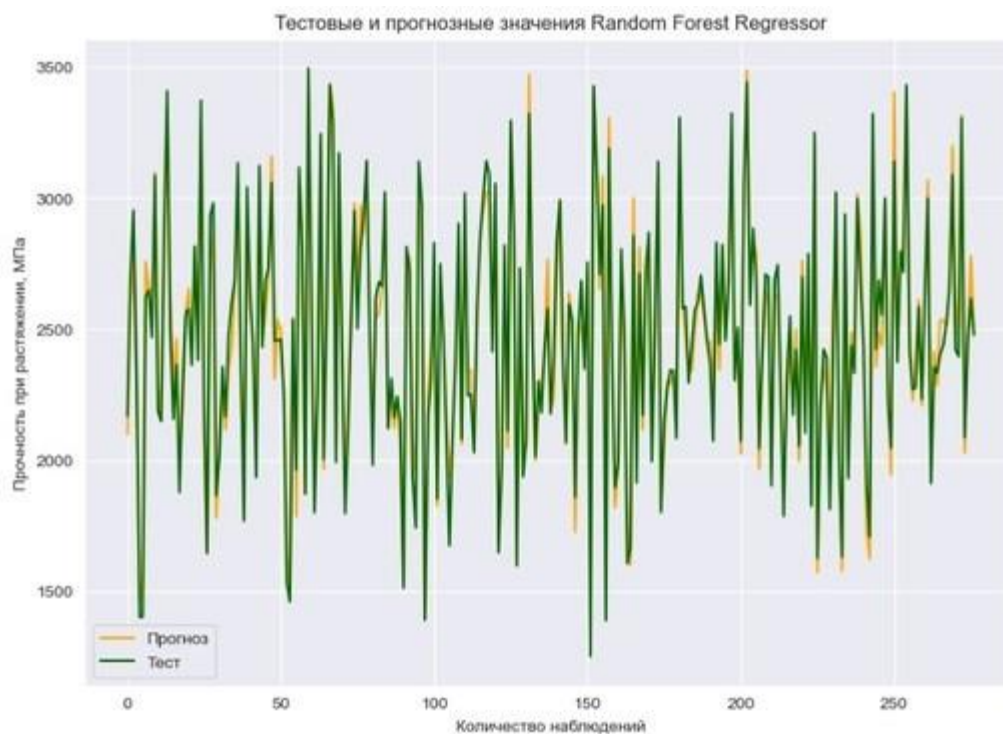


Рисунок 5 - график "случайного леса" для прочности при растяжении, МПа

Линейная регрессия (Linear regression) — это алгоритм машинного обучения, основанный на контролируемом обучении, рассматривающий зависимость между одной входной и выходными переменными. Это один из самых простых и

эффективных инструментов статистического моделирования. Она определяет зависимость переменных с помощью линии наилучшего соответствия. Модель регрессии создаёт несколько метрик.  $R^2$ , или коэффициент детерминации, 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 процентов дисперсии данных. Оставшиеся отклонения не имеют объяснения. Чем ближе  $R^2$  к единице, тем лучше.

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.

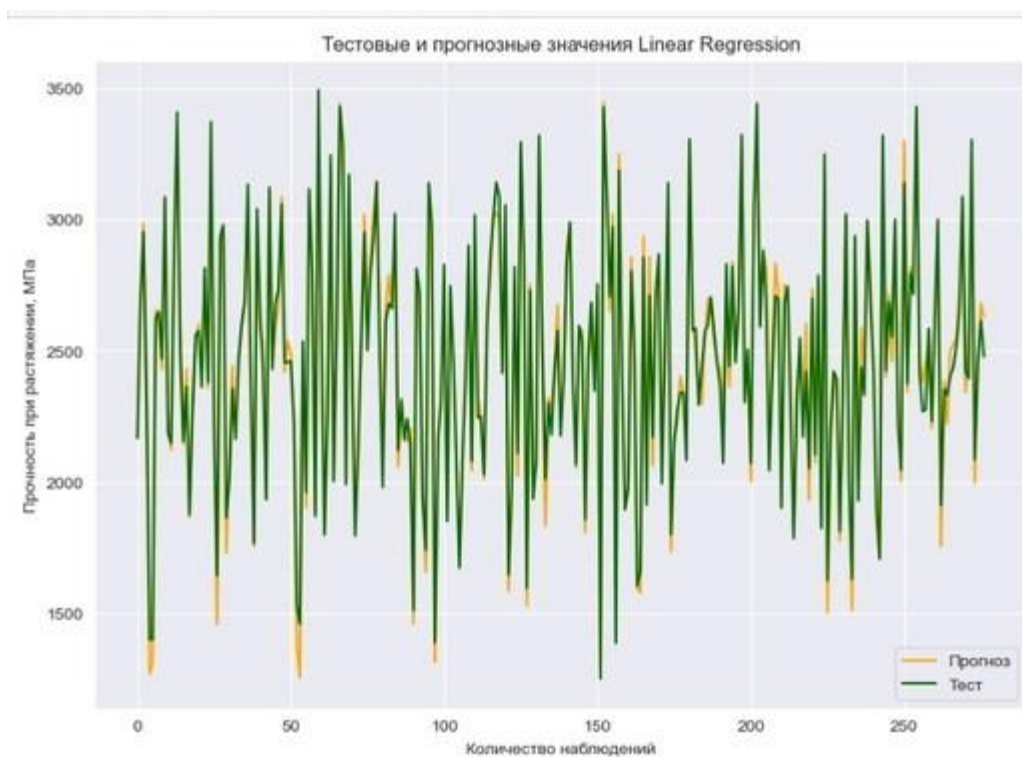


Рисунок 6- график линейной регрессии для прочности при растяжении, МПа

Градиентный бустинг (Gradient Boosting) — 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 гибко чем нейронные сети.

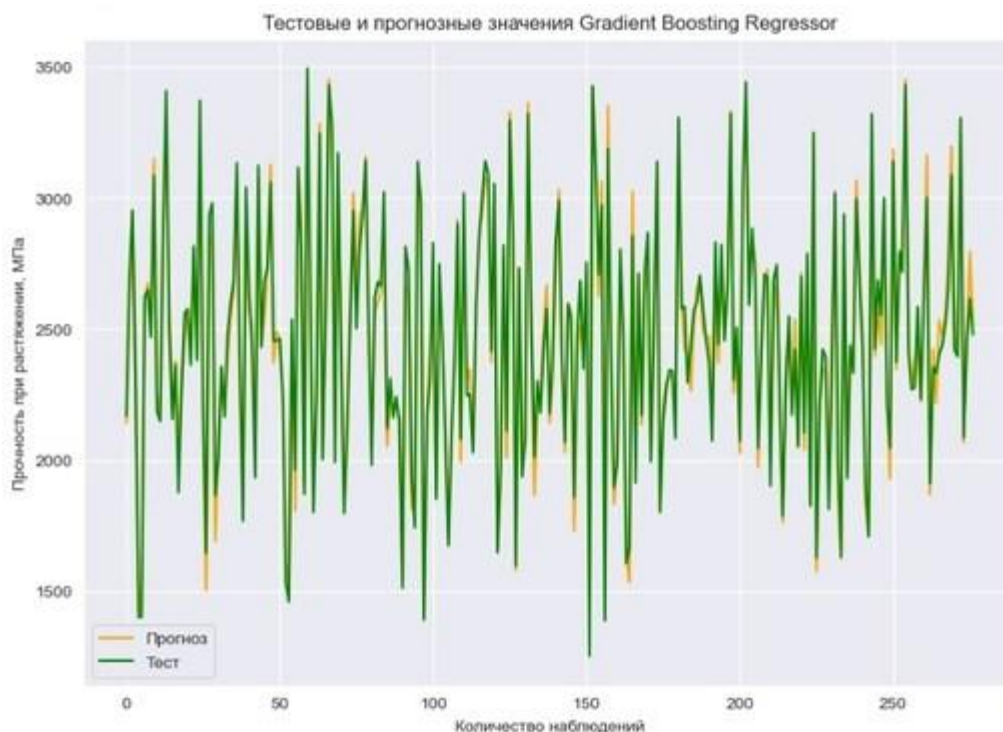


Рисунок 7 - график градиентного бустинга для прочности при растяжении, МПа

Метод ближайших соседей - К-ближайших соседей (kNN - k Nearest Neighbours) ищет ближайшие объекты с известными значения целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами.

Алгоритм находит расстояния между запросом и всеми примерами в данных, выбирая определенное количество примеров ( $k$ ), наиболее близких к запросу, затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Достоинства метода: прост в реализации и понимании полученных результатов; имеет низкую чувствительность к выбросам; не требует построения модели; допускает настройку нескольких параметров; позволяет делать дополнительные допущения; универсален; находит лучшее решение из возможных; решает задачи не большой размерности.

Недостатки метода: замедляется с ростом объёма данных; не создаёт правил; не обобщает предыдущий опыт; основывается на всем массиве доступных исторических данных; невозможно сказать, на каком основании строятся ответы; сложно выбрать близость метрики; имеет высокую зависимость результатов классификации от выбранной метрики; полностью перебирает всю обучающую выборку при распознавании; имеет вычислительную трудоёмкость.

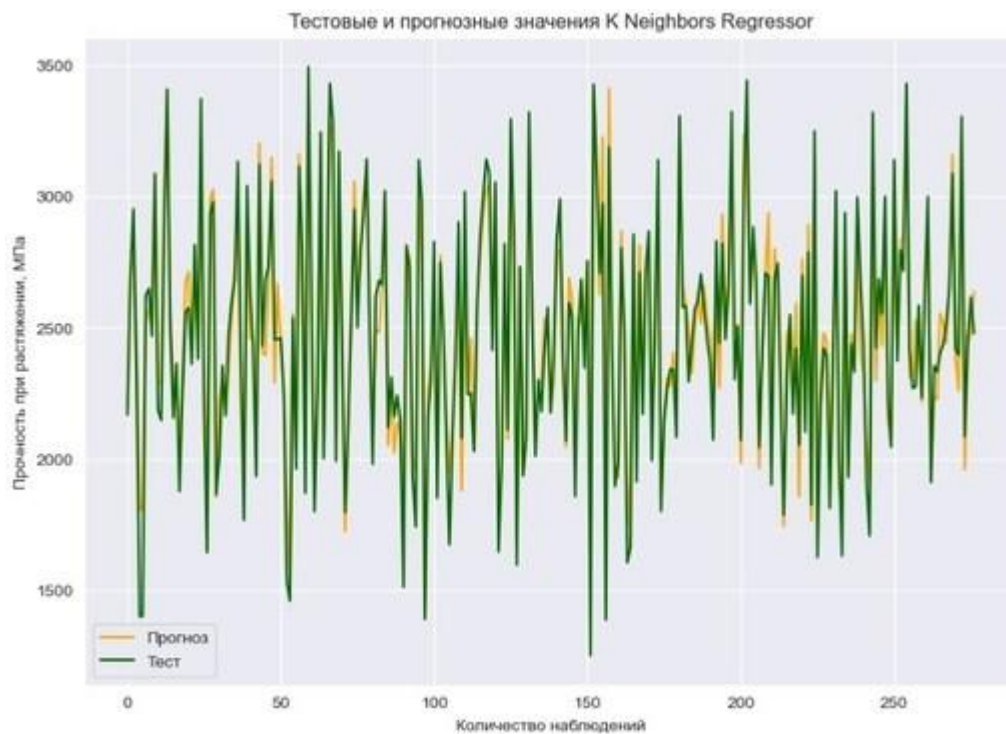


Рисунок 8 - график K-ближайших соседей для прочности при растяжении, МПа

Дерево принятия решений (DecisionTreeRegressor) – метод автоматического анализа больших массивов данных. Это инструмент принятия решений, в котором используется древовидная структура, подобная блок-схеме, или модель решений и всех их возможных результатов, включая результаты, затраты и полезность. Дерево принятия решений - эффективный инструмент интеллектуального анализа данных и предсказательной аналитики. Алгоритм дерева решений подпадает под категорию контролируемых алгоритмов обучения. Он работает как для непрерывных, так и для категориальных выходных переменных. Правила генерируются за счёт обобщения множества отдельных наблюдений (обучающих примеров), описывающих предметную область. Регрессия дерева решений отслеживает особенности объекта и обучает модель в структуре дерева прогнозированию данных в будущем для получения значимого непрерывного вывода. Дерево решений один из вариантов решения регрессионной задачи, в случае если зависимость в данных не имеет очевидной корреляции.

Достоинства метода: помогают визуализировать процесс принятия решения и сделать правильный выбор в ситуациях, когда результаты одного решения влияют на результаты следующих решений; создаются по понятным правилам; просты в применении и интерпретации; заполняют пропуски в данных наиболее вероятным решением; работают с разными переменными; выделяют наиболее важные поля для прогнозирования;

Недостатки метода: ошибаются при классификации с большим количеством классов и небольшой обучающей выборкой; имеют нестабильный процесс (изменение в одном узле может привести к построению совсем другого дерева); имеет затратные вычисления; необходимо обращать внимание на размер; ограниченное число вариантов решения проблемы.



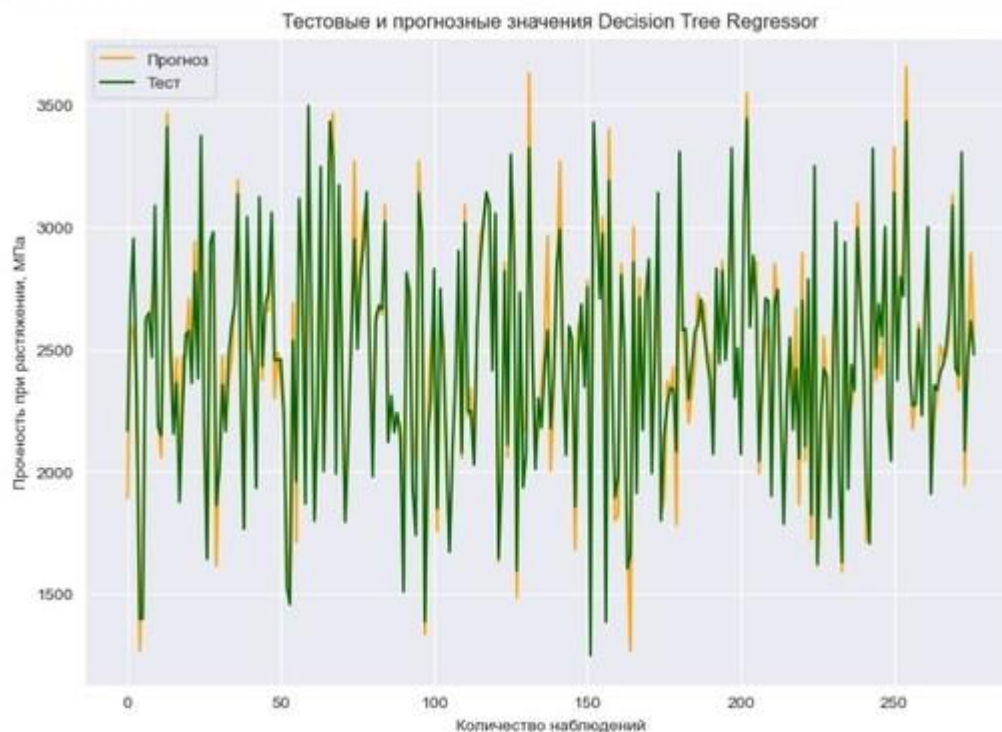


Рисунок 9 - график дерева принятия решений для прочности при растяжении

Стохастический градиентный спуск (SGDRegressor) — это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь. Этот подход подразумевает корректировку весов нейронной сети, используя аппроксимацию градиента функционала, вычисленную только на одном случайном обучающем примере из выборки.

Достоинства метода: эффективен; прост в реализации; имеет множество возможностей для настройки кода; способен обучаться на избыточно больших выборках.

Недостатки метода: требует ряд гиперпараметров; чувствителен к масштабированию функций; может не сходиться или сходиться слишком медленно; функционал многоэкстремален; процесс может "застрять" в одном из локальных минимумов; возможно переобучение.

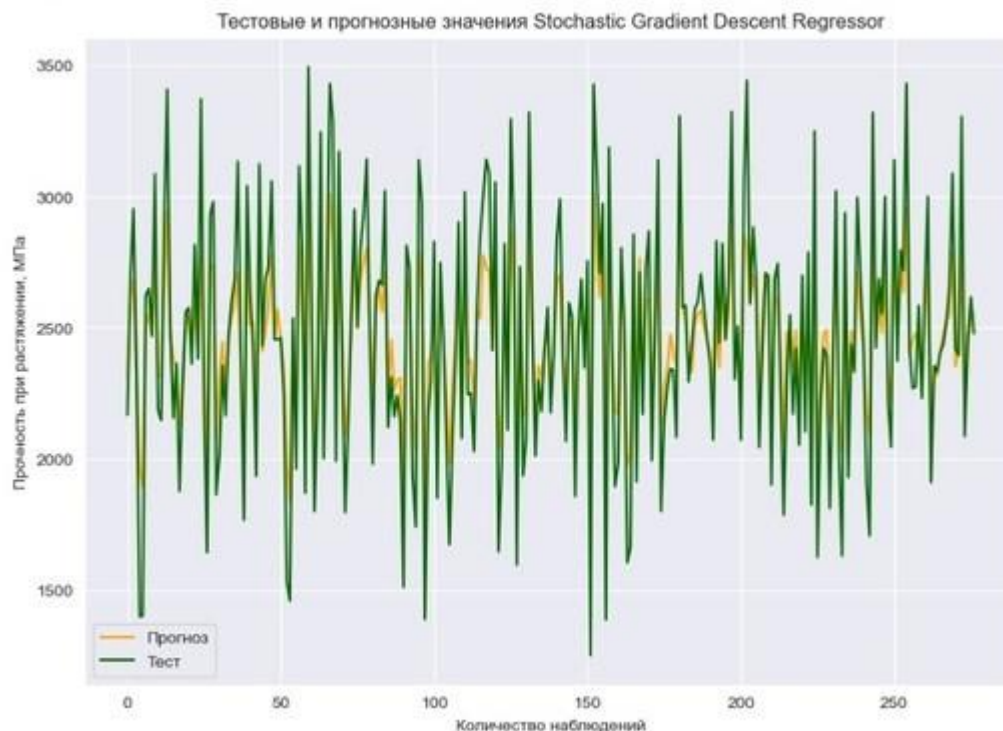


Рисунок 10-график стохастического градиентного спуска для модуля упругости

Многослойный перцептрон (MLPRegressor) — это алгоритм обучения с учителем, который изучает функцию  $f(\cdot): R_m \rightarrow R_o$  обучением на наборе данных, где  $m$  — количество измерений для ввода и  $o$  — количество размеров для вывода. Это искусственная нейронная сеть, имеющая 3 или более слоёв перцептронов. Эти слои - один входной слой, 1 или более скрытых слоёв и один выходной слой перцептронов.

Достоинства метода: построение сложных разделяющих поверхностей; возможность осуществления любого отображения входных векторов в выходные; легко обобщает входные данные; не требует распределения входных векторов; изучает нелинейные модели.

Недостатки метода: имеет невыпуклую функцию потерь; разные инициализации случайных весов могут привести к разной точности проверки; требует настройки ряда гиперпараметров; чувствителен к масштабированию функций.

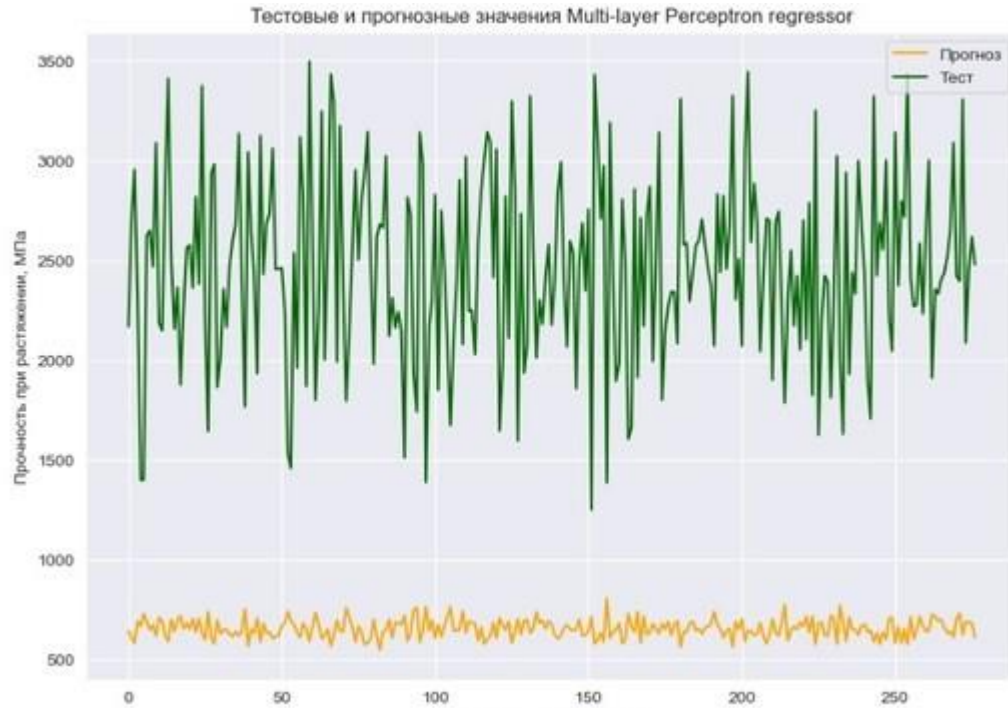


Рисунок 11-график многослойного персептрона для модуля упругости, ГПа

Лассо регрессия (Lasso) — 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.

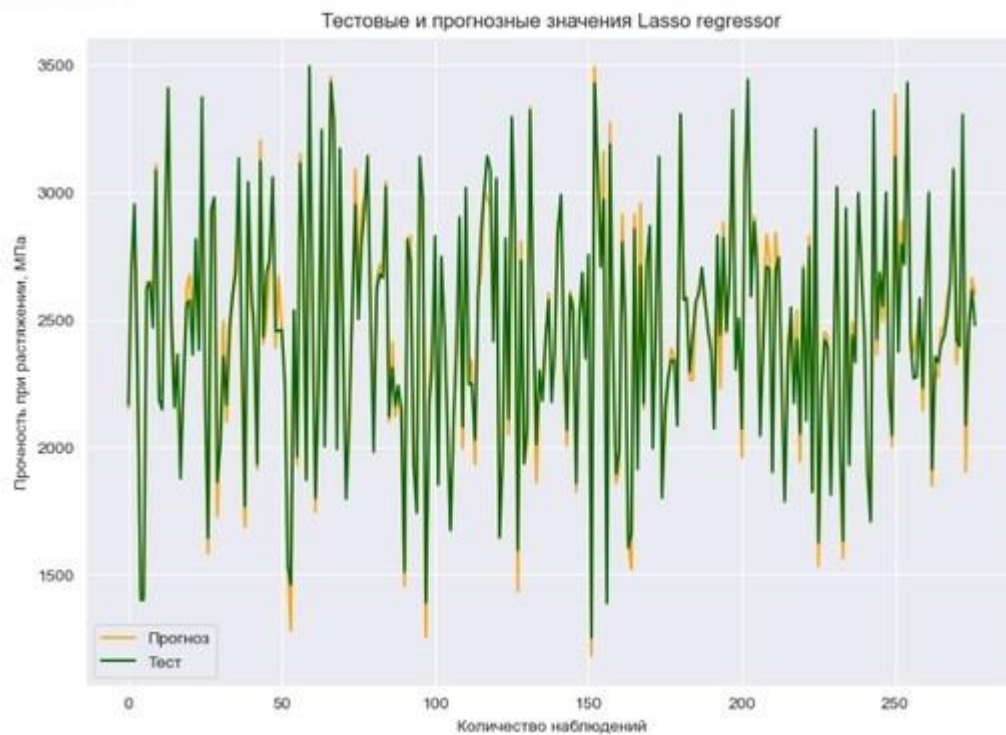


Рисунок 12 - график метода Лассо для модуля упругости, ГПа

Немного расскажем об используемых метриках качества моделей: R2 или коэффициент детерминации измеряет долю дисперсии, объяснённую моделью, в общей дисперсии целевой переменной.

```
#Выводим гиперпараметры для оптимальной модели
print(grid.best_estimator_)
knr_upr = grid.best_estimator_
print(f'R2-score RFR для прочности при растяжении, МПа: {knr_upr.score(x_test_1, y_test_1).round(3)}')

RandomForestRegressor(criterion='mse', max_depth=15, n_estimators=200,
                       random_state=33)
R2-score RFR для прочности при растяжении, МПа: 0.963
```

Рисунок 13 – часть кода для результата метрики R2 для метода «Случайный лес»

Если он близок к единице, то модель хорошо объясняет данные, если же он близок к нулю, то качество прогноза идентично средней величине целевой переменной (т.е. очень низкое). Отрицательные значения коэффициента детерминации означают плохую объясняющую способность модели.

MSE (Mean Squared Error) или средняя квадратичная ошибка принимает значения в тех же единицах, что и целевая переменная. Чем ближе к нулю MSE, тем лучше работают предсказательные качества модели.

```
svr2 = make_pipeline(StandardScaler(), SVR(kernel = 'rbf', C = 500.0, epsilon = 1.0))
#обучаем модель
svr2.fit(x_train_2, np.ravel(y_train_2))
#вычисляем коэффициент детерминации
y_pred_svr2 = svr2.predict(x_test_2)
mae_svr2 = mean_absolute_error(y_pred_svr2, y_test_2)
mse_svr_elast2 = mean_squared_error(y_test_2, y_pred_svr2)
print('Support Vector Regression Results Train:')
print("Test score: {:.2f}".format(svr2.score(x_train_2, y_train_2))) # Скор для тренировочной выборки
print('Support Vector Regression Results:')
print('SVR_MAE:', round(mean_absolute_error(y_test_2, y_pred_svr2)))
print('SVR_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test_2, y_pred_svr2)))
print('SVR_MSE: {:.2f}'.format(mse_svr_elast2))
print('SVR_RMSE: {:.2f}'.format(np.sqrt(mse_svr_elast2)))
print("Test score: {:.2f}".format(svr2.score(x_test_2, y_test_2))) # Скор для тестовой выборки

Support Vector Regression Results Train:
Test score: 0.90
Support Vector Regression Results:
SVR_MAE: 3
SVR_MAPE: 0.05
SVR_MSE: 18.38
SVR_RMSE: 4.29
Test score: -0.85
```

Рисунок 14 - код для вывода различных метрик для метода опорных векторов

### 1.3. Разведочный анализ данных

Прежде чем передать данные в работу моделей машинного обучения, необходимо обработать и очистить их. Очевидно, что «грязные» и необработанные данные могут содержать искажения и пропущенные значения – это ненадёжно, поскольку способно привести к крайне неверным результатам по итогам моделирования. Но безосновательно удалять что-либо тоже неправильно. Именно поэтому сначала датасет надо изучить.

```
In [23]: a = df.describe()
a.T
```

	count	mean	std	min	25%	50%	75%	max
<b>Соотношение матрица-наполнитель</b>	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
<b>Плотность, кг/м3</b>	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
<b>модуль упругости, ГПа</b>	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
<b>Количество отвердителя, м.%</b>	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
<b>Содержание эпоксидных групп,%_2</b>	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
<b>Температура вспышки, С_2</b>	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
<b>Поверхностная плотность, г/м2</b>	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
<b>Модуль упругости при растяжении, ГПа</b>	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
<b>Прочность при растяжении, МПа</b>	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
<b>Потребление смолы, г/м2</b>	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
<b>Угол нашивки</b>	1023.0	0.491691	0.500175	0.000000	0.000000	0.000000	1.000000	1.000000
<b>Шаг нашивки</b>	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
<b>Плотность нашивки</b>	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 15 - описательная статистика датасета

Цель разведочного анализа - получение первоначальных представлений о характерах распределений переменных исходного набора данных, формирование оценки качества исходных данных (наличие пропусков, выбросов), выявление характера взаимосвязи между переменными с целью последующего выдвижения гипотез о наиболее подходящих для решения задачи моделях машинного обучения.

```
# Проверим датасет на дубликаты
df.duplicated().sum()
#Дубликатов нет
```

0

Рисунок 16 - проверка датасета на наличие дубликатов

В качестве инструментов разведочного анализа используется: оценка статистических характеристик датасета; гистограммы распределения каждой из переменной (несколько различных вариантов); диаграммы ящика с усами (несколько интерактивных вариантов); попарные графики рассеяния точек (несколько вариантов); график «квантиль-квантиль»; тепловая карта (несколько вариантов); описательная статистика для каждой переменной; анализ и полное исключение выбросов (5

повторных итераций); проверка наличия пропусков и дубликатов; ранговая корреляция Кендалла и Пирсона.

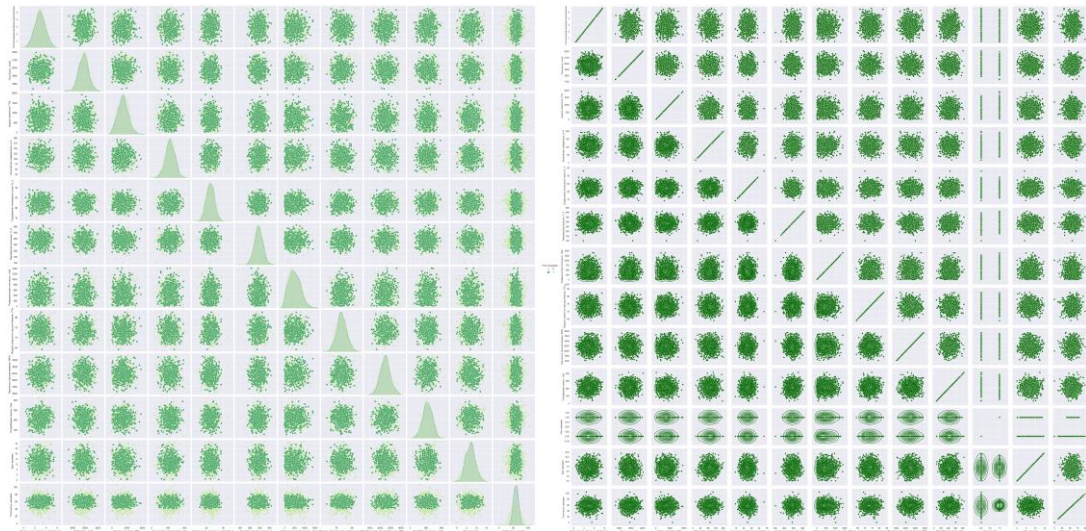


Рисунок 17 - попарные графики рассеяния точек (два разных варианта)

Гистограммы переменных

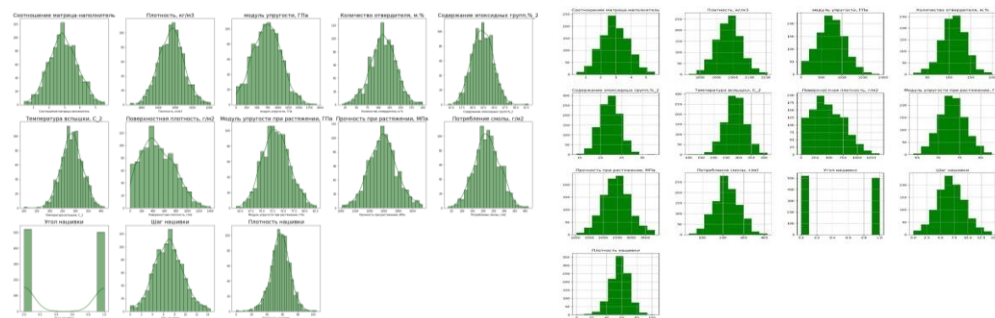


Рисунок 18 - гистограммы распределения (2 разных варианта)

Гистограммы используются для изучения распределений частот значений переменных. Мы видим очень слабую корреляцию между переменными.

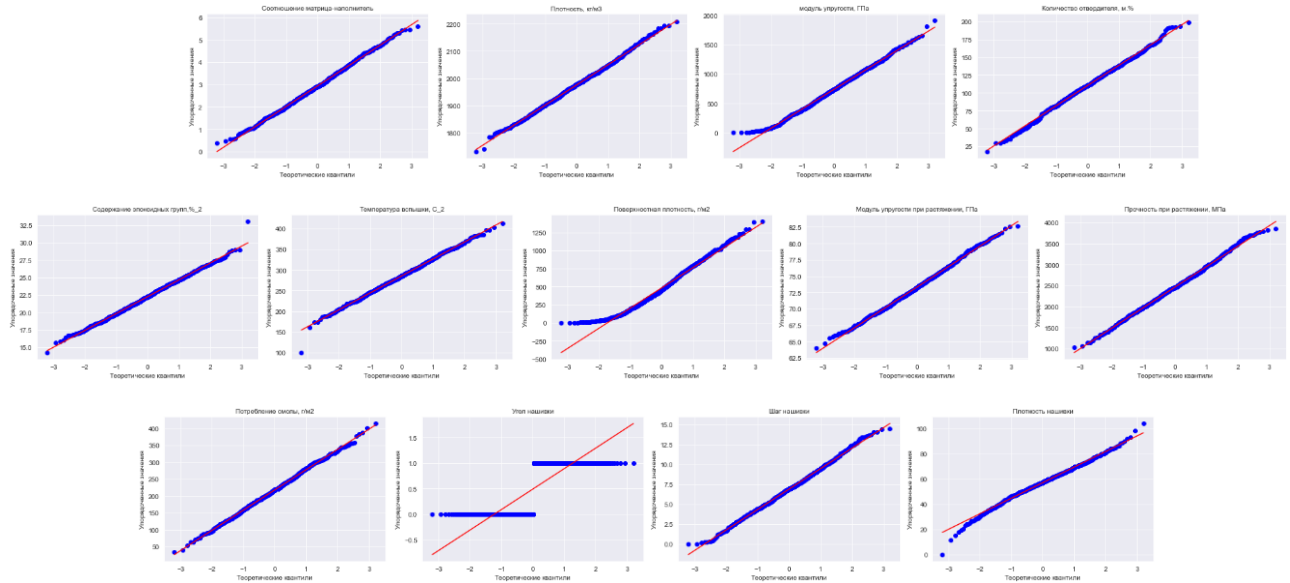


Рисунок 19 - графики «квантиль-квантиль»

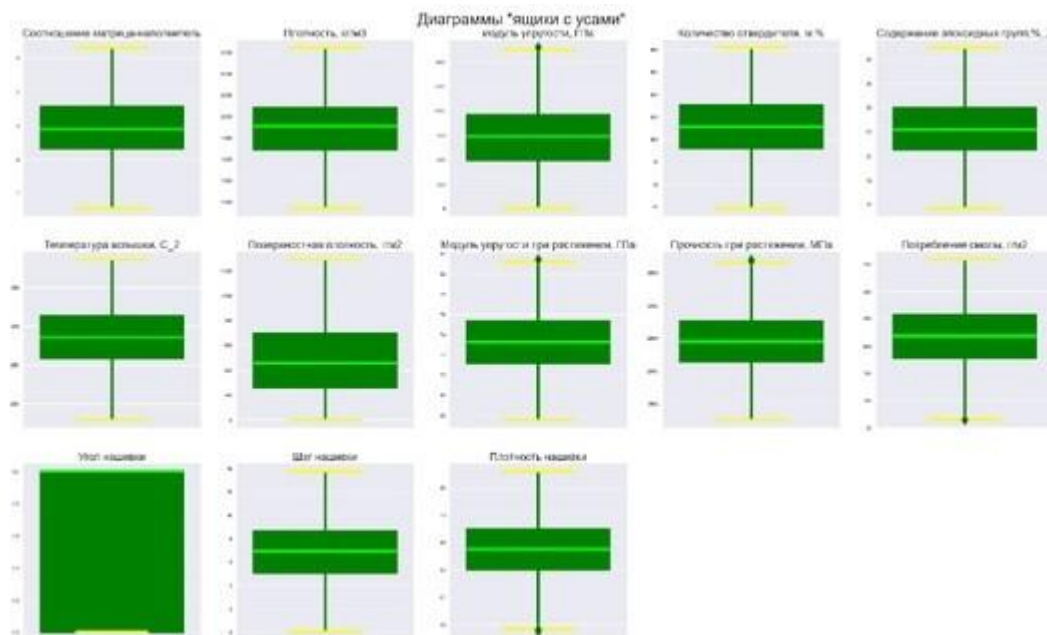


Рисунок 20 - график "ящиков с усами" для всех переменных

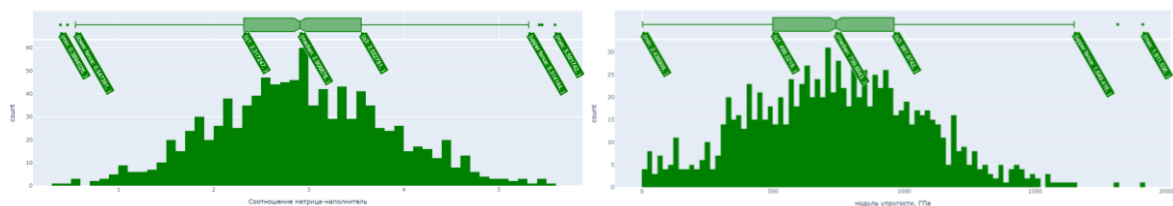


Рисунок 21 - интерактивный график "ящиков с усами" и гистограммы распределения на примере двух переменных



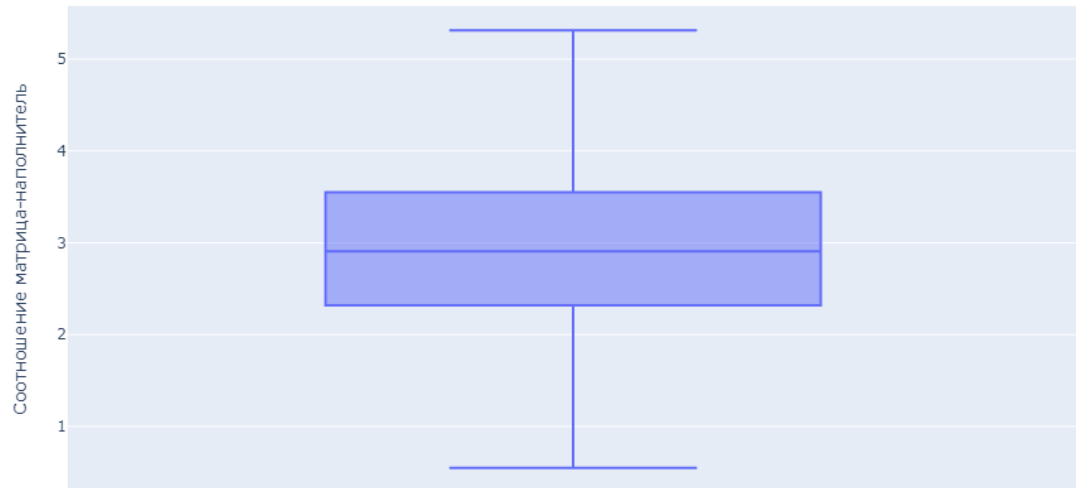


Рисунок 22 - график "ящиков с усами" для одной переменной

После обнаружения выбросов данные, значительно отличающиеся от выборки, будут полностью удалены. Для расчёта этих данных мы будем использовать методы трех сигм и межквартильного расстояния.

Данные объединённого датасета не имеют чётко выраженной зависимости, что подтверждает тепловая карта с матрицей корреляции и матрицы диаграмм рассеяния.

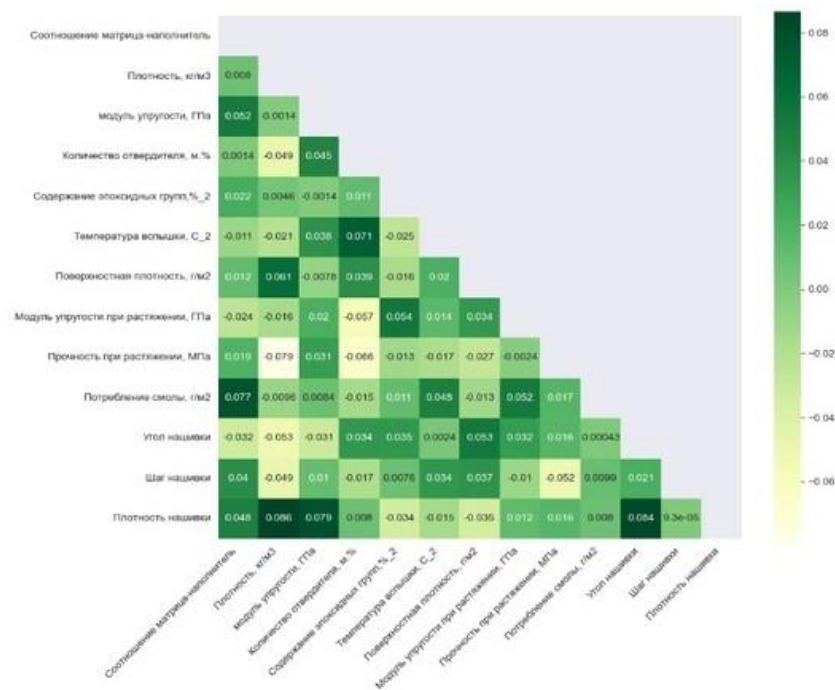


Рисунок 23 - тепловая карта с корреляцией данных

Максимальная корреляция между плотностью нашивки и углом нашивки 0.08, значит нет зависимости между этими данными. Корреляция между всеми параметрами очень близка к 0, корреляционные связи между переменными не наблюдаются.

## 2. Практическая часть

### 2.1. Предобработка данных

В ходе проведённого анализа принимаем решение столбец "Угол нашивки" привести к виду «0» и «1».

```
#Поиск уникальных значений с помощью функции nunique
df.nunique()
#Видим в основном общее число уникальных значений в каждом столбце, но в столбце "Угол нашивки" всего 2 значения. Поработаем с ним.
```

Соотношение матрица-наполнитель	1014
Плотность, кг/м3	1013
модуль упругости, ГПа	1020
Количество отвердителя, м.%	1005
Содержание эпоксидных групп, %_2	1004
Температура вспышки, C_2	1003
Поверхностная плотность, г/м2	1004
Модуль упругости при растяжении, ГПа	1004
Прочность при растяжении, МПа	1004
Потребление смолы, г/м2	1003
Угол нашивки, град	2
Шаг нашивки	989
Плотность нашивки	988
dtype:	int64

```
# Поработаем со столбцом "Угол нашивки"

df['Угол нашивки, град'].nunique()
#Так как кол-во уникальных значений в колонке Угол нашивки равно 2, можем привести данные в этой колонке к значениям 0 и 1
2

#Проверим кол-во элементов, где Угол нашивки равен 0 градусов
df['Угол нашивки, град'][df['Угол нашивки, град'] == 0.0].count()

520

# Приведем столбец "Угол нашивки" к значениям 0 и 1 и integer
df = df.replace({'Угол нашивки, град': {0.0 : 0, 90.0 : 1}})
df['Угол нашивки, град'] = df['Угол нашивки, град'].astype(int)

#Переименуем столбец
df = df.rename(columns={'Угол нашивки, град' : 'Угол нашивки'})
df
```

Рисунок 24 - часть кода с преобразованием столбца "Угол нашивки"

По условиям задания нормализуем значения. Для этого применим MinMaxScaler(), затем применим Normalizer(). Второе даёт нам больше выбросов.

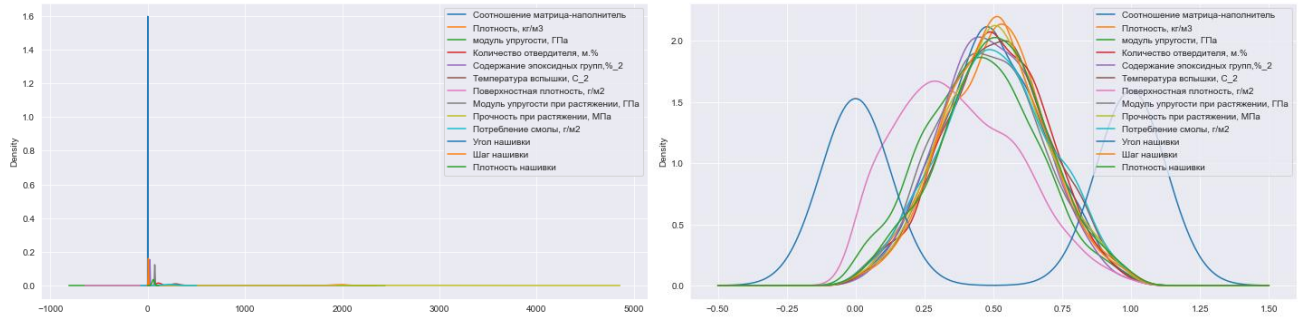


Рисунок 25 - визуализированные данные до и после нормализации

## 2.2. Разработка и обучение модели

Разработка и обучение моделей машинного обучения осуществлялась для двух выходных параметров: «Прочность при растяжении» и «Модуль упругости при растяжении» отдельно. Для решения применим все методы, описанные выше.

```
Ввод [152]: parameters = { 'n_estimators': [200, 300],
                          'max_depth': [9, 15],
                          'max_features': ['auto'],
                          'criterion': ['mse'] }
grid = GridSearchCV(estimator = rfr, param_grid = parameters, cv = 10)
grid.fit(x_train_1, y_train_1)

Out[152]: GridSearchCV(cv=10,
                      estimator=RandomForestRegressor(max_depth=7, n_estimators=15,
                                                         random_state=33),
                      param_grid={'criterion': ['mse'], 'max_depth': [9, 15],
                                   'max_features': ['auto'], 'n_estimators': [200, 300]})

Ввод [153]: grid.best_params_

Out[153]: {'criterion': 'mse',
           'max_depth': 15,
           'max_features': 'auto',
           'n_estimators': 200}

Ввод [154]: #выводим гиперпараметры для оптимальной модели
print(grid.best_estimator_)
knr_upr = grid.best_estimator_
print(f'R2-score RFR для прочности при растяжении, МПа: {knr_upr.score(x_test_1, y_test_1).round(3)}')

RandomForestRegressor(criterion='mse', max_depth=15, n_estimators=200,
                      random_state=33)
R2-score RFR для прочности при растяжении, МПа: 0.963

Ввод [155]: #подставляем оптимальные гиперпараметры в нашу модель случайного леса
rfr_grid = RandomForestRegressor(n_estimators = 200, criterion = 'mse', max_depth = 15, max_features = 'auto')
#обучаем модель
rfr_grid.fit(x_train_1, y_train_1)

predictions_rfr_grid = rfr_grid.predict(x_test_1)
#Оцениваем точность на тестовом наборе
mae_rfr_grid = mean_absolute_error(predictions_rfr_grid, y_test_1)
mae_rfr_grid

Out[155]: 68.58092215839413

Ввод [156]: new_row_in_mae_df = {'Perpeccop': 'RandomForest_GridSearchCV', 'MAE': mae_rfr_grid}
mae_df = mae_df.append(new_row_in_mae_df, ignore_index=True)
```

Рисунок 26- поиск гиперпараметров



Порядок разработки модели для каждого параметра и для каждого выбранного метода можно разделить на следующие этапы: разделение нормализованных данных на обучающую и тестовую выборки (в соотношении 70 на 30%); проверка моделей при стандартных значениях; сравнение с результатами модели, выдающей среднее значение; создание графика; сравнение моделей по метрике MAE; поиск сетки гиперпараметров, по которым будет происходить оптимизация модели. В качестве параметра оценки выбран коэффициент детерминации ( $R^2$ ); оптимизация подбора гиперпараметров модели с помощью выбора по сетке и перекрёстной проверки; подстановка оптимальных гиперпараметров в модель и обучение модели на тренировочных данных; оценка полученных данных; сравнение со стандартными значениями.

```
Наилучшие параметры:  
{ 'preprocessing': MinMaxScaler(), 'regressor': Lasso(alpha=0.1) }  
  
Наилучшее значение правильности перекрестной проверки: 0.97  
Правильность на тестовом наборе: 0.97
```

Рисунок 27 - наилучшие гиперпараметры

Модель после настройки гиперпараметров показала результат немного лучше. Однако, ниже, чем базовая модель. Прочность при растяжении и модуль упругости не имеет линейной зависимости. Все использованные модели не справились с задачей. Результат неудовлетворительный. Свойства композитных материалов в первую очередь зависят от используемых материалов.

### 2.3. Тестирование модели

После обучения моделей была проведена оценка точности этих моделей на обучающей и тестовых выборках.

	<b>Перепеccep</b>	<b>MAE</b>
<b>0</b>	Support Vector	78.477914
<b>1</b>	RandomForest	76.589025
<b>2</b>	Linear Regression	61.986894
<b>3</b>	GradientBoosting	64.728717
<b>4</b>	KNeighbors	102.030259
<b>5</b>	DecisionTree	107.158013
<b>6</b>	SGD	181.624450
<b>7</b>	MLP	1808.547264
<b>8</b>	Lasso	69.474334
<b>9</b>	RandomForest_GridSearchCV	67.603567
<b>10</b>	KNeighbors_GridSearchCV	99.281694
<b>11</b>	DecisionTree_GridSearchCV	168.624997

Рисунок 28 - результат оценки точности по MAE

В целом при таких результатах можно применять среднее значение переменной в качестве прогнозного.

#### **2.4. Написать нейронную сеть, которая будет рекомендовать соотношение «матрица – наполнитель».**

Обучение нейронной сети— это такой процесс, при котором происходит подбор оптимальных параметров модели, с точки зрения минимизации функционала ошибки. Начнём строить нейронную сеть с помощью класса `keras.Sequential`.

```
# Сформируем входы и выход для модели

tv = df['Соотношение матрица-наполнитель']
tr_v = df.loc[:, df.columns != 'Соотношение матрица-наполнитель']

# Разбиваем выборки на обучающую и тестовую
x_train, x_test, y_train, y_test = train_test_split(tr_v, tv, test_size = 0.3, random_state = 14)

# Нормализуем данные

x_train_n = tf.keras.layers.Normalization(axis = -1)
x_train_n.adapt(np.array(x_train))
```

Рисунок 29 - создание нейронной сети

Определим параметры, поищем оптимальные параметры, посмотрим на результаты. С помощью KerasClassifier выйдем на наилучшие параметры для нашей нейронной сети и построим окончательную нейросеть.

```
Ввод [67]: # построение окончательной модели
model = create_model(layers=[128, 64, 16, 3], dr=0.05)
print(model.summary())

Model: "sequential_195"

```

Layer (type)	Output Shape	Param #
dense_493 (Dense)	(None, 128)	1792
dense_494 (Dense)	(None, 64)	8256
dense_495 (Dense)	(None, 16)	1040
dense_496 (Dense)	(None, 3)	51
dropout_195 (Dropout)	(None, 3)	0
dense_497 (Dense)	(None, 3)	12

```

Total params: 11,151
Trainable params: 11,151
Non-trainable params: 0
None

```

Рисунок 30 - построение первой нейросети

Обучим и оценим модель, посмотрим на потери, зададим функцию для визуализации факт/прогноз для результатов моделей.

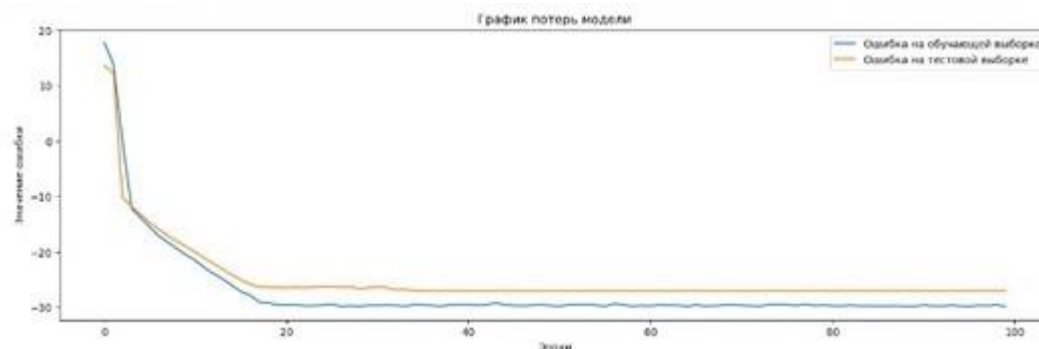


Рисунок 31 - график потерь модели 1

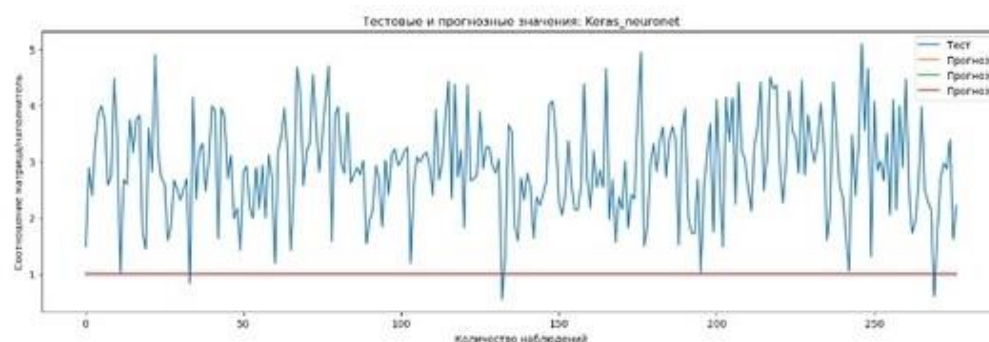


Рисунок 32 - тестовые и прогнозные значения модели 1

Не удовлетворившись таким результатом, создадим другую простую модель глубокого обучения с другой архитектурой. Обучим её, посмотрим на потери, оценим MSE, построим график.

#### 8.12. Сконфигурируем другую модель, зададим слои

```

from tensorflow.keras.layers import Dense

model1 = tf.keras.Sequential([x_train_n, Dense(128, activation='relu'),
                               Dense(128, activation='relu'),
                               Dense(128, activation='relu'),
                               Dense(64, activation='relu'),
                               Dense(64, activation='relu'),
                               Dense(32, activation='relu'),
                               Dense(16, activation='relu'),
                               Dense(1)

                               ])

model1.compile(optimizer = tf.keras.optimizers.Adam(0.001), loss = 'mean_squared_error', metrics = [tf.keras.metrics.RootMeanSqu
# Посмотрим на архитектуру модели
model1.summary()
    
```

Рисунок 33 - создание второй модели

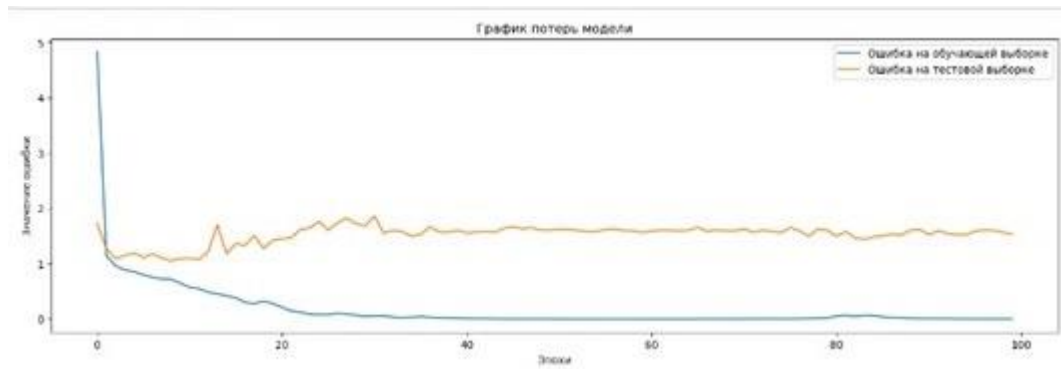


Рисунок 34 - график потерь второй модели

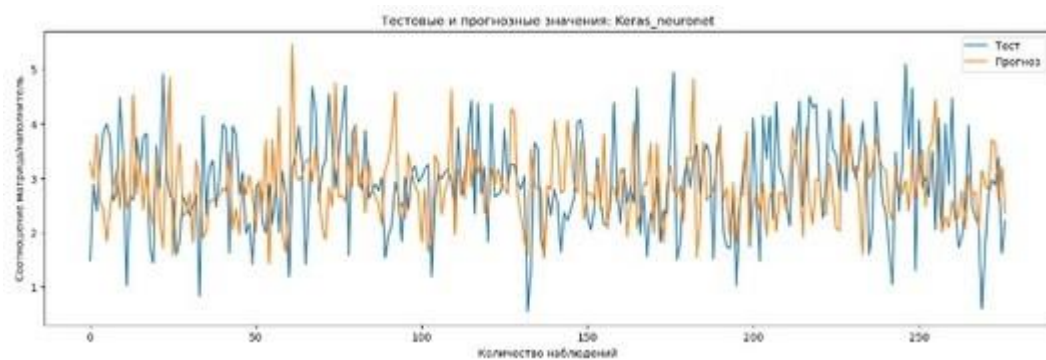


Рисунок 35- тестовые и прогнозные значения второй модели

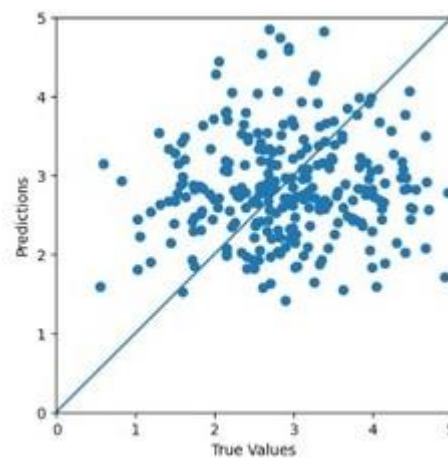


Рисунок 36- график прогнозных и настоящих значений.

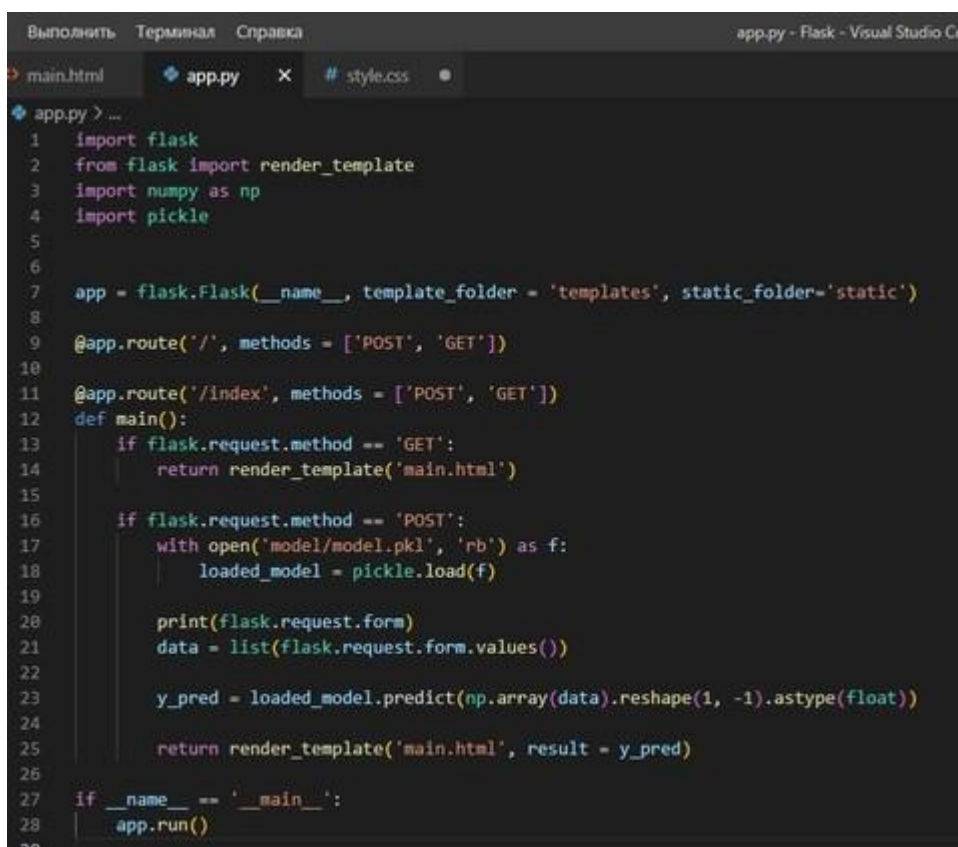
### Разработка приложения

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

Модуль упругости при растяжении, ГПа  
[72.16438914]

Рисунок 37 - пример результата работы приложения

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



```
app.py > ...
1  import flask
2  from flask import render_template
3  import numpy as np
4  import pickle
5
6
7  app = flask.Flask(__name__, template_folder = 'templates', static_folder='static')
8
9  @app.route('/', methods = ['POST', 'GET'])
10
11  @app.route('/index', methods = ['POST', 'GET'])
12  def main():
13      if flask.request.method == 'GET':
14          return render_template('main.html')
15
16      if flask.request.method == 'POST':
17          with open('model/model.pkl', 'rb') as f:
18              loaded_model = pickle.load(f)
19
20          print(flask.request.form)
21          data = list(flask.request.form.values())
22
23          y_pred = loaded_model.predict(np.array(data).reshape(1, -1).astype(float))
24
25          return render_template('main.html', result = y_pred)
26
27  if __name__ == '__main__':
28      app.run()
29
```

Рисунок 38 - часть кода приложения

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

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

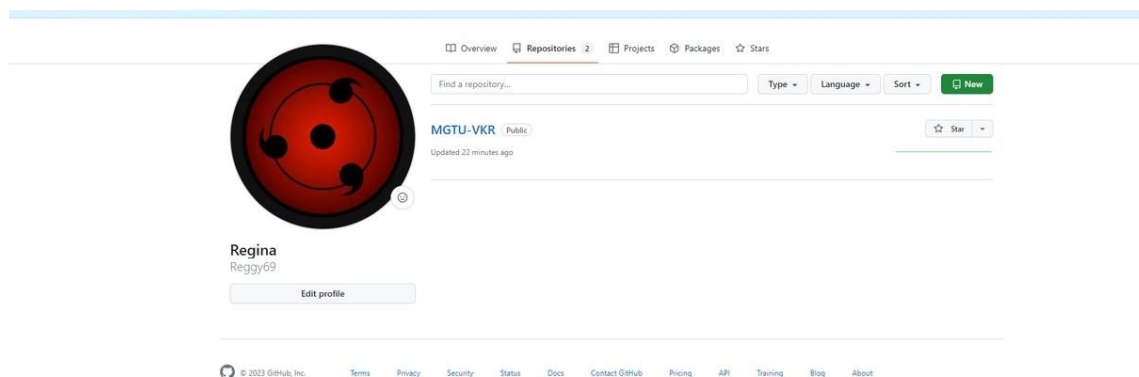
Соотношение матрица-наполнитель	<input type="text"/>
Плотность, кг/м <sup>3</sup>	<input type="text"/>
модуль упругости, ГПа	<input type="text"/>
Количество отвердителя, м. %	<input type="text"/>
Содержание эпоксидных групп, % 2	<input type="text"/>
Температура вспышки, С 2	<input type="text"/>
Поверхностная плотность, г/м <sup>2</sup>	<input type="text"/>
Прочность при растяжении, МПа	<input type="text"/>
Потребление смолы, г/м <sup>2</sup>	<input type="text"/>
Угол нашивки	<input type="text"/>
Шаг нашивки	<input type="text"/>
Плотность нашивки	<input type="text"/>
<input type="button" value="Отправить"/>	

Рисунок 39- скриншот пользовательского приложения

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

## 2.5. Создание удалённого репозитория и загрузка

Репозиторий был создан на [github.com](https://github.com) по адресу:  
<https://github.com/Reggy69/MGTU-VKR>



## 2.6. Заключение

Данная исследовательская работа позволяет сделать некоторые основные выводы по теме. Распределение полученных данных в объединённом датасете близко к нормальному, но коэффициенты корреляции между парами признаков стремятся к нулю. Используемые при разработке моделей подходы не позволили получить сколько-нибудь достоверных прогнозов. Применённые модели регрессии не показали высокой эффективности в прогнозировании свойств композитов. Лучшие метрики для модуля упругости при растяжении, ГПа – лассо-регрессия, для прочности при растяжении, МПа – метод опорных векторов.

Был сделан вывод, что невозможно определить из свойств материалов соотношение «матрица – наполнитель». 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.

Необходимы дополнительные вводные данные, получение новых результирующих признаков в результате математических преобразований, релевантных доменной области, консультации экспертов предметной области, новые исследования, работа эффективной команды, состоящей из различных учёных.



В целом 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. Отсюда, также учитывая отсутствие корреляции между признаками, делаем вывод, что текущим набором алгоритмов задача не решается, возможно, решается трудно или не решается совсем.

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

11. Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html#user>. (дата обращения: 03.06.2022).

12. Документация по библиотеке pandas: – Режим доступа: [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide). (дата обращения: 04.06.2022).

13. Документация по библиотеке scikit-learn: – Режим доступа: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html). (дата обращения: 05.06.2022).

14. Документация по библиотеке seaborn: – Режим доступа: <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 доступа: <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 машинное 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)

## 2.8. Приложение 1

Подробный план работы:

1. Загружаем и обрабатываем входящие датасеты
  - 1.1. Удаляем неинформативные столбцы
  - 1.2. Объединяем датасеты по методу INNER
2. Проводим разведочный анализ данных:
  - 2.1. Данные в столбце "Угол нашивки» приведём к 0 и 1
  - 2.2. Изучим описательную статистику каждой переменной - среднее, медиана, стандартное отклонение, минимум, максимум, квартили
  - 2.3. Проверим датасет на пропуски и дубликаты данных
  - 2.4. Получим среднее, медианное значение для каждой колонки (по заданию необходимо получить их отдельно, поэтому продублируем их только отдельно)
  - 2.5. Вычислим коэффициенты ранговой корреляции Кендалла
  - 2.6. Вычислим коэффициенты корреляции Пирсона
3. Визуализируем наш разведочный анализ сырых данных (до выбросов и нормализации)

- 3.1. Построим несколько вариантов гистограмм распределения каждой переменной
- 3.2. Построим несколько вариантов диаграмм ящиков с усами каждой переменной
- 3.3. Построим гистограмму распределения и диаграмма "ящик с усами" одновременно вместе с данными по каждому столбцу
- 3.4. Построим несколько вариантов попарных графиков рассеяния точек (матрицы диаграмм рассеяния)
- 3.5. Построим графики квантиль-квантиль
- 3.6. Построим корреляционную матрицу с помощью тепловой карты
4. Проведём предобработку данных (в данном пункте только очистка датасета от выбросов)
  - 4.1. Проверим выбросы по 2 методам: 3-х сигм или межквартильных расстояний
  - 4.2. Посчитаем распределение выбросов по каждому столбцу (с целью предотвращения удаления особенностей признака или допущения ошибки)
  - 4.3. Исключим выбросы методом межквартильного расстояния
  - 4.4. Удалим строки с выбросами
  - 4.5. Визуализируем датасет без выбросов, и убедимся, что выбросы еще есть.
  - 4.6. Для полной очистки датасета от выбросов повторим пункты (4.3 – 4.5) ещё 3 раза.
  - 4.7. Сохраняем идеальный, без выбросов датасет
  - 4.8. Изучим чистые данные по всем параметрам
  - 4.9. Визуализируем «чистый» датасет (без выбросов)
5. Проведём нормализацию и стандартизацию (продолжим предобработку данных)
  - 5.1. Визуализируем плотность ядра
  - 5.2. Нормализуем данные с помощью MinMaxScaler()
  - 5.3. Нормализуем данные с помощью Normalizer()
  - 5.4. Сравним с данными до нормализации

- 5.5. Проверим перевод данных из нормализованных в исходные
- 5.6. Рассмотрим несколько вариантов корреляции между параметрами после нормализации
- 5.7. Стандартизируем данные
- 5.8. Визуализируем данные корреляции
- 5.9. Посмотрим на описательную статистику после нормализации и после стандартизации
- 6. Разработаем и обучим нескольких моделей прогноза прочности при растяжении (с 30% тестовой выборки)
  - 6.1. Определим входы и выходы для моделей
  - 6.2. Разобьём данные на обучающую и тестовую выборки
  - 6.3. Проверим правильность разбивки
  - 6.4. Построим модели и найдём лучшие гиперпараметры (задача по заданию):
  - 6.5. Построим и визуализируем результат работы метода опорных векторов
  - 6.6. Построим и визуализируем результат работы метода случайного леса
  - 6.7. Построим и визуализируем результат работы линейной регрессии
  - 6.8. Построим и визуализируем результат работы метода градиентного бустинга
  - 6.9. Построим и визуализируем результат работы метода К ближайших соседей
  - 6.10. Построим и визуализируем результат работы метода дерева решений
  - 6.11. Построим и визуализируем результат работы стохастического градиентного спуска
  - 6.12. Построим и визуализируем результат работы многослойного перцептрона
  - 6.13. Построим и визуализируем результат работы лассо регрессии
  - 6.14. Сравним наши модели по метрике MAE
  - 6.15. Найдём лучшие гиперпараметры для случайного леса
  - 6.16. Подставим значения в нашу модель случайного леса
  - 6.17. Найдём лучшие гиперпараметры для К ближайших соседей

- 6.18. Подставим значения в нашу модель K ближайших соседей
  - 6.19. Найдём лучшие гиперпараметры метода дерева решений
  - 6.20. Подставим значения в нашу модель метода дерева решений
  - 6.21. Проверим все модели и процессинги и выведем лучшую модель и процессинг
7. Разработаем и обучим нескольких моделей прогноза модуля упругости при растяжении (с 30% тестовой выборки)
- 7.1. Определим входы и выходы для моделей
  - 7.2. Разобьём данные на обучающую и тестовую выборки
  - 7.3. Проверим правильность разбивки
  - 7.4. Построим модели и найдём лучшие гиперпараметры (задача по заданию):
  - 7.5. Построим и визуализируем результат работы метода опорных векторов
  - 7.6. Построим и визуализируем результат работы метода случайного леса
  - 7.7. Построим и визуализируем результат работы линейной регрессии
  - 7.8. Построим и визуализируем результат работы метода градиентного бустинга
  - 7.9. Построим и визуализируем результат работы метода K ближайших соседей
  - 7.10. Построим и визуализируем результат работы метода дерева решений
  - 7.11. Построим и визуализируем результат работы стохастического градиентного спуска
  - 7.12. Построим и визуализируем результат работы многослойного перцептрона
  - 7.13. Построим и визуализируем результат работы лассо регрессии
  - 7.14. Сравним наши модели по метрике MAE
  - 7.15. Найдём лучшие гиперпараметры для случайного леса
  - 7.16. Подставим значения в нашу модель случайного леса
  - 7.17. Найдём лучшие гиперпараметры для K ближайших соседей
  - 7.18. Подставим значения в нашу модель K ближайших соседей
  - 7.19. Найдём лучшие гиперпараметры метода дерева решений

- 7.20. Подставим значения в нашу модель метода дерева решений
- 7.21. Проверим все модели и процессинги и выведем лучшую модель и процессинг
- 8. Нейронная сеть для рекомендации соотношения матрица-наполнитель
  - 8.1. Сформируем входы и выход для модели
  - 8.2. Нормализуем данные
  - 8.3. Построим модель, определим параметры
  - 8.4. Найдем оптимальные параметры для модели
  - 8.5. Посмотрим на результаты
  - 8.6. Повторим шаги 8.4 – 8.5 до построения окончательной модели
  - 8.7. Обучим нейросеть 80/20
  - 8.8. Оценим модель
  - 8.9. Посмотрим на потери модели
  - 8.10. Посмотрим на график результата работы модели
  - 8.11. Посмотрим на график потерь на тренировочной и тестовой выборках
  - 8.12. Сконфигурируем другую модель, зададим слои
  - 8.13. Посмотрим на архитектуру другой модели
  - 8.14. Обучим другую модель
  - 8.15. Посмотрим на потери другой модели
  - 8.16. Посмотрим на график потерь на тренировочной и тестовой выборках
  - 8.17. Зададим функцию для визуализации факт/прогноз для результатов моделей
  - 8.18. Посмотрим на график результата работы модели
  - 8.19. Оценим модель MSE
  - 8.20. Сохраняем вторую модель для разработки веб-приложения для прогнозирования соотношения "матрица-наполнитель" в фреймворке Flask
- 9. Создаём приложение
  - 9.1. Импортируем необходимые библиотеки



- 9.2. Загрузим модель и определим параметры функции
- 9.3. Получим данные из наших форм и положим их в список
- 9.4. Укажем шаблон и прототип сайта для вывода
- 9.5. Запустим приложение
- 9.6. Откроем <http://127.0.0.1:5000/>
- 10. Создание удалённого репозитория и загрузка результатов работы на него.
  - 10.1. <https://github.com/Reggy69/MGTU-VKR>
  - 10.2. Создадим README (<https://github.com/Reggy69/MGTU-VKR/blob/main/README.md>)
  - 10.3. Выгрузим все необходимые файлы в репозиторий