|  |
| --- |
| Москва 2022 |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**по курсу**

**«Data Science»**

**Прогнозирование конечных свойств новых материалов**

**(композиционных материалов)**

Слушатель Певневец Святослав Олегович

**Содержание**

Введение 3

1. Аналитическая часть 4

1.1 Постановка задачи 4

1.2 Описание используемых методов. 5

1.3 Разведочный анализ данных. 6

2. Практическая часть 7

2.1 Предобработка данных 7

2.2 Разработка и обучение моделей 14

2.3 Написание нейронной сети 21

3. Создание удаленного репозитория 27

4. Заключение 28

5. Список используемой литературы 29

**Введение**.

Целью  выпускной квалификационной работы является закрепление теоретических и практических материалов пройденных по курсу Data Science и посвящена созданию и обучению моделей по прогнозированию конечных свойств новых 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.

Основная задача изготовления композиционного материала – разработать новое вещество, которое сочетает в себе характеристики ее составляющих наиболее полезным способом.

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).

Физические испытания по созданию новых композиционных материалов как правило являются достаточно затратными в сравнении с экспериментальными.

В связи с этим разработка моделей машинного обучения для прогнозирования новых композиционных материалов является на текущий момент актуальной.

1. **Аналитическая часть*.***

**1.1 Постановка задачи**

Основной задачей настоящей работы является построение при помощи методов машинного обучения моделей прогнозирования «модуля упругости при растяжении» и «прочности при растяжении», написание нейронной сети, которая будет рекомендовать «соотношение матрица-наполнитель» конечных свойств нового композиционного материала.

Исходные данные предоставлены структурным подразделением МГТУ им. Н.Э. Баумана – Центр НТИ «Цифровое материаловедение: новые материалы и вещества».

Данные предоставлены в виде excel-таблиц:

Таблица «X\_bp.xlsx» содержит 1023 строк и 10 столбца с данными;

Таблица «X\_nup.xlsx» содержит 1040 строк и 3 столбца с данными.

Входных параметров 10:

|  |
| --- |
| * Плотность, кг/м3 |
| * модуль упругости, ГПа |
| * Количество отвердителя, м.% |
| * Содержание эпоксидных групп,%\_2 |
| * Температура вспышки, С\_2 |
| * Поверхностная плотность, г/м2 |
| * Потребление смолы, г/м2 |
| * Угол нашивки, град |
| * Шаг нашивки |
| * Плотность нашивки |

Выходных параметров 3:

|  |
| --- |
| * Соотношение матрица-наполнитель |
| * Модуль упругости при растяжении, ГПа |
| * Прочность при растяжении, МПа |

Анализ и обработку исходных данных будем проводить с помощью языка программирования Python с использованием библиотек Pandas, Matplotlib, Sklearn, TensorFlow и др.

**1.2 Описание используемых методов.**

Для решения поставленной задачи использованы следующие методы:

**Линейная регрессия (**Linear regression) — модель зависимости переменной x от одной или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости. Данный метод позволяет предсказывать значения зависимой переменной y по значениям независимой переменной x.

**«Случайный лес» (**Random forest) — алгоритм [машинного обучения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предложенный [Лео Брейманом](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%9B%D0%B5%D0%BE)  и [Адель Катлер](https://en.wikipedia.org/wiki/Adele_Cutler)  (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D1%82%D0%BB%D0%B5%D1%80,_%D0%90%D0%B4%D0%B5%D0%BB%D1%8C&action=edit&redlink=1), заключающийся в использовании комитета (ансамбля) [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9). Алгоритм сочетает в себе две основные идеи: метод [бэггинга](https://ru.wikipedia.org/wiki/%D0%91%D1%8D%D0%B3%D0%B3%D0%B8%D0%BD%D0%B3) Бреймана, и [метод случайных подпространств](https://en.wikipedia.org/wiki/Random_subspace_method)  (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D0%BF%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2&action=edit&redlink=1), предложенный [Тин Кам Хо](https://en.wikipedia.org/wiki/Tin_Kam_Ho)  (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B8%D0%BD_%D0%9A%D0%B0%D0%BC_%D0%A5%D0%BE&action=edit&redlink=1).

Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого [ансамбля](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%81%D0%B0%D0%BC%D0%B1%D0%BB%D1%8C_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B2_(%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD)) [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9), каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

**Метод «k-ближайших соседей»** (k-nearest neighbors algorithm, k-NN) —метрический алгоритм для автоматической [классификации](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8) объектов или [регрессии](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). В случае использования метода для [классификации](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8) объект присваивается тому классу, который является наиболее распространённым

среди соседей данного элемента, классы которых уже известны. В случае использования метода регрессии, объекту присваивается значение по ближайшим к нему объектам, значения которых уже известны.

Алгоритм может быть применим к выборкам с большим количеством атрибутов (многомерным). Для этого перед применением нужно определить [функцию расстояния](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0_(%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)).

**Нейронная сеть** — [математическая модель](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей [нервных клеток](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD) живого организма. Это понятие возникло при изучении процессов, протекающих в [мозге](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B7%D0%B3), и при попытке [смоделировать](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) эти процессы. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в [задачах прогнозирования](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D0%BD%D0%BE%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), для [распознавания образов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2), в задачах [управления](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) и др.

Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными [алгоритмами](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC). Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять [обобщение](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5). Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

**1.3 Разведочный анализ данных.**

Разведочный анализ — это предварительный анализ данных с целью выявления наиболее общих зависимостей, закономерностей и тенденций, характера и свойств анализируемых данных, законов распределения анализируемых величин. Применяется для нахождения связей между переменными в ситуациях, когда отсутствуют (или недостаточны) априорные представления о природе этих связей.

К методам разведочного анализа относится процедура анализа распределений переменных, корреляционный анализ c целью поиска коэффициентов, превосходящих по величине определенные пороговые значения, факторный анализ, дискриминантный анализ, многомерное шкалирование, визуальный анализ гистограмм и т.д.

1. **Практическая часть.**
   1. **Предобработка данных.**

С целью решенияпоставленной задачи проведем объединение полученных исходных данных (excel-таблиц) по типу INNER по полю индекс в DataFrame = df\_bn.

Сформированный DataFrame исходных данных содержит 1023 записи с 9 входными параметрами и 3 выходными параметрами вещественного типа, пропуски значений отсутствуют.

Для анализа полученного DataFrama проведем следующие действия:

1. рассчитаем показатели описательной статистики (рис.1);
2. построим гистограммы плотности распределения данных (рис.2);
3. построим попарные графики рассеивания точек (рис.3);
4. построим диаграмму размаха («ящик с усами») (рис.4).

Визуализация данных позволяет определить характер распределений переменных.



Рисунок 1 - Описательная статистика

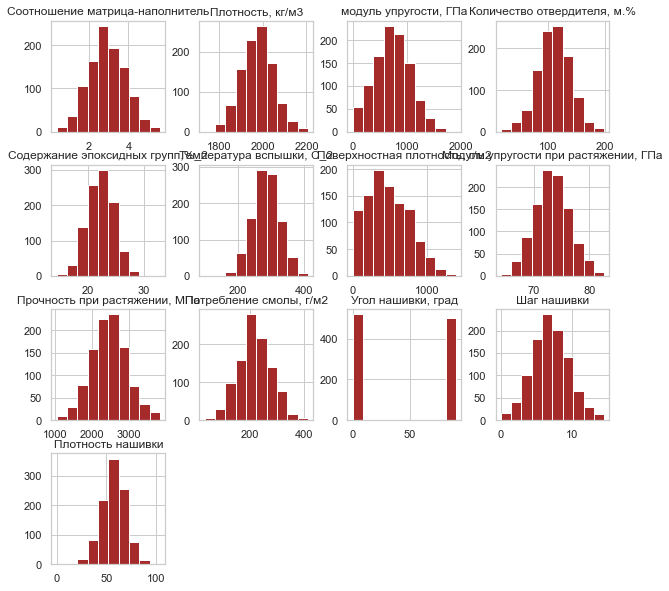


Рисунок 2 – Гистограммы плотности распределения

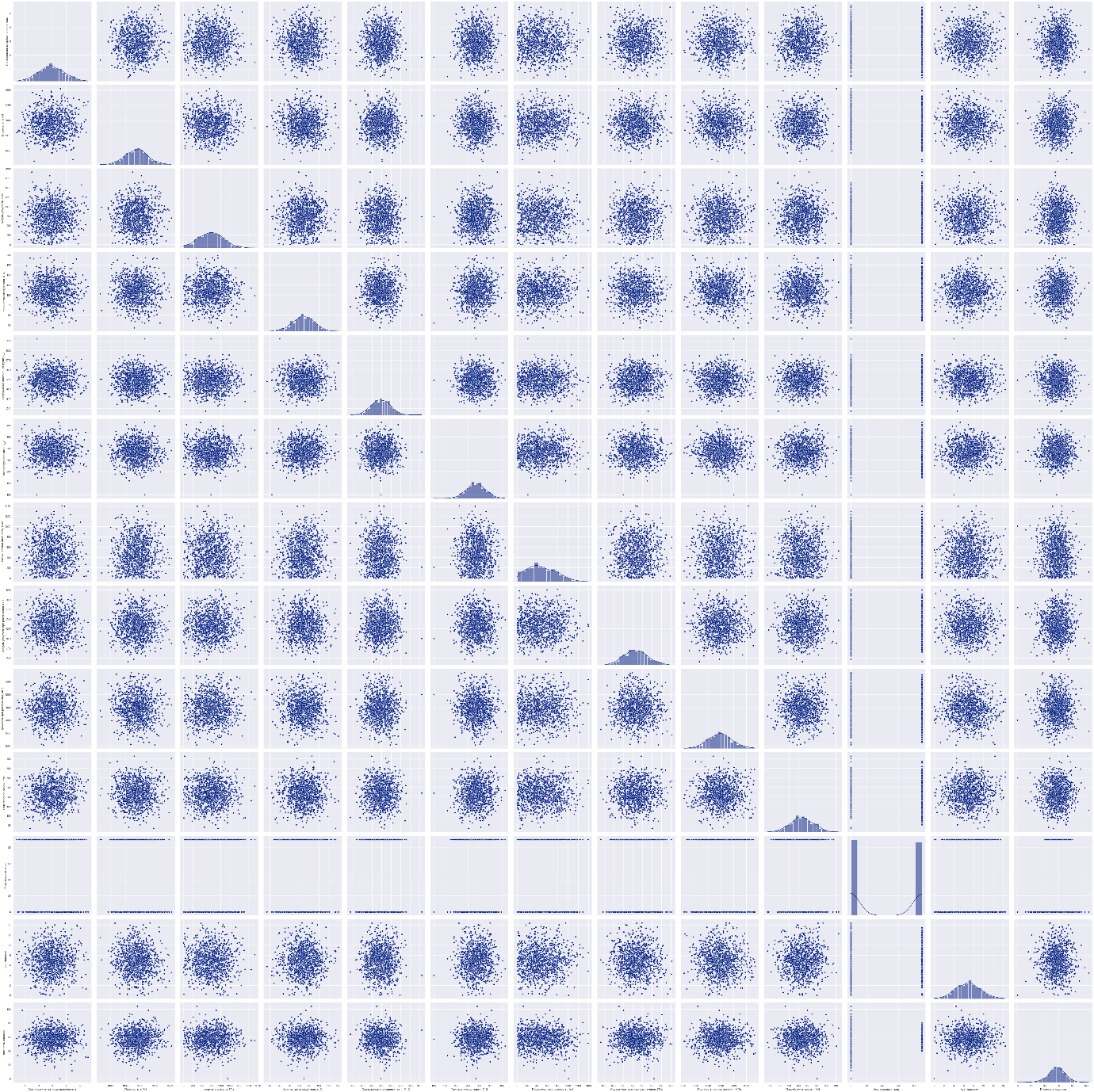


Рисунок 3 – Попарные графиков рассеяния точек

Для построения диаграммы размаха «ящик с усами» проведем нормализацию данных, то есть все переменные приведем к одному масштабу [0,1], что позволит нам определить выбросы (рис.4).

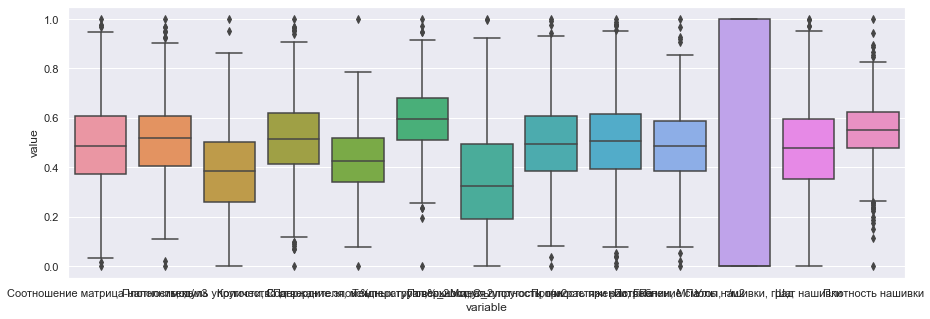


Рисунок 4 – диаграмма размаха «ящик с усами»

На основании диаграммы (рисунок 4) можно сделать вывод, что выбросы имеются по каждому из параметров, за исключением «Угол нашивки, град».

Общее количество полученных с помощью диаграммы «ящик с усами» выбросов составляет 92.

С использование метода Трех сигм проведем поиск выбросов по стандартизованному DataFrame. Выбросов получилось 25.

Не смотря на то, что по методу Трех сигм выбросов получилось в четыре раза меньше удалять выбросы не будем, а произведем замену выбросов с помощью метода «К-ближайших соседей».

Данное решение было принято на предположении, что данные выбросы являются фактическими значениями, тогда как значения которые попали в диапозон 3 сигм и «ящика с усами» были смоделированы опытным путем.

Проведем повторный анализ данных после проведенной замены экстремальных значений (выбросов):

1. рассчитаем показатели описательной статистики (рис.5);
2. построим гистограммы плотности распределения данных (рис.6);
3. построим попарные графики рассеивания точек (рис.7);
4. построим диаграмму размаха («ящик с усами») (рис.8).
5. построим тепловую карту попарной корреляции (рис.9)



Рисунок 5 - Описательная статистика по замены выбросов

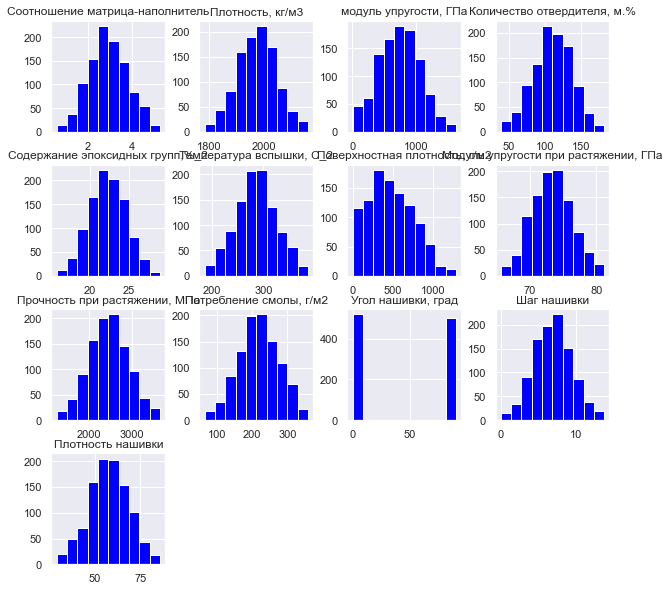


Рисунок 6 – Гистограммы плотности распределения после замены выбросов

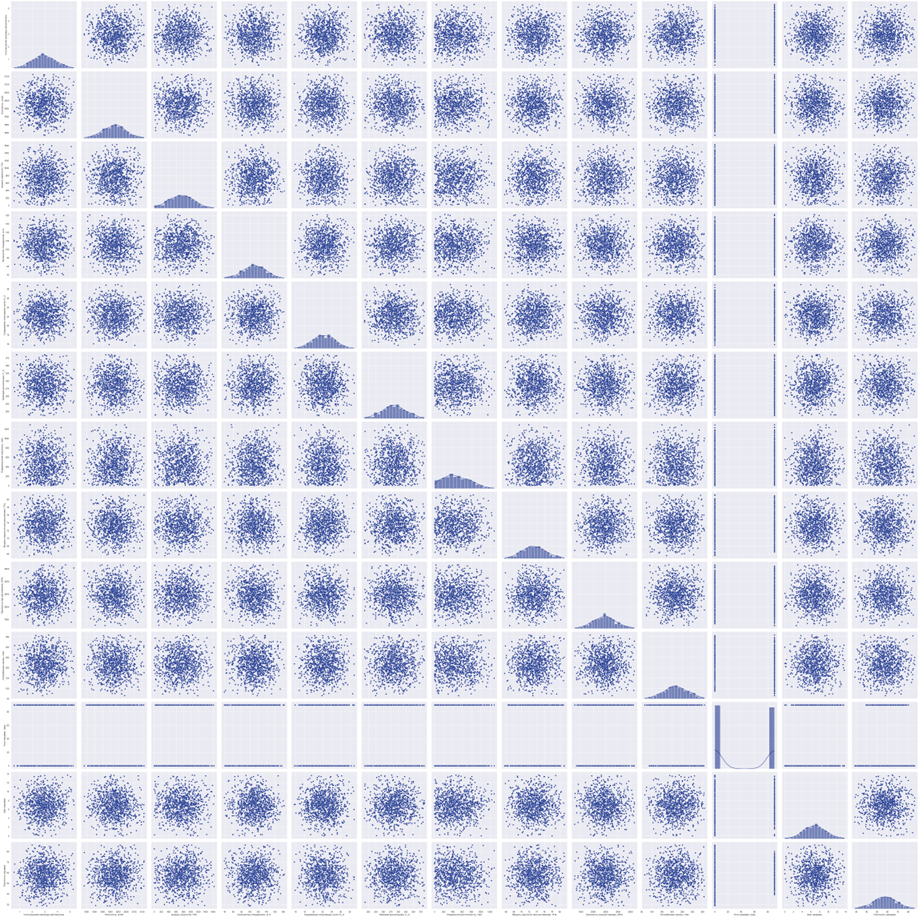


Рисунок 7 – Попарные графиков рассеяния точек после замены выбросов

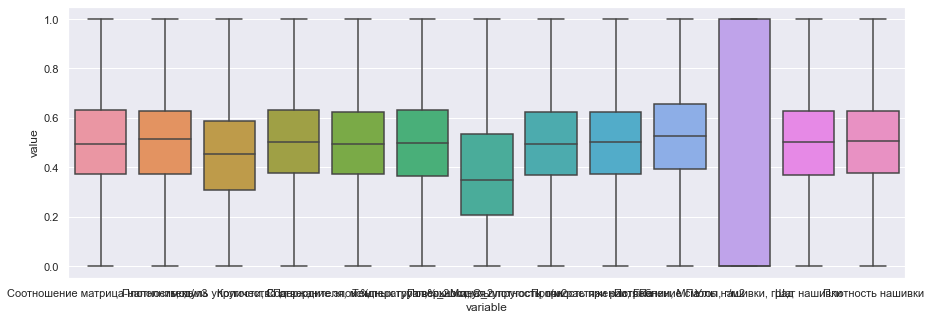


Рисунок 8 - диаграмма «ящик с усами» после замены выбросов

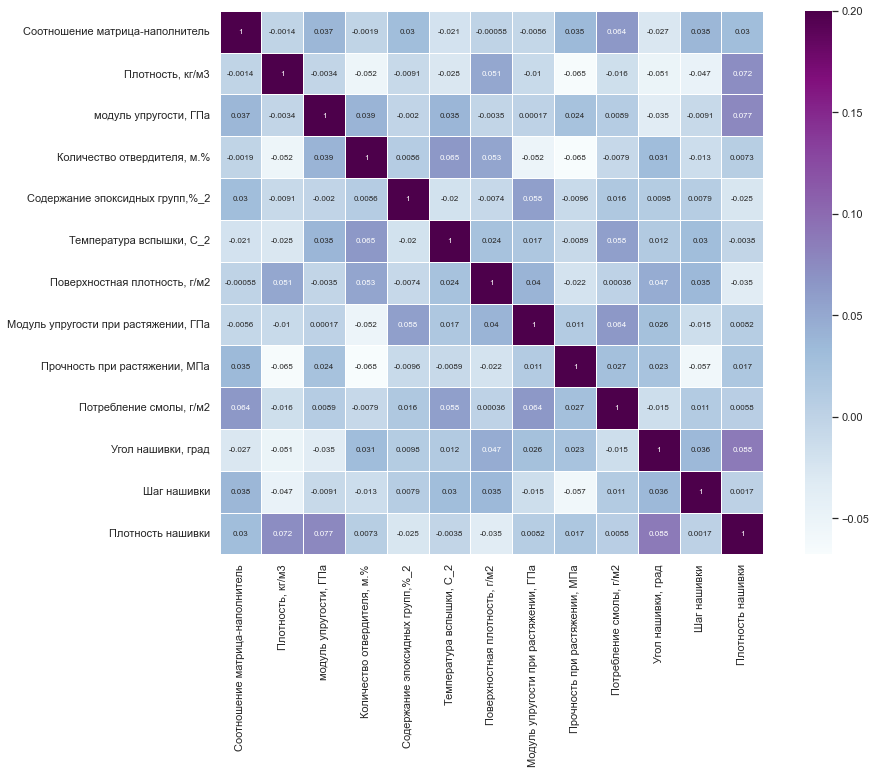


Рисунок 9 – тепловая карта попарной корреляции

Следующим этапом предобработки данных является их нормализация, то есть приведение всех входных данных к заданному диапазону, в данной работе [0…1].

Нормализацию проводим с помощью метода MinMaxScaler библиотеки Scikit-Learn.

Проведенный анализ показал, что между параметрами слабая зависимость. На это указывают полученные значения корреляции (max R = 0.077), а так же равномерное распределение значений на графиках попарного рассеяния и гистограммах плотности распределения.

Распределение практически идеальное для каждого параметра, что позволяет предположить о том, что данные были размножены (доведены до текущего объема) опытным путем.

**2.2 Разработка и обучение моделей.**

В данной части работы необходимо обучить нескольких моделей для прогноза параметров «модуль упругости при растяжении» и «прочность при растяжении». Выбрать наилучшую руководствуясь сравнением показателей моделей по выбранным метрикам.

При построении модели необходимо 30% данных оставить на тестирование модели, на остальных будет происходить обучение моделей.

При построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

Для прогноза параметров «модуль упругости при растяжении» и «прочность при растяжении» использованы следующие модели:

Линейная регрессия – простая модель, используется на небольших DataFrame, исследует взаимосвязь данных. Чувствительна к выбросам.

«Случайный лес» универсальная модель, нечувствительная к выбросам.

Метод «k-ближайших соседей» модель чувствительна к нормализации данных, подходит как для задач классификации, так и для задач регрессии.

Для оценки качества модели будем использовать следующие метрики:

1. Средняя квадратическая ошибка (Mean Square Error);
2. Среднеквадратическая ошибка (Root Mean Square Error);
3. Коэффициент детерминации R2.

Для каждого параметра строилась(обучалась) своя модель.

С помощью метода train\_test\_split библиотеки Scikit-Learn данные были разделены на обучающую и тестовую выборки.

Далее с помощью библиотеки Scikit-Learn строим и обучаем модели:

1. модель Линейной регрессии, с помощью метода LinearRegression(): model = LinearRegression();
2. модель «Случайный лес», с помощью метода RandomForestRegressor()

RFReg = Model(

name = 'RandomForestRegressor',

method = RandomForestRegressor(),

parametrs = {

'n\_estimators': [100, 200, 500],

'max\_features': ['auto', 'sqrt', 'log2'],

'max\_depth' : [4,8,12],

'criterion' : ['mse']

})

1. модель «k-ближайших соседей», с помощью метода KNeighborsRegressor()

KBS = Model(

name = 'KNeighbors',

method = KNeighborsRegressor(),

parametrs = {

'n\_neighbors': [10, 50, 100, 200]

})

Для поиска гиперпараметров будем использовать метод под названием GridsearchCV , а именно поиск по сетке и перекрестную проверку:

GSCV = GridSearchCV(estimator=model, param\_grid=parametrs, cv=10, verbose=2)

GSCV.fit(X\_train, y\_train)

model = GSCV.best\_estimator\_

param = GSCV.best\_params\_

Для проверки обученной модели рассчитываем «предсказания» на тестовой выборке и с помощью визуализации определяем на сколько точно предсказанные данные совпадают с реальными значениями.

Считаем ошибку построенной и обученной модели, для этого сравниваем «предсказанные» значения со значениями в тестовой выборке.

Выведем полученные результаты по каждой модели и параметру:

**Модель линейной регрессии**



Рисунок 10 Метрики линейной регрессии

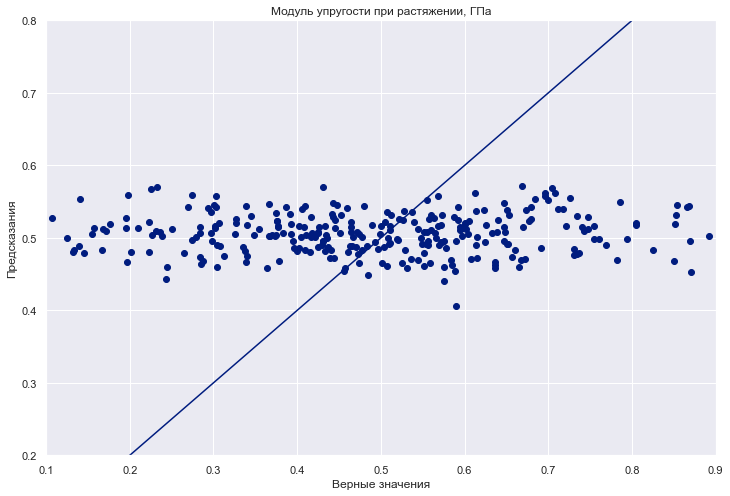


Рисунок 11 График точности предсказанной модели линейной регрессии

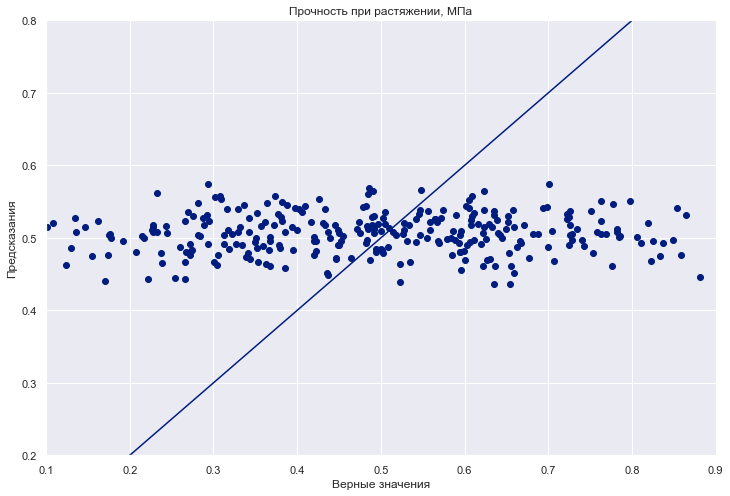


Рисунок 12 График точности предсказанной модели линейной регрессии

**Модель «случайный лес»**



Рисунок 13 Метрики модели «случайный лес»



Рисунок 14 Модель «случайный лес» с лучшими параметрами

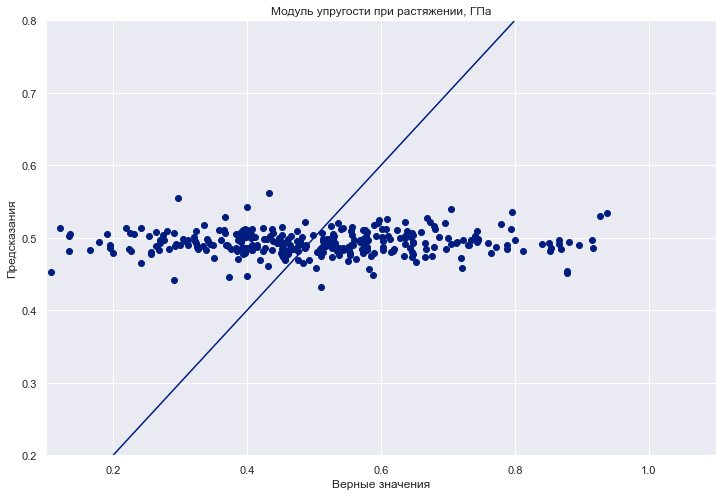


Рисунок 15 График точности предсказанной модели «случайный лес»

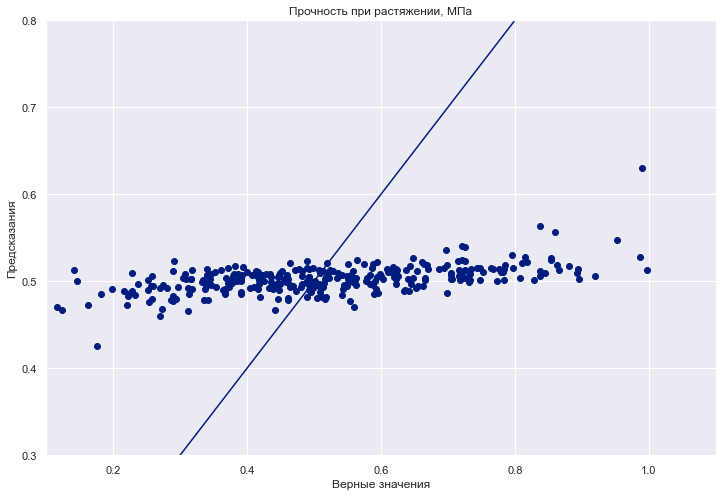


Рисунок 16 График точности предсказанной модели «случайный лес»

**Модель «k-ближайших соседей»**



Рисунок 17 Метрики модели «к-ближайших соседей»



Рисунок 18 Модель «к-ближайших соседей» с лучшими параметрами

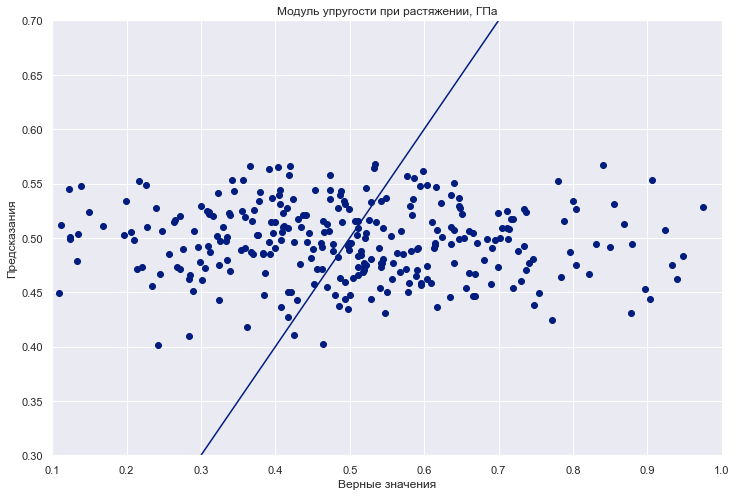


Рисунок 19 График точности предсказанной модели «к-ближайших соседей»

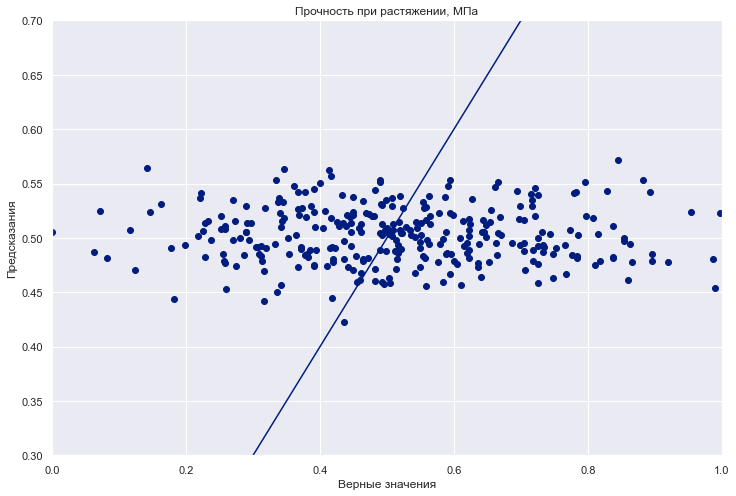


Рисунок 20 График точности предсказанной модели «к-ближайших соседей

Выведем метрики моделей в общую таблицу:



Как видим из таблицы лучший прогнозный результат для параметра «модуль упругости при растяжении» показала модель на основе метода «случайный лес», а для параметра «прочность при растяжении» модель на основе метода «К-ближайших соседей».

**2.3 Разработка нейронной сети.**

В данном разделе необходимо написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнить.

Рассмотрим реализацию многослойного персептрона на Python с использованием надстройки для глубокого обучения Keras. Написание нейронной сети будем проводить на полном DataFrame с ненормализованными данными, но очищенным от выбросов.

Выходной параметр – «Соотношение матрица – наполнитель»

Входные параметры – все остальные (12 параметров)

Следующим этапом разбиваем DataFrame на выборки: обучающую, валидационную и тестовую.

Строим нейронные сети. Используем Последовательную модель Sequential.

1. Линейная модель.

Построим нейронную сеть состоящую из входного слоя (12 параметров) и выходного (1 параметр).

В качестве входного слоя используем встроенный в TensorFlow метод Normalization (стандартизатор).

Для обучения модели в качестве оптимизатора выбираем последовательно Adam, SGD, RMSprop. В качестве критерия loss используем Mean Square Error (средняя квадратическая ошибка).

В целях минимизации последствий минимизации переобучения модели используем функцию CALLBACKS (остановка обучения когда целевой показатель перестает улучшаться):

%%time

history = SMNlin\_model2.fit(

X\_train,

y\_train,

epochs=1023,

verbose=0,

callbacks=[early\_stop],

validation\_data=(X\_val, y\_val)

)

Архитектура сети:

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

normalization (Normalizatio (None, 12) 25

n)

dense\_1 (Dense) (None, 1) 13

=================================================================

Total params: 38

Trainable params: 13

Non-trainable params: 25

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выведем метрики по результатам обучения линейной модели



Посмотрим процесс обучения и график точности предсказания модели SMNlin\_model4.

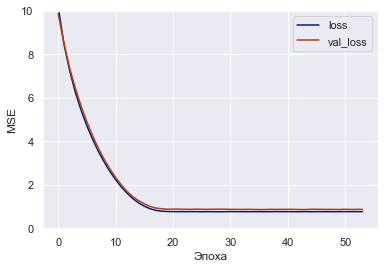


Рисунок 21 Процесс обучения линейной модели

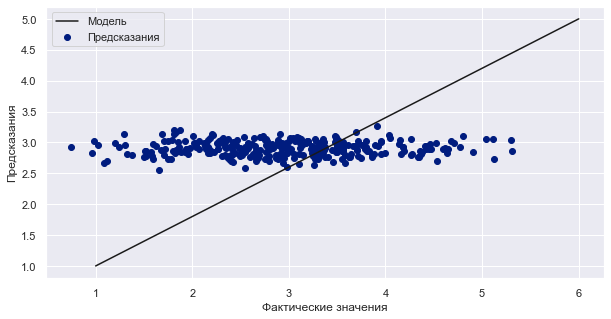


Рисунок 22 График точности предсказанной линейной модели

1. Многослойный персептрон (MLP)

Добавим нашей нейронной сети скрытых слоев. В качестве активационной функции используем ReLu. В качестве оптимизатора выбираем последовательно Adam, SGD, RMSprop. В качестве критерия loss используем Mean Square Error (средняя квадратическая ошибка).

В целях минимизации последствий минимизации переобучения модели используем функцию CALLBACKS (остановка обучения когда целевой показатель перестает улучшаться).

Выведем метрики по результатам обучения моделей с различным количеством скрытых слоев и других параметров:



Посмотрим архитектуру сети, процесс обучения и график точности предсказания модели SMNmp\_264\_adam.

Model: "sequential\_4"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

normalization (Normalizatio (None, 12) 25

n)

dense\_4 (Dense) (None, 64) 832

dense\_5 (Dense) (None, 64) 4160

dense\_6 (Dense) (None, 1) 65

=================================================================

Total params: 5,082

Trainable params: 5,057

Non-trainable params: 25

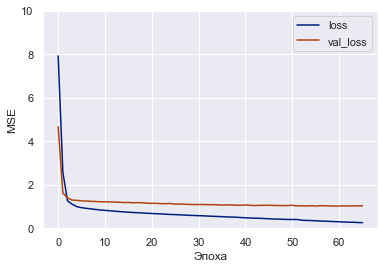


Рисунок 23 Процесс обучения модели SMNmp\_264\_adam

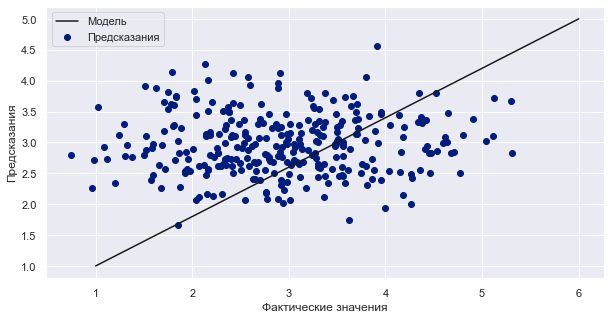


Рисунок 24 График точности предсказанной модели SMNmp\_264\_adam

1. Многослойный персептрон с Dropout слоями.

Попробуем в модель SMNmp\_3128\_adam добавить Dropout слои.

Dropout — метод решения проблемы переобучения в нейронных сетях.

Главная идея Dropout — вместо обучения одной DNN обучить ансамбль нескольких DNN, а затем усреднить полученные результаты.

Обучим модель и посмотрим результаты:

Model: "sequential\_13"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

normalization (Normalizatio (None, 12) 25

n)

dense\_37 (Dense) (None, 128) 1664

dropout (Dropout) (None, 128) 0

dense\_38 (Dense) (None, 128) 16512

dropout\_1 (Dropout) (None, 128) 0

dense\_39 (Dense) (None, 128) 16512

dropout\_2 (Dropout) (None, 128) 0

dense\_40 (Dense) (None, 1) 129

=================================================================

Total params: 34,842

Trainable params: 34,817

Non-trainable params: 25

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рисунок 25 Архитектура модели SMNmp\_3128\_adamdrop

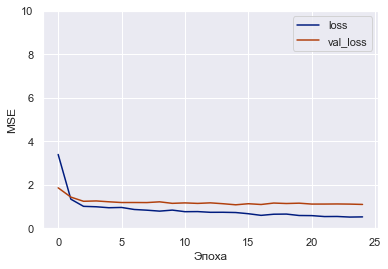


Рисунок 26 Процесс обучения модели SMNmp\_3128\_adamdrop

Посчитаем ошибки, запишем в таблицу:



Рисунок 27 Оценка качества обучения модели SMNmp\_3128\_adamdrop

Построим график-сравнение предсказанных моделью значений с фактическими:

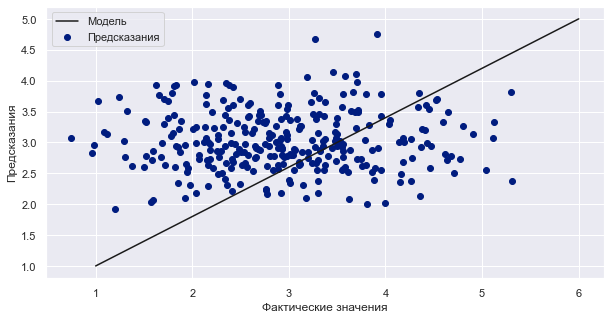
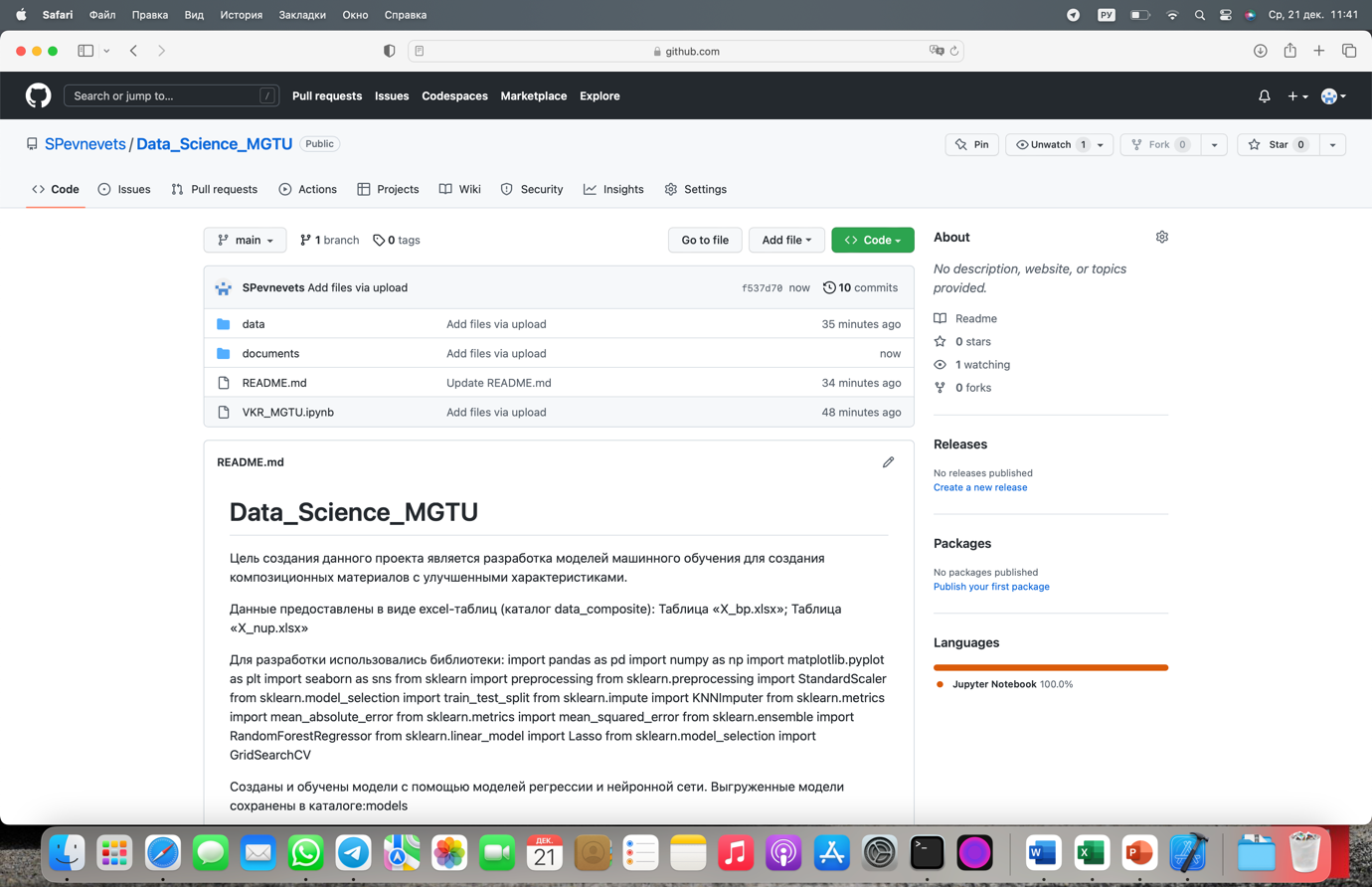


Рисунок 28 График точности предсказанной модели SMNmp\_3128\_adamdrop

**3. Создание удаленного репозитория**

Создана страница на GitHub.

Адрес страницы https://github.com/SPevnevets/Data\_Science\_MGTU



В репозитории загружены: наборы данных, пояснительная записка, презентация и jupyterbook.

Рисунок 29 – Репозиторий на GitHub

**4. Заключение**

В ходе работы были разработаны модели машинного обучения и нейронные сети для прогноза конечных свойств получаемых композиционных материалов.

Разработанные прогнозные модели не могут использоваться на практике так как создавались на основе недостаточного набора и качества исходных данных. Возможно при условии повышения качества исходных данных удастся повысить и качество самих моделей, что позволит использовать их для прогнозирования и тем самым сократить количество проводимых испытаний, а также создавать композиционные материалы с улучшенными характеристиками.

**5. Список используемой литературы**

1. Статистика: Курс лекций/Харченко Л.П. и др.-Новосибирск: Изд-во НГАЭиУ,М.:ИНФАРА-М, 1997 -310с
2. Аллен Б. Дауни – Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. - Москва : Манн, Иванов и Фербер, 2021. -304 с.
3. Джоэл Грас, Data Sciense. Наука о данных с нуля: Пер. с англ. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2022. – 416с.
4. Библиотека Pandas – Режим доступа: https://pandas.pydata.org/ (дата обращения 10.06.2022)
5. Библиотека Matplotlib – Режим доступа: https:// https://matplotlib.org/ (дата обращения 10.06.2022)
6. Язык программирования Python: - Режим доступа: https://www.python.org/ (дата обращения 10.06.2022)
7. Библиотека Sklearn – Режим доступа: https://scikit-learn.org/stable/ (дата обращения 10.06.2022)