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 Васильева Оксана Валерьевна

Москва, 2022

Оглавление

Введение

Композиционные материалы – это многокомпонентные материалы, состоящие из двух или более компонентов, один из которых называется матрица, в которой помещаются материалы называемые арматурой. Композиты при этом являются монолитным материалом, то есть компоненты не являются отделимыми.

Композиты обладают свойствами, которыми не обладает ни один из материалов в отдельности. Поэтому даже зная характеристики связующего и армирующего компонентов, свойства композита будут неизвестны. Для определения этих свойства есть два способа: первый – построение композита и его физическое испытание и второй – построение прогностических моделей. Первый способ зачастую затруднителен и весьма затратен, поэтому прогнозирование свойств композитов на основе свойств материалов компонентов является вполне актуальным.

В рамках текущей работы были разработаны модели машинного обучения для прогнозирования модуля упругости при растяжении, прочности при растяжении, а также разработана нейронная сеть для соотношения матрица-наполнитель. С помощью этой нейронной сети было разработано приложение с графическим интерфейсом.

1. Аналитическая часть

1.1. Постановка задачи

Цель работы разработать модели машинного обучения для прогноза модуля упругости при растяжении, прочности при растяжении и соотношения матрица-наполнитель. По одной из разработанных моделей требуется создать приложение с графическим интерфейсом или интерфейсом командной строки.

По заданию имеются два excel файла: X_bp.xlsx (данные с параметрами) и X_nup.xlsx (данные нашивок углепластика). В первом файле 1023 строки и 11 столбцов , во втором – 1040 и 4. Первые столбцы в обоих датасетах индексные и их можно удалить. Требуется объединить оба файла методом по индексам по типу INNER, лишние строки удалить.

```
df1=pd.read_excel('Датасет для ВКР_композиты\X_bp.xlsx')
df1=df1.drop(['Unnamed: 0'], axis=1)
df1.shape
```

```
(1023, 10)
```

```
df1.head(2)
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2
0	1.857143	2030.0	738.736842	30.0	22.267857	100.000000	210.0	70.0	3000.0	220.0
1	1.857143	2030.0	738.736842	50.0	23.750000	284.615385	210.0	70.0	3000.0	220.0

Рис. 1.1.1. Загрузка данных из X_bp.xlsx и удаления столбца

```
df2=pd.read_excel('Датасет для ВКР_композиты\X_nup.xlsx')
df2=df2.drop(['Unnamed: 0'], axis=1)
df2.shape
```

```
(1040, 3)
```

```
df2.head(2)
```

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0.0	4.0	57.0
1	0.0	4.0	60.0

Рис. 1.1.2. Загрузка данных X_nup.xlsx и удаления столбца

Необходимо провести разведочный анализ предложенных данных. Необходимо нарисовать гистограммы распределения каждой из переменной, диаграммы ящика с усами, попарные графики рассеяния точек. Необходимо также для каждой колонке получить среднее, медианное значение, провести анализ и исключение выбросов, проверить наличие пропусков.

Требуется провести предобработку данных.

Затем надо обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели необходимо 30% данных оставить на тестирование модели, на остальных происходит обучение моделей. При построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

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

Требуется разработать приложение с графическим интерфейсом или интерфейсом командной строки, которое будет выдавать прогноз. Оценить точность модели на тренировочном и тестовом датасете. Создать репозиторий в GitHub / GitLab и разместить там код исследования. Оформить файл README.

1.2. Описание используемых методов

В рамках поставленного задания будут строиться три типа моделей для прогноза модуля упругости, прочности при растяжении и соотношения «матрица-наполнитель». Каждая из этих моделей будет решать задачу регрессии. Для этих целей выбраны следующие алгоритмы

- Линейная регрессия;
- Метод случайного леса;
- Метод опорных векторов;
- Метод k-ближайших соседей;
- Градиентный бустинг (библиотека sklearn);
- Градиентный бустинг (библиотека xgboost);
- А также использована библиотека autogluon, которая использует некоторый набор алгоритмов машинного обучения.
- В качестве нейронной сети выбран многослойный персептрон.

Линейная регрессия

Линейная регрессия — это алгоритм машинного обучения, основанный на обучении с учителем. Он выполняет задачу регрессии. Регрессия моделирует значение зависимой на основе независимых. Алгоритм в основном используется для выяснения взаимосвязи между переменными и прогнозирования. Различные модели регрессии различаются в зависимости от типа взаимосвязи между зависимыми и независимыми переменными, которые они рассматривают, и количества используемых независимых переменных.

Достоинства метода:

- скорость,
- простота,
- интерпретируемость.

Недостатки метода:

- хорошо описывает только линейные зависимости.

Метод случайного леса

Метод случайного леса – алгоритм машинного обучения на основе ансамбля деревьев решений. В задаче регрессии берётся усредненный ответ множества деревьев решений.

Достоинства метода:

- Снижение риска переобучения: деревья решений подвержены риску переобучения, поскольку они, как правило, точно соответствуют всем образцам в обучающих данных. Однако при наличии большого количества деревьев решений в случайном лесу классификатор не будет соответствовать модели, поскольку усреднение некоррелированных деревьев снижает общую дисперсию и ошибку прогноза.
- Обеспечивает гибкость: Пакетирование признаков делает классификатор случайного леса эффективным инструментом для оценки пропущенных значений, поскольку он вполне работоспособен, когда часть данных отсутствует.
- Интерпретируемость: случайный лес позволяет легко оценить важность переменной или ее вклад в модель. Есть несколько способов оценить важность функции. Критерий Джини и среднее уменьшение примеси (MDI) обычно используются для измерения того, насколько снижается точность модели при исключении данной переменной.

Недостатки метода:

- Метод достаточно медленный
- Требует достаточно много памяти

Метод опорных векторов

Цель алгоритма машины опорных векторов состоит в том, чтобы найти гиперплоскость в n-мерном пространстве, которая четко классифицирует точки данных. Точки данных по обе стороны от гиперплоскости, которые находятся ближе всего к гиперплоскости, называются опорными векторами. Они влияют на

положение и ориентацию гиперплоскости и, таким образом, помогают построить SVM.

Достоинства метода:

- Он устойчив к выбросам;
- Модель решения может быть легко обновлена;
- Он обладает отличной способностью к обобщению с высокой точностью предсказания.

Недостатки метода:

- Не подходит для больших датасетов;
- Плохая интерпретируемость.

Метод k-ближайших соседей

Метод k-ближайших соседей (KNN) – метод машинного обучения с учителем, при котором в задачах регрессии объекту присваивается среднее значение от k ближайших соседей.

Алгоритм KNN также является частью семейства моделей «ленивого обучения», что означает, что он хранит только обучающий набор данных, а не проходит этап обучения. Это также означает, что все вычисления происходят во время классификации или предсказания. Поскольку он в значительной степени зависит от памяти для хранения всех своих обучающих данных, его также называют методом обучения на основе экземпляров или памяти.

Достоинства метода:

- Точность и простота
- Адаптируемость. Новые данные легко добавляются.
- Малое число гиперпараметров: методу требуется лишь значение k и метрика.

Недостатки метода:

- Плохо масштабируется: поскольку KNN является ленивым алгоритмом, он занимает больше памяти и места для хранения данных по сравнению с методами.
- Алгоритм плохо работает с многомерными входными данными.
- Склонность к переобучению.

Градиентный бустинг (sklearn)

Градиентный бустинг – метод машинного обучения на основе ансамбля слабых алгоритмов прогнозирования, чаще всего деревьев решений. Бустинг – ансамблевый метод последовательного применения алгоритмов. Так как новые алгоритмы учатся на ошибках совершенных предыдущими, необходимо меньше вычислений для уменьшения ошибки. Функция потерь обучается с помощью градиентного спуска.

Преимущества метода:

- Алгоритм работает с любыми функциями потерь
- Достаточно точные предсказания

Недостатки метода:

- Алгоритм чувствителен к выбросам. Имеет смысл использовать MAE, а не mse.
- Имеет склонность к переобучению при большом числе деревьев
- Достаточно трудоёмкий в плане вычислений алгоритм.

Градиентный бустинг (xgboost)

Быстрая и масштабируемая реализация градиентного бустинга.

Преимущества метода:

- Может справляться с пропущенными данными.
- Быстрая и точная реализация градиентного бустинга

Autogluon

Autogluon – библиотека автоматического машинного обучения. В отличие от существующих сред AutoML, которые в основном ориентированы на при выборе модели/гиперпараметра, AutoGluon успешно объединяет несколько моделей и укладывать их в несколько слоев. Эксперименты показали, что многослойная комбинация многих моделей более эффективна [7].

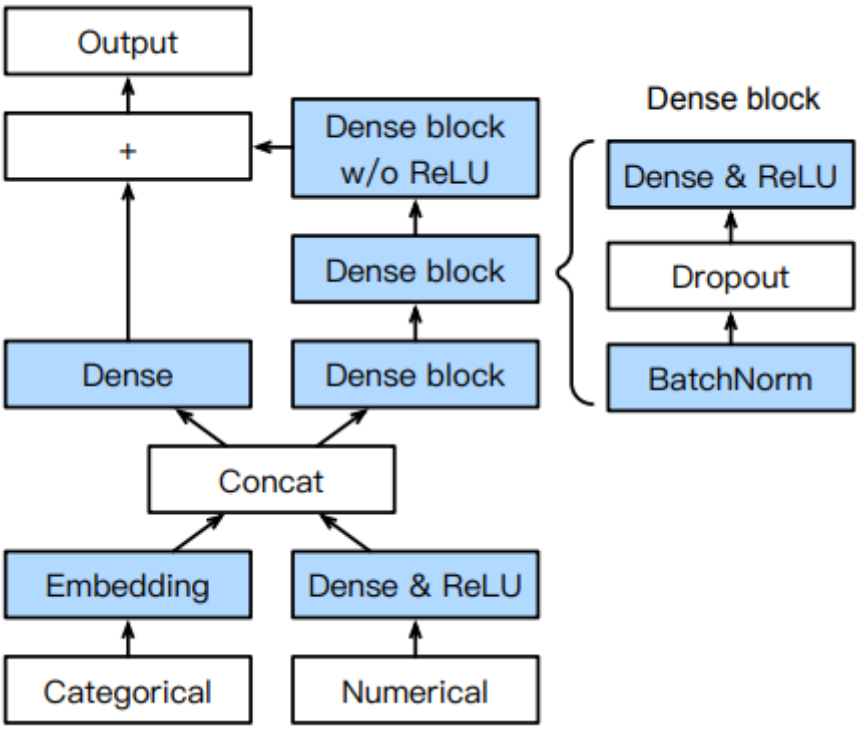

Рис. Структура нейронной сети из Autogluon

Достоинства:

- Набор разных алгоритмов, гибкий и точный
- Хорошие результаты в соревнованиях. Имеется оценка платформ машинного обучения, включая TPOT, H2O, AutoWEKA, auto-sklearn, AutoGluon и таблицы Google AutoML, на наборе из 50 задач классификации и регрессии от Kaggle и OpenML AutoML Benchmark. По результат тестов известно, что AutoGluon быстрее, надежнее, и гораздо точнее. В двух популярных Kaggle, AutoGluon победил 99% участвующих датасайентистов [7].

Недостатки:

- Требует python 3.9 и младше.

- Медленно обучается

Многослойный персептрон

Многослойный персептрон – нейронная сеть прямого распространения, имеющая один входной слой, один выходной и один или несколько скрытых слоёв. В персептроне используется обратное распространение ошибки.

Нейронные сети достаточно гибки в настройке и позволяют использовать достаточно большое число гиперпараметров: количество слоёв и количество нейронов на них, различные функции активации, алгоритмы оптимизации, количество эпох обучения, разные размеры батчей, и прочими параметрами.

Достоинства:

- Точность при достаточно удачной архитектуре нейронной сети, скорее всего это будет самый точный алгоритмом из представленных.

Недостатки:

- Сложность в настройке
- Медленный
- Возможно переобучение.

1.3. Разведочный анализ данных

На первом этапе работы с данными их необходимо изучить. Необходимо выявить пропуски, выбросы, дубликаты, несоответствие типов, если имеется. Данные объединённого на первоначальном этапе не требуется преобразовывать, так нет пропусков, нет дубликатов и имеющиеся типы данных годятся для дальнейшей работы.

```
Ввод [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Соотношение матрица-наполнитель        1023 non-null   float64
 1   Плотность, кг/м3                       1023 non-null   float64
 2   модуль упругости, ГПа                  1023 non-null   float64
 3   Количество отвердителя, м.%            1023 non-null   float64
 4   Содержание эпоксидных групп,%_2        1023 non-null   float64
 5   Температура вспышки, С_2               1023 non-null   float64
 6   Поверхностная плотность, г/м2          1023 non-null   float64
 7   Модуль упругости при растяжении, ГПа   1023 non-null   float64
 8   Прочность при растяжении, МПа          1023 non-null   float64
 9   Потребление смолы, г/м2                1023 non-null   float64
 10  Угол нашивки, град                     1023 non-null   float64
 11  Шаг нашивки                            1023 non-null   float64
 12  Плотность нашивки                      1023 non-null   float64
dtypes: float64(13)
memory usage: 111.9 KB

Все данные имеют тип float64, пропущенных значений нет.

Поиск дубликатов

Ввод [8]: df.duplicated().sum()
Out[8]: 0
```

Рис.1.3.1. Данные о датасете

Для оценки статистических данных использовались гистограммы; «ящики с усами»; графики рассеяния; тепловая карта для корреляционной матрицы; выводились некоторые статистические показатели.

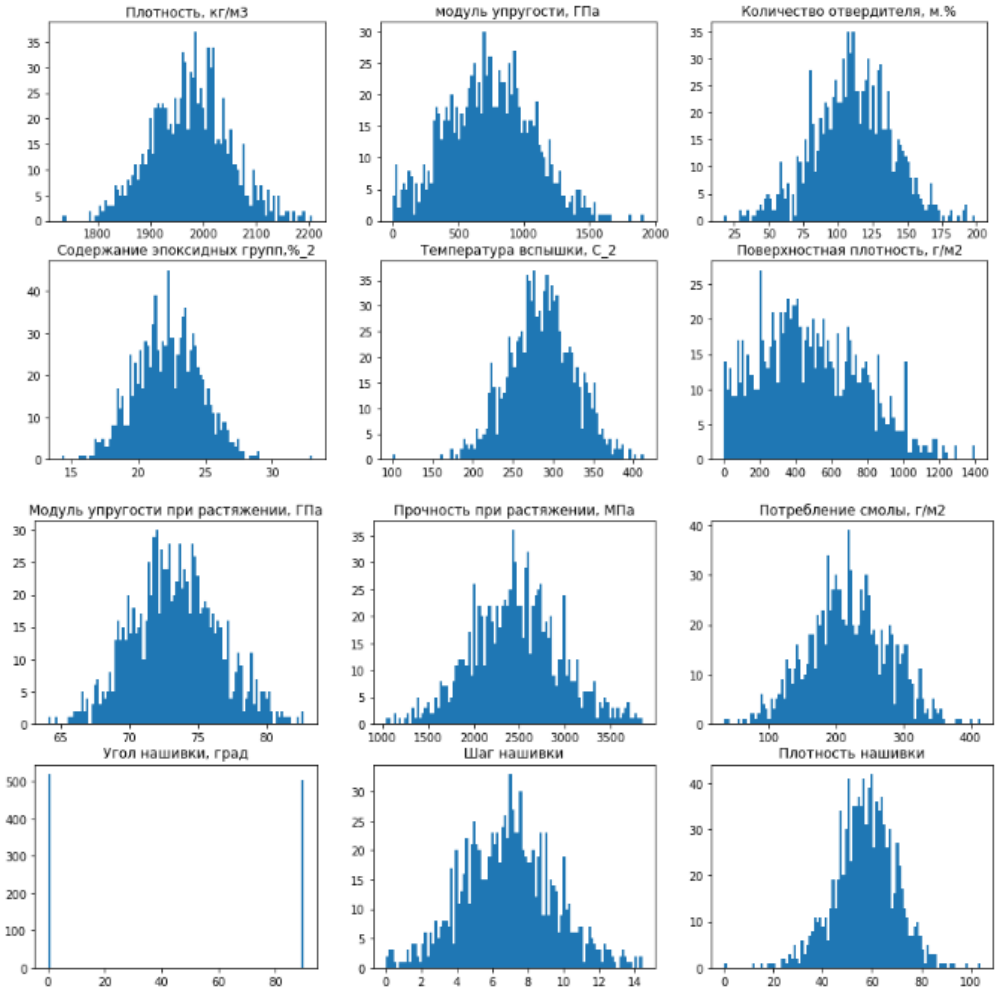

Рис. 1.3.2. Гистограммы

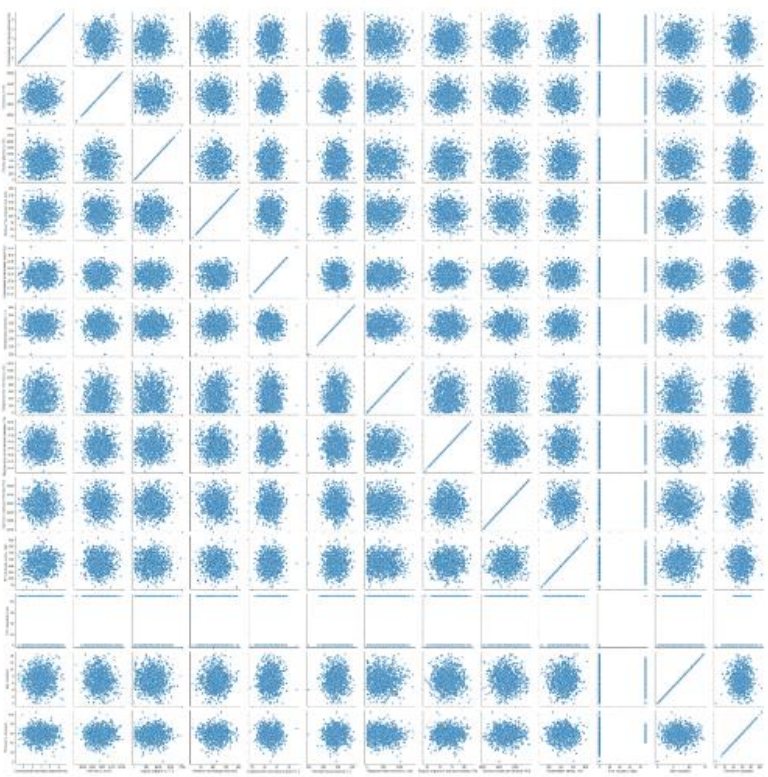

Рис. 1.3.3. Диаграммы рассеяния

```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```

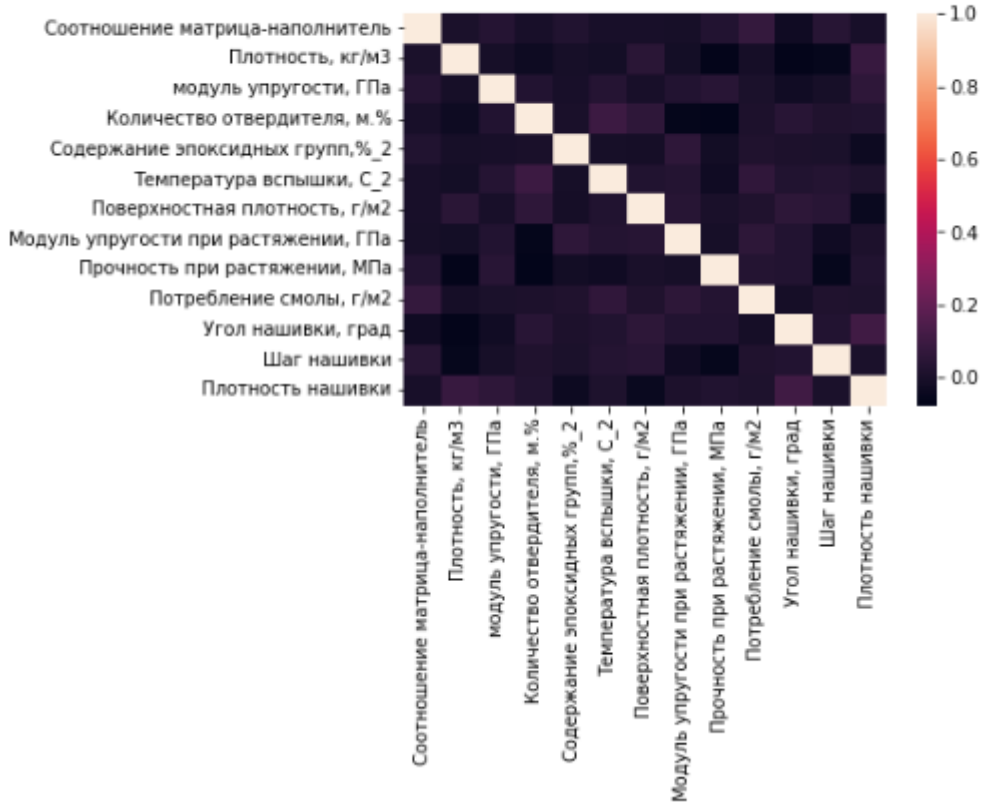

Рис. 1.3.4. Тепловая карта корреляционной матрицы

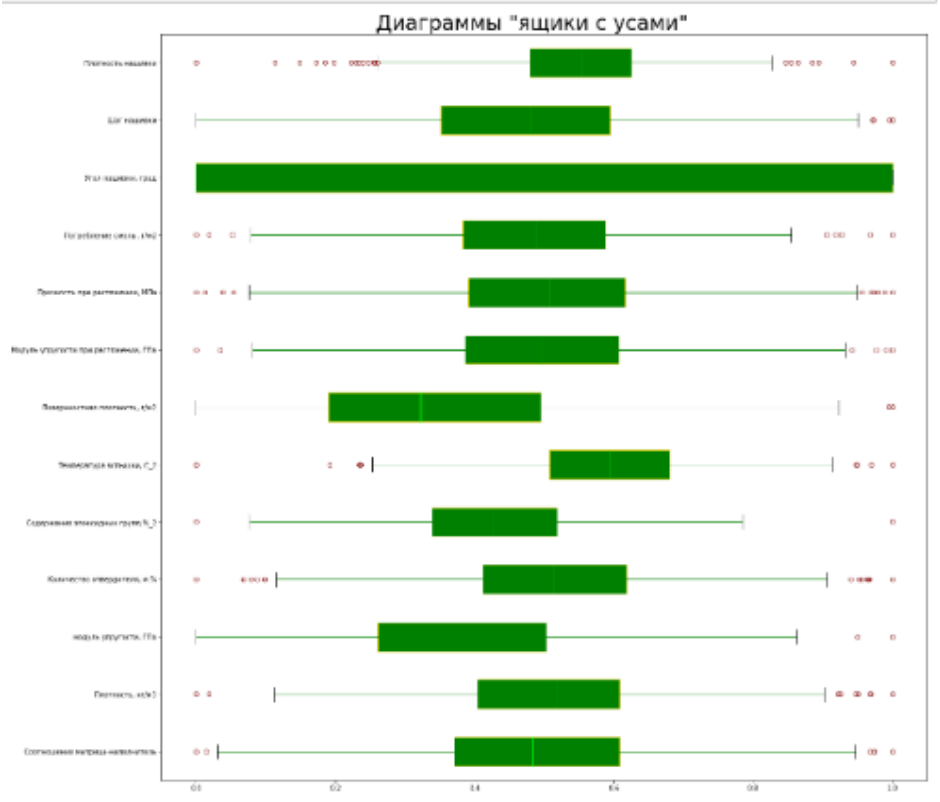

Рис. 1.3.5. «Ящики с усами»

2. Практическая часть

2.1. Предобработка данных

Удалим выбросы с помощью интерквартильного размаха

```
for i in df.drop(['Угол нашивки, град'], axis=1).columns:
    quart3, quart1 = np.percentile(df.loc[:,i], [75,25])
    max = quart3 + (1.5 * (quart3-quart1))
    min = quart1 - (1.5 * (quart3-quart1))
    df.loc[df[i] < min, i] = np.nan
    df.loc[df[i] > max, i] = np.nan
```

```
df.isnull().sum()
```

```
Соотношение матрица-наполнитель          6
Плотность, кг/м3                         9
модуль упругости, ГПа                    2
Количество отвердителя, м.%              14
Содержание эпоксидных групп,%_2          2
Температура вспышки, С_2                 8
Поверхностная плотность, г/м2            2
Модуль упругости при растяжении, ГПа     6
Прочность при растяжении, МПа            11
Потребление смолы, г/м2                  8
Угол нашивки, град                       0
Шаг нашивки                              4
Плотность нашивки                        21
dtype: int64
```

```
Удалим выбросы.
```

```
df=df.dropna()
```

Рис. 2.1.1. Нахождение и удаление выбросов

Количество строк после удаления выбросов – 936.

2.2. Разработка и обучение модели

Необходимо разработать модели машинного обучения для предсказания двух выходных параметров: «Прочность при растяжении» и «Модуль упругости при растяжении».

Порядок разработки модели выглядит следующим образом: разделение данных на тренировочные и тестовые в соотношении 70 на 30; нормализация выборок; обучение моделей.

```
X, X_test, y, y_test = train_test_split(df.drop([target], axis=1), df[target], test_size = 0.3, random_state = 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(X)
X = scaler.transform(X)
X_test = scaler.transform(X_test)
```

Рис. 2.2.1. Разделение на обучающую и тестовую выборки. Нормализация выборок.

Были выбраны описанные в 1.2 модели с параметрами по умолчанию.

Проведена гиперпараметрическую оптимизацию градиентного бустинга по следующим параметрам: функция ошибки(loss), количество и глубина деревьев, минимальный размер листа, минимальное ветвление, функция лучшего разделения. Каждый параметр рассматривался в отдельности, поэтому итоговый результат не обязательно будет лучше стандартного случая.

```
loss = ['ls', 'absolute_error', 'huber']
#деревья
n_estimators = [100, 500, 900, 1100, 1500]
# Глубина деревьев
max_depth = [2, 3, 5, 10, 15]
# минимальный размер листа
min_samples_leaf = [1, 2, 4, 6, 8]
# Минимальное количество ветвлений на узел
min_samples_split = [2, 4, 6, 10]
# Размер делящегося множества
max_features = ['auto', 'sqrt', 'log2', None]

# Сетка гиперпараметров
hyperparameter_grid = [{'loss': loss},
                       {'n_estimators': n_estimators},
                       {'max_depth': max_depth},
                       {'min_samples_leaf': min_samples_leaf},
                       {'min_samples_split': min_samples_split},
                       {'max_features': max_features}]
```

Рис. 2.2.2. Сетка гиперпараметров

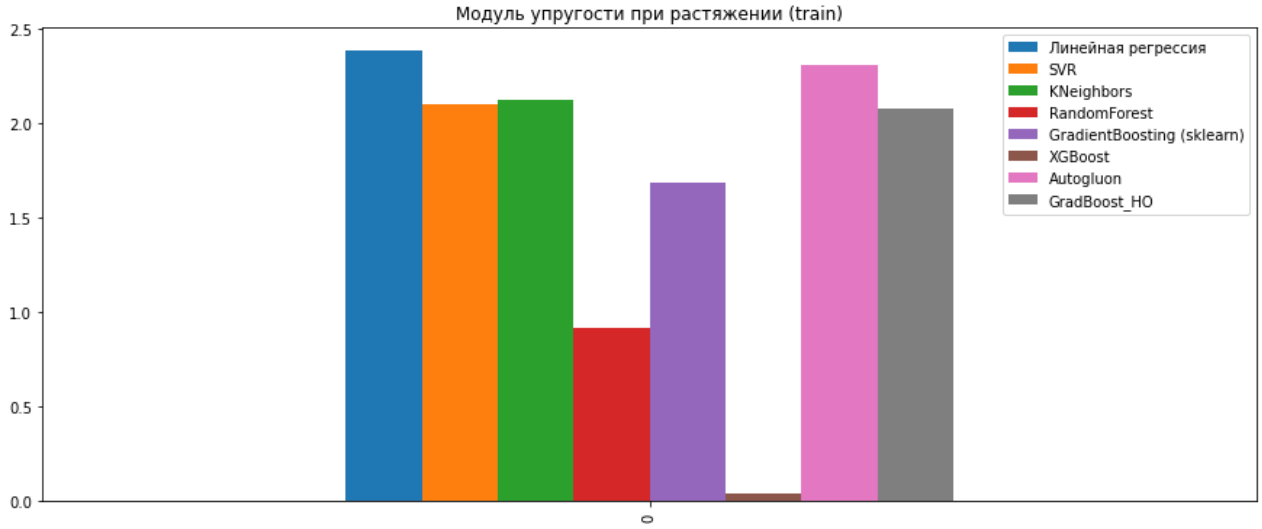

Рис. 2.2.3.MAE тренировочной выборки для «Модуль упругости при растяжении»

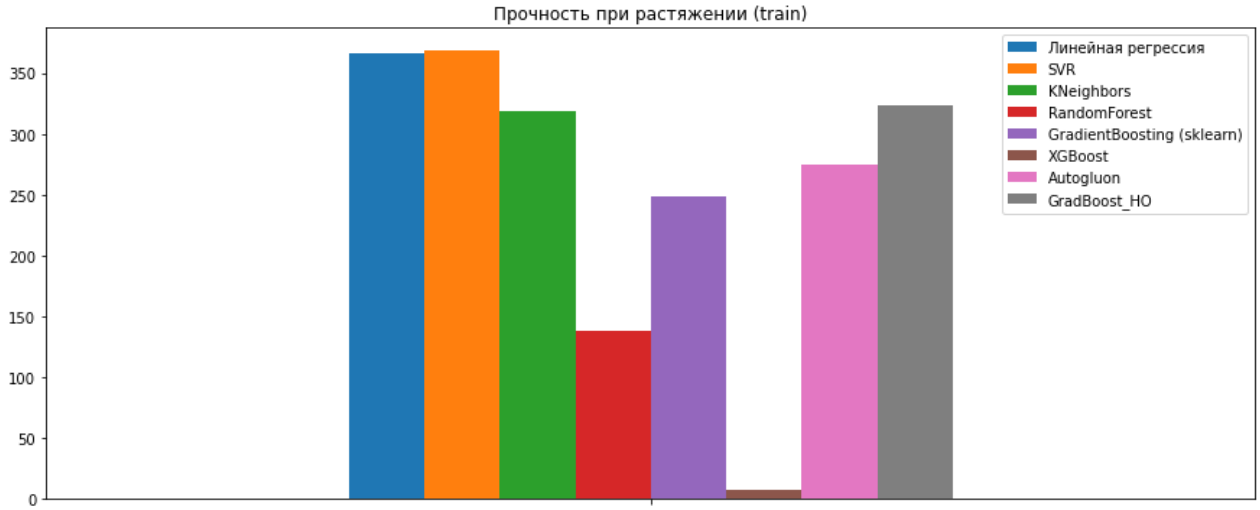

Рис. 2.2.4.MAE тренировочной выборки для «Прочность при растяжении»

2.3. Тестирование модели

Таблица 2.3.1. Результаты тестирования моделей

Метод	Модуль упругости при растяжении, MAE	Прочность при растяжении, MAE
Линейная регрессия	2.5464192980820095	370.5426179675644
Метод опорных векторов	2.6116531333051856	366.5975512794085
Метод k-ближайших соседей	2.818761217799366	405.2389030228364
Метод случайного леса	2.5959501252029167	374.8823137442501
Градиентный бустинг (sklearn)	2.691237353851188	389.8646208989257
Градиентный бустинг (xgboost)	2.836931371656709	391.90658604954945
autogluon	2.5477529120587277	373.7778693276293
Градиентный бустинг (sklearn) с лучшими гиперпараметрами	2.5832219351980394	375.38057907539417

```
for hyperparameter in hyperparameter_grid:
    model=GradientBoostingRegressor()
    grid_result = GridSearchCV(estimator = model, param_grid=hyperparameter, cv = 10,
                               scoring = 'neg_mean_absolute_error', verbose = 1,
                               n_jobs = -1, return_train_score = True)
    grid_result.fit(X, y)
    best_params.update(grid_result.best_params_)
gradboost=GradientBoostingRegressor(loss=best_params['loss'],
                                    n_estimators=best_params['n_estimators'],
                                    max_depth=best_params['max_depth'],
                                    min_samples_leaf=best_params['min_samples_leaf'],
                                    min_samples_split=best_params['min_samples_split'],
                                    max_features=best_params['max_features'])
gradboost.fit(X, y)
y_pred=gradboost.predict(X_test)
err=mean_absolute_error(y_pred, y_test)
print(f'GradBoost after hyperparam_opt, mae= {err}')
```

Рис. 2.3.1. Последовательный подбор лучших значений гиперпараметров

Последовательной перебором были выбраны лучшие гиперпараметры. Их сочетание по итогу не улучшило стандартную модель.

```
print(f'Среднее абсолютное отклонение {df["Модуль упругости при растяжении, ГПа"].mad()}')
```

```
Среднее абсолютное отклонение 2.4487129099420786
```

```
print(f'Среднее абсолютное отклонение {df["Прочность при растяжении, МПа"].mad()}')
```

```
Среднее абсолютное отклонение 368.6882107135672
```

Рис 2.3.2. Среднее абсолютное отклонение

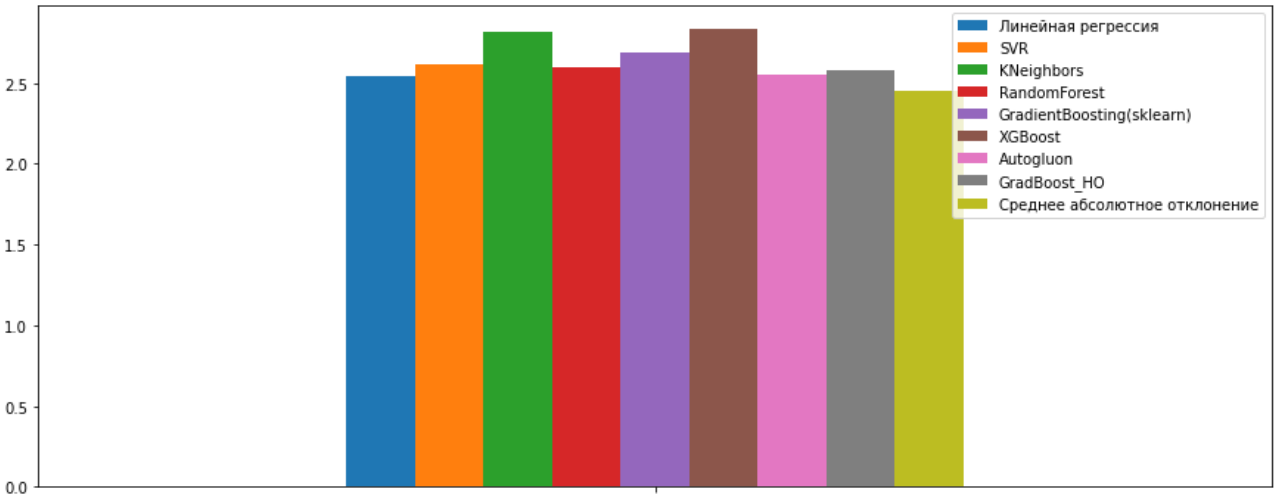

Рис. 2.3.3. MAE и MAD для «Модуль упругости при растяжении»

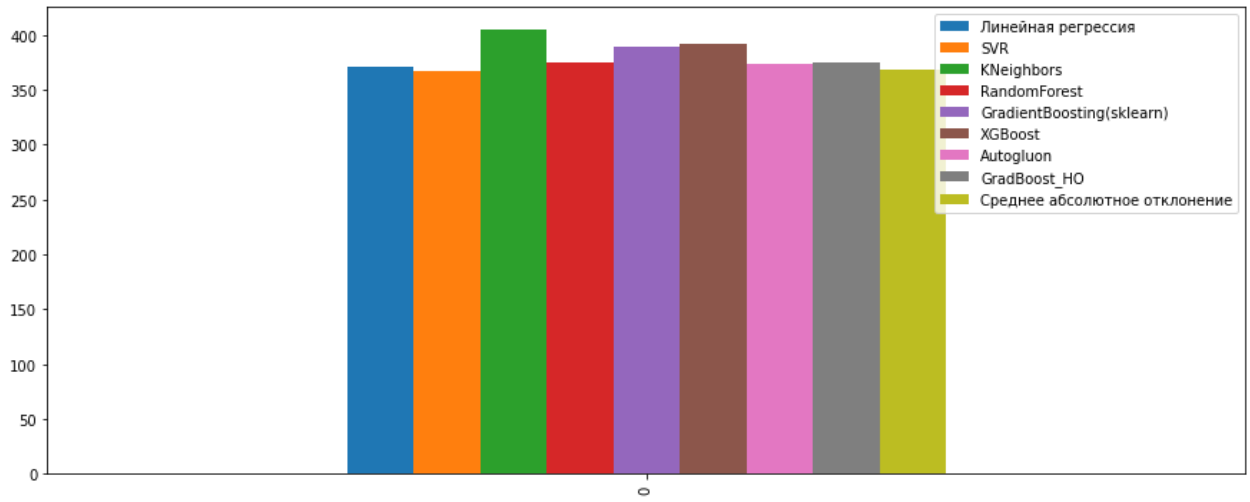

Рис. 2.3.4. MAE и MAD для «Прочность при растяжении»

В целом же результаты прогнозов достаточно плохие. Вполне уместно использовать среднее значение.

2.4. Разработка нейронной сети

Для создания нейронной сети использовалась библиотека Tensorflow.

```
X, X_test, y, y_test = train_test_split(df.drop(['Соотношение матрица-наполнитель'], axis=1), df['Соотношение матрица-наполнитель
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(X)
X = scaler.transform(X)
X_test = scaler.transform(X_test)
```

Рис. 2.4.1. Создание тренировочной и тестовой выборки их нормализация

Поскольку предполагается использовать многослойный персептрон, то можно попытаться найти лучшие гиперпараметры для модели: количество слоёв, функции активации, параметр dropout, метод оптимизации функции, размер батча.

```
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import GridSearchCV
```

Для этих целей предполагается использовать GridSearchCV из sklearn и KerasRegressor из библиотеки Keras.

```
# Задаём сетку гиперпараметров
batch_size = [4, 10, 20]
epochs = [10, 50, 100]
params = dict(batch_size=batch_size, epochs=epochs)
best_par1=param_grid(params)
```

```
Best: -1.9232882119524557 using {'batch_size': 4, 'epochs': 100}
%-2.021548045374733 (0.022856332893846833) with: {'batch_size': 4, 'epochs': 10}
%-1.932616030273732 (0.020723844141854703) with: {'batch_size': 4, 'epochs': 50}
%-1.9232882119524557 (0.020886501332792372) with: {'batch_size': 4, 'epochs': 100}
%-2.158585412396672 (0.025955599500545726) with: {'batch_size': 10, 'epochs': 10}
%-1.9633189066688896 (0.02094650804322701) with: {'batch_size': 10, 'epochs': 50}
%-1.937481182026467 (0.020910916381568553) with: {'batch_size': 10, 'epochs': 100}
%-2.2605530284521946 (0.019717617887267568) with: {'batch_size': 20, 'epochs': 10}
%-2.0192464072383003 (0.021192471306209135) with: {'batch_size': 20, 'epochs': 50}
%-1.9639918748807812 (0.020987075608635584) with: {'batch_size': 20, 'epochs': 100}
```

Рис.2. 4.2. Поиск оптимального количества эпох и параметра batch_size

```
# Подбор алгоритма
optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Nadam']
params = dict(opt=optimizer)

best_par2=param_grid(params)
```

```
Best: -2.385381023900114 using {'opt': 'RMSprop'}
%-2.4023782394333364 (0.018513587387797348) with: {'opt': 'SGD'}
%-2.385381023900114 (0.020283847364619873) with: {'opt': 'RMSprop'}
%-2.4114794978171936 (0.006609999072938584) with: {'opt': 'Adagrad'}
%-2.408458623971145 (0.02166126516243861) with: {'opt': 'Adadelta'}
%-2.398694554476326 (0.022320008454315754) with: {'opt': 'Adam'}
%-2.399611435750899 (0.018806755433834272) with: {'opt': 'Nadam'}
```

Рис. 2.4.3. Поиск лучшего алгоритма обучения

```
# Подбор количества слоёв
layers = [[8],[16, 4],[32, 8, 3],[12, 6, 3], [64, 64, 3], [128, 64, 16, 3]]
params = dict(lyrs=layers)

best_par3=param_grid(params)
```

```
Best: -2.3296137128343557 using {'lyrs': [12, 6, 3]}
%-2.415844247325594 (0.03430937813971773) with: {'lyrs': [8]}
%-2.367029073551433 (0.04613587132184879) with: {'lyrs': [16, 4]}
%-2.441432704752371 (0.013817024077156893) with: {'lyrs': [32, 8, 3]}
%-2.3296137128343557 (0.08630109322176374) with: {'lyrs': [12, 6, 3]}
%-2.3910099559841584 (0.09692103160006162) with: {'lyrs': [64, 64, 3]}
%-2.457231443279937 (0.06681273667712599) with: {'lyrs': [128, 64, 16, 3]}
```

Рис. 2.4.4. Поиск оптимального количества слоёв и нейронов на нём

```
# Подбор функций активации
activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']
params = dict(act=activation)

best_par4=param_grid(params)
```

```
Best: -2.09210093749614 using {'act': 'softplus'}
%-2.4018848145791387 (0.02943334486277002) with: {'act': 'softmax'}
%-2.09210093749614 (0.08437513358850252) with: {'act': 'softplus'}
%-2.2978935442287205 (0.10442327001160373) with: {'act': 'softsign'}
%-2.3401837467706628 (0.040126372956336254) with: {'act': 'relu'}
%-2.3332479171255276 (0.10726824149748491) with: {'act': 'tanh'}
%-2.2441279636371774 (0.08476932016398807) with: {'act': 'sigmoid'}
%-2.331727761219751 (0.16696023039771082) with: {'act': 'hard_sigmoid'}
%-2.195773514770851 (0.08766898949722653) with: {'act': 'linear'}
```

Рис. 2.4.5. Подбор функции активации для скрытых слоёв

```
# Подбор dropout-ов
drops = [0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5]
params = dict(dr=drops)

best_par5=param_grid(params)
```

```
Best: -2.3893332335243382 using {'dr': 0.5}
%-2.4095063048303014 (0.018972548776840428) with: {'dr': 0.0}
%-2.4100775696200465 (0.021887100742393358) with: {'dr': 0.01}
%-2.4041158417503845 (0.025130203077815667) with: {'dr': 0.05}
%-2.398047267944196 (0.036482842240976324) with: {'dr': 0.1}
%-2.4024936162112946 (0.02421669290930377) with: {'dr': 0.2}
%-2.398429464201918 (0.03620154980686437) with: {'dr': 0.3}
%-2.3893332335243382 (0.014853220277022839) with: {'dr': 0.5}
```

Рис. 2.4.6. Подбор параметра dropout

```
# Выбранные параметры для нейросети
print(f'Слои: {best_lyrs}')
print(f'Dropout: {best_dr}')
print(f'Функция активации: {best_act}')
print(f'Количество эпох: {best_epoch}')
print(f'Размер батча: {best_batch_size}')
print(f'best optimizer: {best_opt}')
```

```
Слои: [12, 6, 3]
Dropout: 0.0
Функция активации: softplus
Количество эпох: 100
Размер батча: 4
best optimizer: Nadam
```

Рис. 2.4.7. Результат подбора гиперпараметров

После поиска гиперпараметров, создадим нейросеть на их основе. Обучим нейросеть. Выведем график обучения, по которому можно судить о переобучении. Если val_loss начинает расти, это говорит о переобучении.

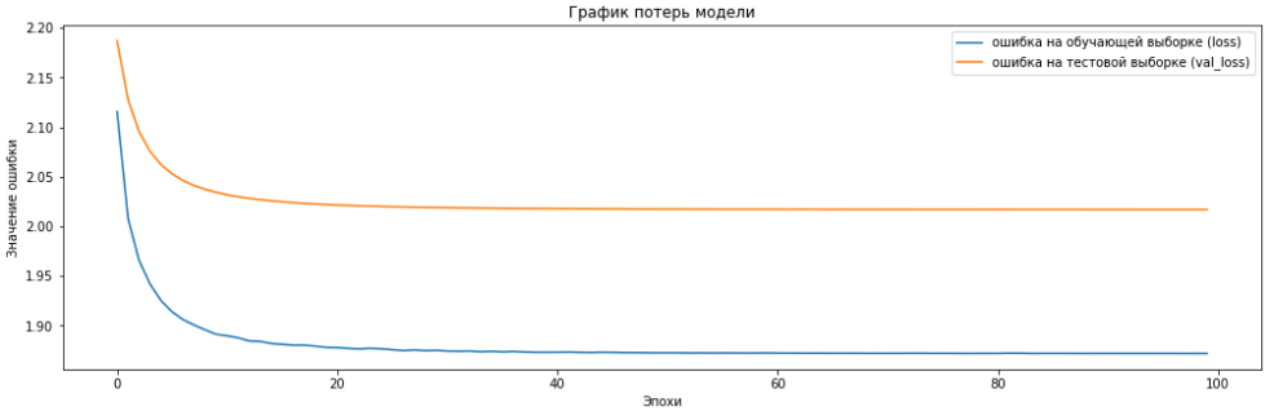

Рис. 2.4.8. Параметры loss и val_loss

Применим обученную модель на тестовой выборке.

```
# оценим модель. Результат mae для тестовой выборки
scores = model.evaluate(X_test, y_test)
scores
```

```
9/9 [==============================] - 0s 3ms/step - loss: 1.9986

1.9986492395401
```

```
# Среднее абсолютное отклонение
df['Соотношение матрица-наполнитель'].mad()
```

```
0.7159293950955805
```

Рис. 2.4.9. MAE на тестовой выборке и MAD

В результате ошибка больше среднее абсолютное отклонение. Как видно из графика, скорее всего проблема не в переобучении.

Для дальнейшей разработки приложения для прогнозирования матрицы-наполнителя необходимо сохранить модель и нормализатор.

```
model.save('saved_model/my_model')
```

```
joblib.dump(scaler, "scaler")
```

```
['scaler']
```

2.5. Разработка приложения

Для написания приложения выбрана нейронная сеть из предыдущего раздела и библиотека Tkinter. Для нормализации данных взят нормализатор, который нормализовал данные для нейронной сети.

```
import joblib
scaler = joblib.load('scaler')
```

Рис. 2.5.1. Загрузка нормализатора

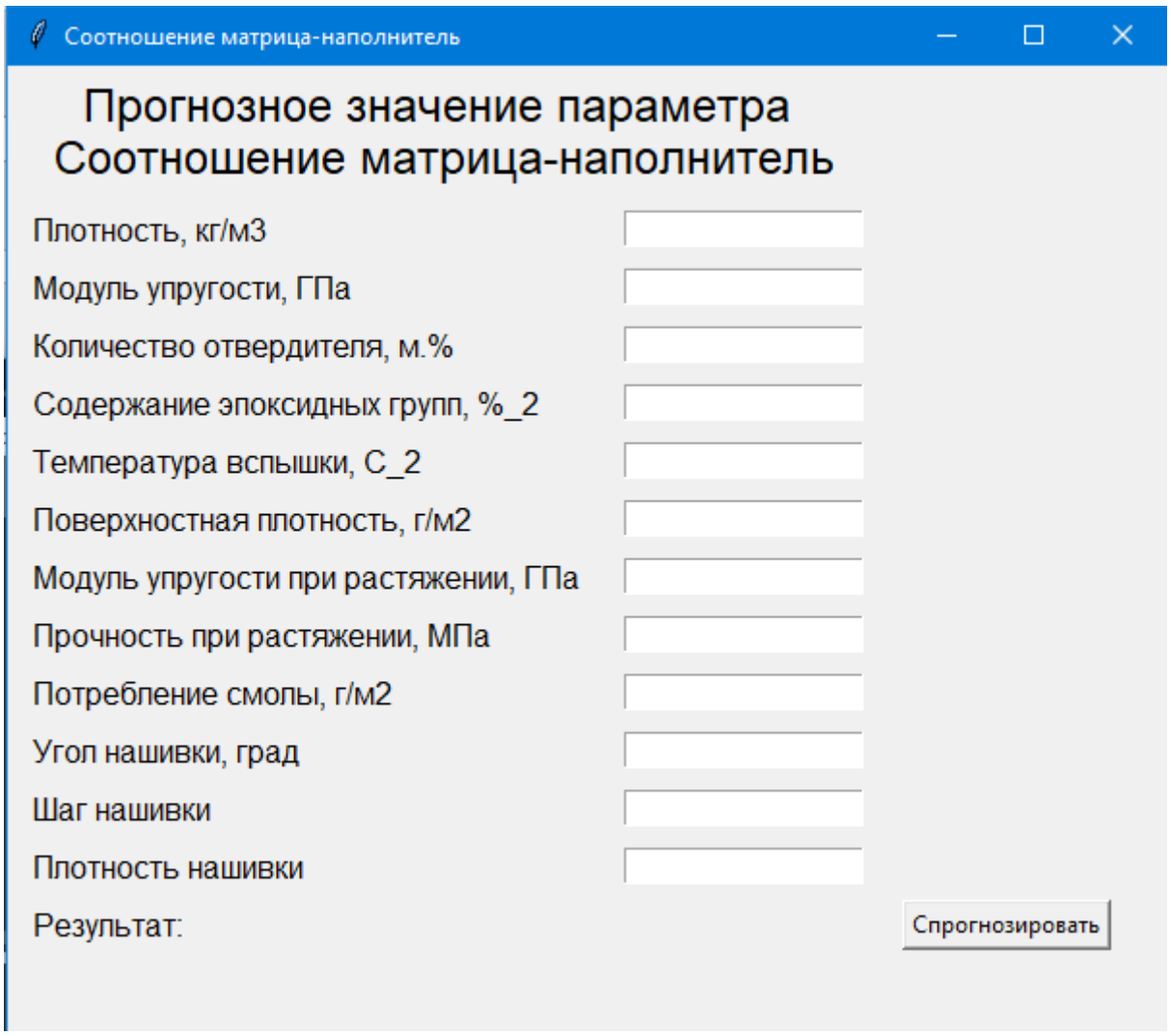

Рис. 2.5.2. Графический интерфейс приложения

После заполнения всех полей пользователь получает прогноз значения параметра матрица-наполнитель.

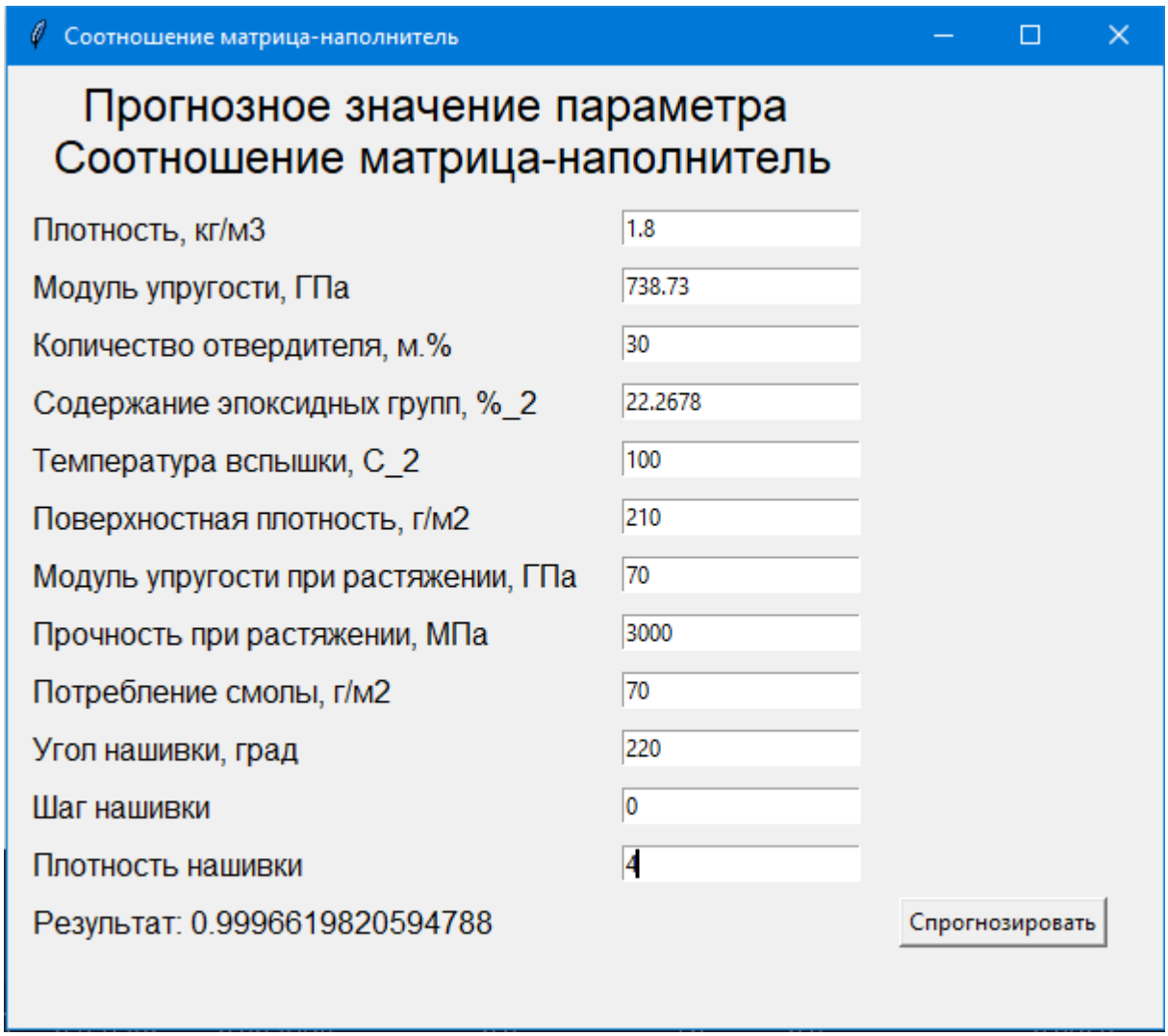

Рис. 2.5.3. Результат работы программы

2.6. Работа с удаленным репозиторием

Был создан удаленный репозиторий на github:

https://github.com/vasileva1979/Composites

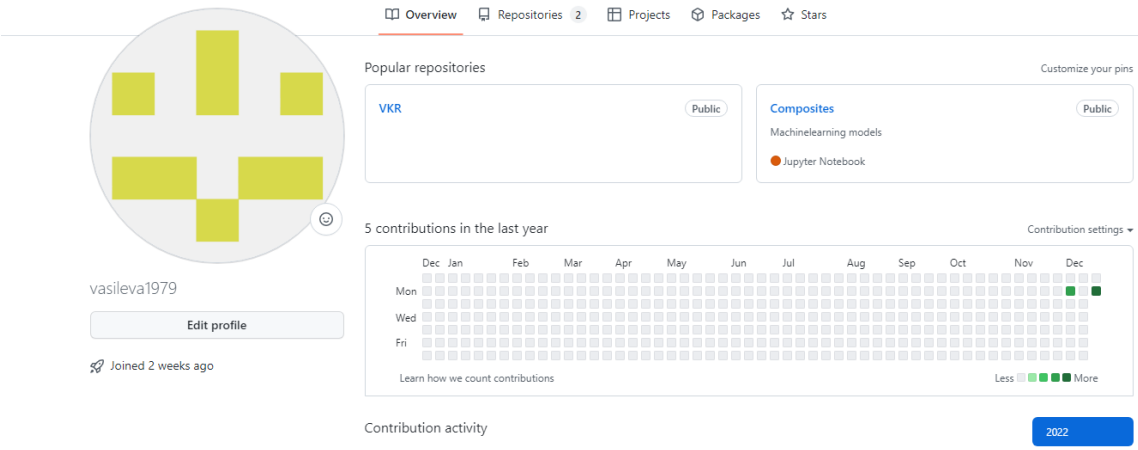

Рис. 2.6.1. Созданный профиль на github

Рис. 2.6.2. Readme-файл

2.7. Заключение

По результатам проведённой работы можно сделать ряд выводов о датасете и методах машинного обучения применённых к нему.

Распределения имеющиеся параметров в датасете близки к нормальным. Корреляционная матрица показывает отсутствие коллинеарных величин и какой-либо достаточно значимой корреляции. Имеется ряд значений, которые можно обозначить выбросами.

Был обучен ряд моделей, но результаты имеют достаточно большую среднюю абсолютную ошибку, которая для некоторых из методов хуже, чем среднее абсолютное отклонение. Лучшие результаты среди сравниваемых моделей показал autogluon, немного отстаёт от метод опорных векторов. У нейронной сети тоже достаточно достаточно большая ошибка, несмотря на гиперпараметрическую оптимизацию.

Для улучшения моделей, по всей видимости, их необходимо обучать на датасете с другим набором данных, с большим числом входных параметров.

2.8. Библиографический список

1. Документация библиотеки numpy: – Режим доступа https://numpy.org/doc/stable/. (дата обращения:8.12.2022)
2. Документация библиотеки pandas: – Режим доступа https://pandas.pydata.org/docs/ (дата обращения: 8.12.2022)
3. Документация библиотеки matplotlib: – Режим доступа https://matplotlib.org/stable/tutorials/(дата обращения: 11.12.2022)
4. Документация библиотеки seaborn: – Режим доступа https://seaborn.pydata.org/. (дата обращения: 11.12.2022)
5. Документация библиотеки xgboost: – Режим доступа https://xgboost.readthedocs.io/en/stable/. (дата обращения: 12.12.2022)
6. Документация библиотеки autogluon: – Режим доступа https://auto.gluon.ai/dev/tutorials. (дата обращения: 11.12.2022)
7. Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, Alexander Smola AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data Режим доступа https://arxiv.org/pdf/2003.06505.pdf . (дата обращения: 11.12.2022)
8. Документация библиотеки tensorflow: – Режим доступа https://www.tensorflow.org/api_docs. (дата обращения: 11.12.2022)
9. A Complete Machine Learning Project Walk-Through in Python: Part One: – Режим доступа https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-one-c62152f39420
10. A Complete Machine Learning Walk-Through in Python: Part Two: – Режим доступа https://towardsdatascience.com/a-complete-machine-learning-project-walk-through-in-python-part-two-300f1f8147e2
11. A Complete Machine Learning Walk-Through in Python: Part Three: – Режим доступа https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-three-388834e8804b
12. Trevor Hastie, Robert Tibshirani, Jerome Friedman The elements of statistical learning Second edition.: – Springer.

Приложение 1

Подробный план работы:

1. Загрузка датасетов

1.1. Удаление ненужных столбцов

1.2. Объединение датасетов по индексу по типу INNER.

2. Разведочный анализ данных:

2.1. Построение гистограмм

2.2. Построение диаграмм рассеяния

2.3. Построение «ящиков с усами»

2.4. Вывод некоторых статистических данных по датасету

3. Предобработка данных

3.1. Нахождение с помощью межквартильных расстояний выбросов

3.2. Удаление выбросов

4. Обучение нескольких моделей машинного обучения

4.1. Разбиение данных на обучающие и тестовые

4.2. Нормализация данных

4.3. Обучение моделей

4.4. Сравнение результатов

5. Нейронная сеть для рекомендации соотношения матрица-наполнитель.

5.1. Разбиение данных на обучающие и тестовые

5.2. Нормализация данных

5.3. Выбор гиперпараметров

5.4. Гиперпараметрическая оптимизация

5.5. Выбор лучших гиперпараметров для нейросети

5.6. Построение модели

5.7. Анализ результатов

6. Создание приложения

6.1. Выбор модели

6.2. Создание приложения

6.3. Тестирование приложения

7. Создание удалённого репозитория и загрузка результатов работы в него

7.1. Загрузка данных

7.2. Написание Readme