МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**по курсу**

«Data Science»

|  |  |
| --- | --- |
| Слушатель | Холманский Евгений Игоревич |

Содержание

Введение3

1. Аналитическая часть4

1.1 Постановка задачи4

1.2 Описание используемых методов4

1.3 Разведочный анализ данных6

1. Практическая часть12

2.1 Предобработка данных12

2.2 Разработка и обучение моделей14

2.3 Нейронная сеть21

2.4 Создание удаленного репозитория и загрузка результатов работы27

Заключение28

Библиографический список29

Введение

Композиционные материалы – это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т. е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита – железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства.

Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично.

Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

1. Аналитическая часть

# Постановка задачи

1. Провести разведочный анализ данных двух датасетов: с базальтопластиками (X\_bp.xlsx) и нашивками углепластикавыми (X\_nup.xlsx). Объединить их методом inner.
2. Нарисовать для каждой переменной из объединенного датасета гистограммы распределения, диаграммы боксплотов (ящик с усами), попарные графики рассеяния точек.
3. Получить для каждого столбца среднее, медианное значение, провести анализ и исключение выбросов, проверить наличие пропусков.
4. Провести предобработку данных в части удаления выбросов и нормализации.
5. Обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении.

Примечание – Для тестирования моделей использовать 30% данных, обучения проводить на 70% данных.

1. Провести поиск гиперпараметров при построении моделей с помощью поиска по сетке с перекрестной проверкой, количество блоков 10.
2. Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель.
3. Оценить точность модели на тренировочном и тестовом датасете.
4. Создать репозиторий на GitHub и разместить там код исследования. Оформить файл README.

# Описание используемых методов

Выполнение данного исследования произведено с помощью языка программирования Python в среде разработке Jupiter Notebook.

В контексте задачи для прогнозирования модуля упругости при растяжении и прочности при растяжении проанализированы следующие модели:

1. Линейная регрессия.

Линейная регрессия (англ. Linear regression) – используемая в статистике регрессионная модель зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости.

Модель линейной регрессии является часто используемой и наиболее изученной в эконометрике. А именно изучены свойства оценок параметров, получаемых различными методами при предположениях о вероятностных характеристиках факторов, и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями. С эконометрической точки зрения более важное значение имеет линейность по параметрам, чем линейность по факторам модели.

1. Random forest (Случайный лес).

Random forest (англ. — «случайный лес») – алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Тин Кам Хо. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

1. Метод К-ближайших соседей.

Метод К-ближайших соседей (англ. *k-nearest neighbors algorithm*, k-NN) – метрический алгоритм для автоматической классификации объектов или регрессии.

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны. В случае использования метода для регрессии, объекту присваивается среднее значение по k 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.

# Разведочный анализ данных

После выполнения операций по загрузке и объединению исходных датасетов проведем анализ основных свойств данных, нахождение в них общих закономерностей, распределений и аномалий, построение начальных моделей, с использованием инструментов визуализации.

1.3.1 Информация о датасете:

bp\_nup\_df.info()

При выполнении данной ячейки на экран выводятся данные приведенные на рисунке 1.

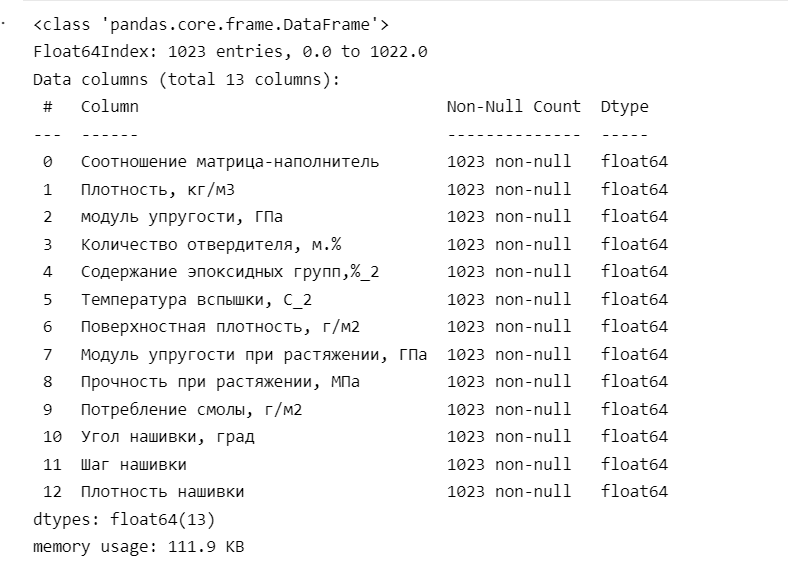


Рисунок 1 – Вывод ячейки с параметрами данных датасета

Согласно данного рисунка видно, что в датасет имеет размерность 1023 строки и 13 столбцов, пропущенные значения отсутствуют, все данные имеют тип float64.

1.3.2 Получим основные статистические данные:

bp\_nup\_df.describe().round(2).T

Вывод ячейки представлен на рисунке 2.

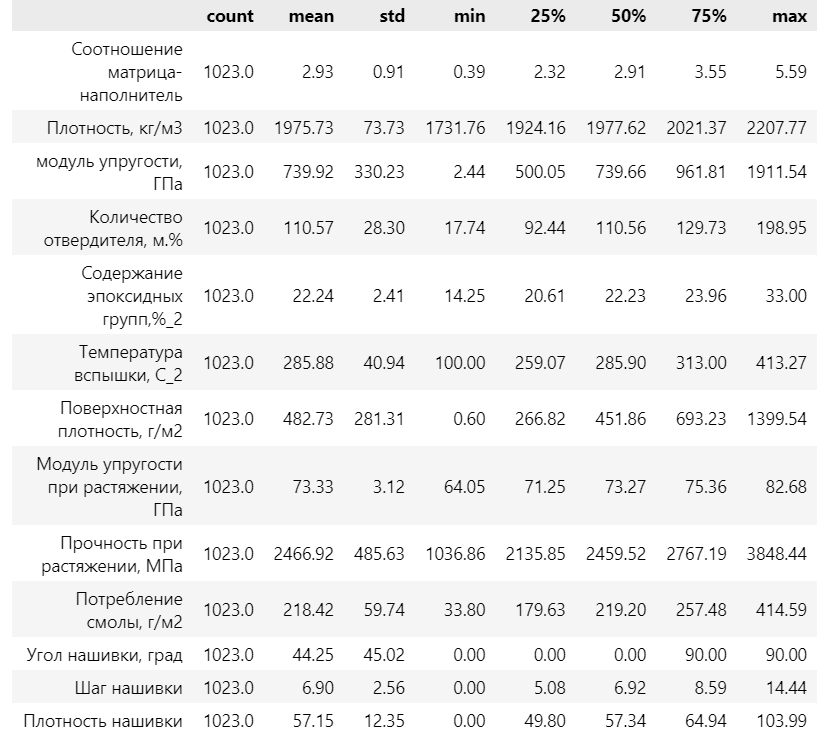


Рисунок 2 – Статистические данные датасета

1.3.3 Построение гистограмм:

plt.figure(figsize=(35,35))

c = 1

for col in bp\_nup\_df.columns:

    plt.subplot(5, 5, c)

    sns.histplot(data = bp\_nup\_df[col], kde=True)

    plt.ylabel(None)

    plt.title(col, size = 20)

    c+=1

Из данных приведенных на рисунке 3 видно, что в основном все гистограммы имеют нормальное распределение, за исключением угла нашивки, параметры которого принимают только 2 значения.

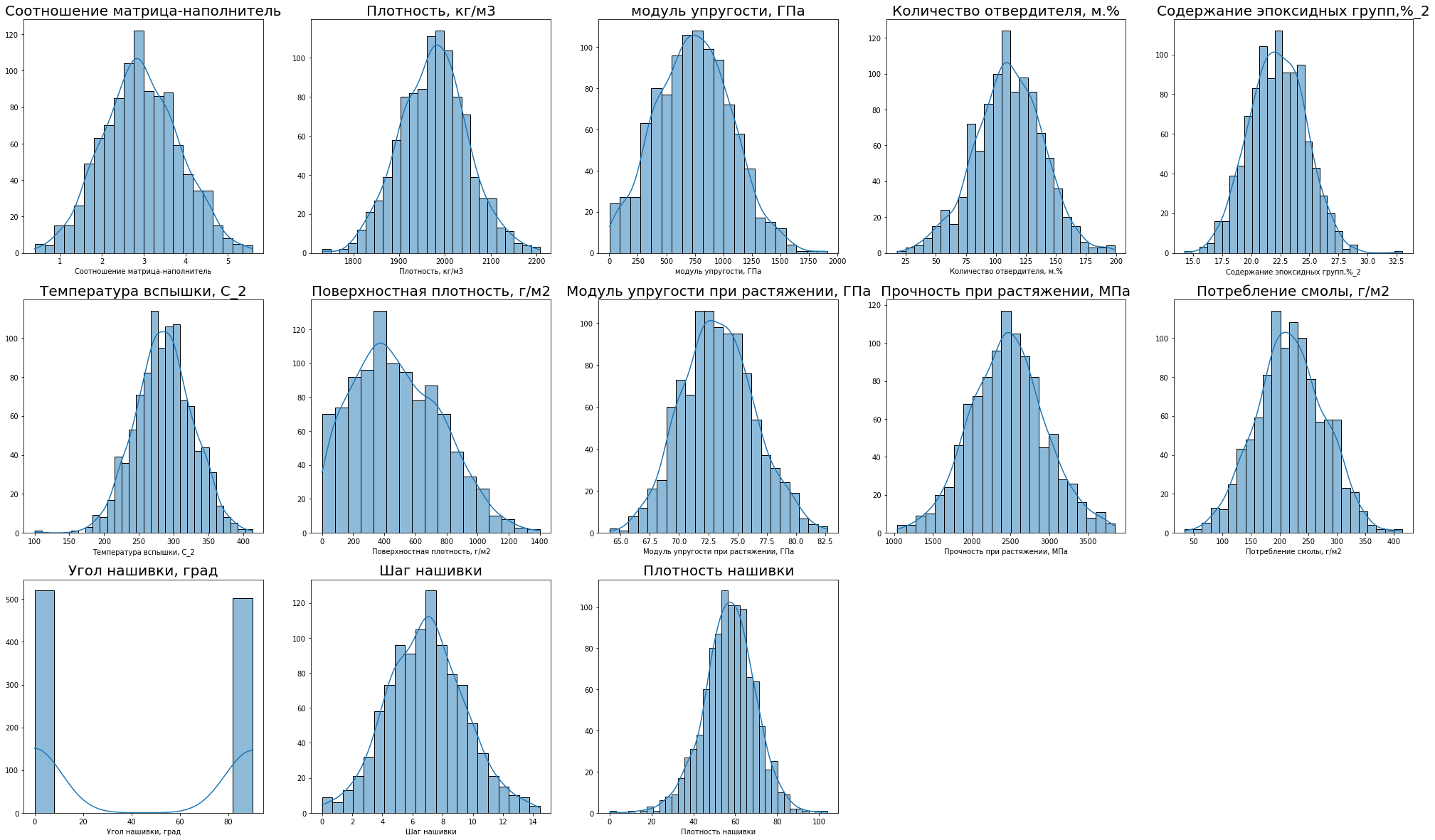


Рисунок 3 – Гистограммы данных

1.3.4 Построение боксплотов (ящиков с усами):

plt.figure(figsize=(35,35))

c = 1

for col in bp\_nup\_df.columns:

    plt.subplot(5, 5, c)

    sns.boxplot(color= "orange", data = bp\_nup\_df, x=bp\_nup\_df[col], fliersize=8,  
    linewidth=2)

    plt.ylabel(None)

    plt.title(col, size = 20)

    c+=1

Ящики с усами представленные на рисунке 4 наглядно отображают медиану, нижний и верхний квартили, минимальное и максимальное значение выборки и выбросы.

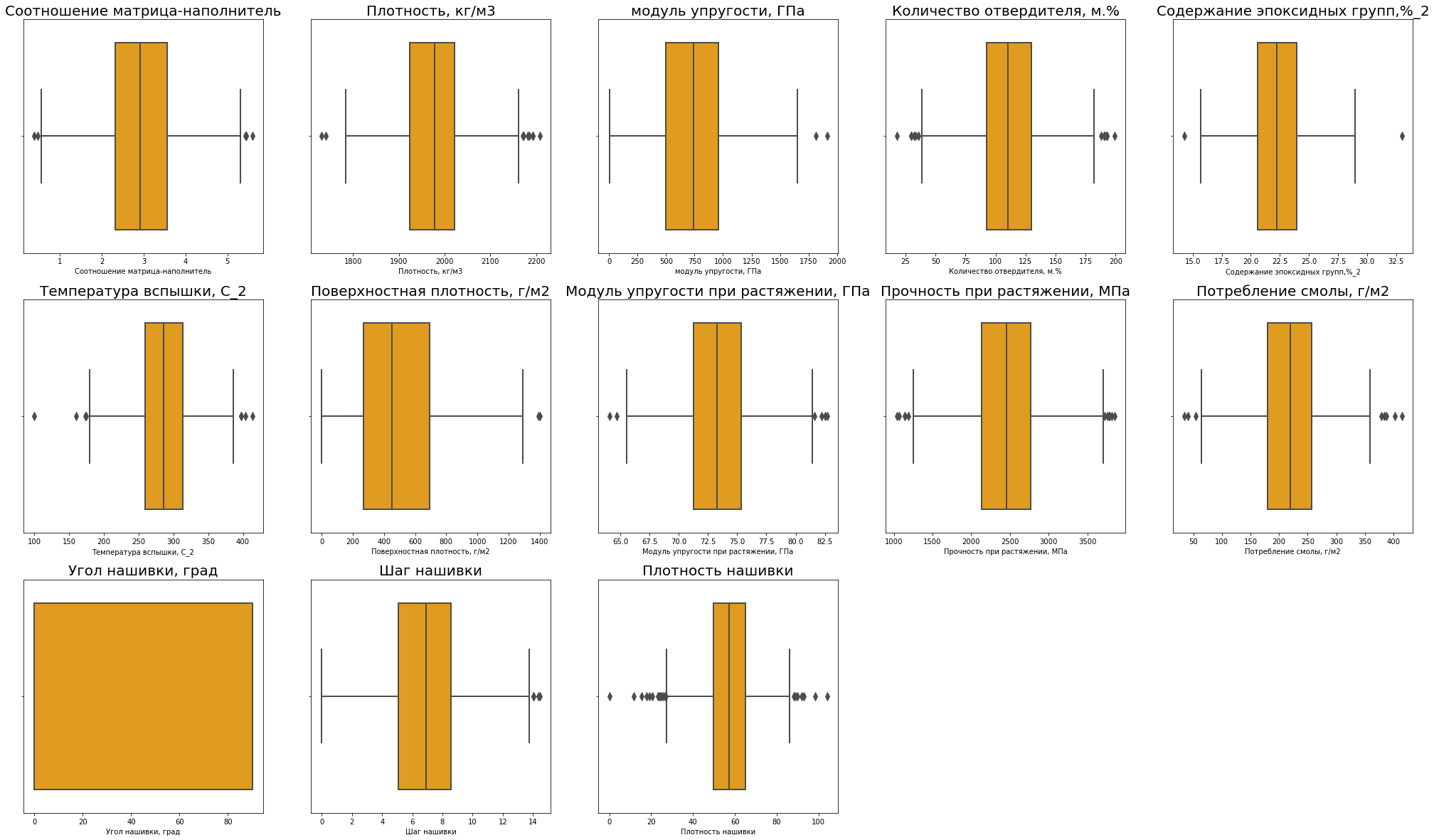


Рисунок 4 – Боксплоты (ящики с усами)

1.3.5 Определение взаимосвязей между переменными с помощью графика рассеяния точек:

sns.pairplot(bp\_nup\_df, height=3.5)

Приведенный на рисунке 5 график попарного рассеяния точек, указывает на отсутствие корреляционных зависимости между переменными.

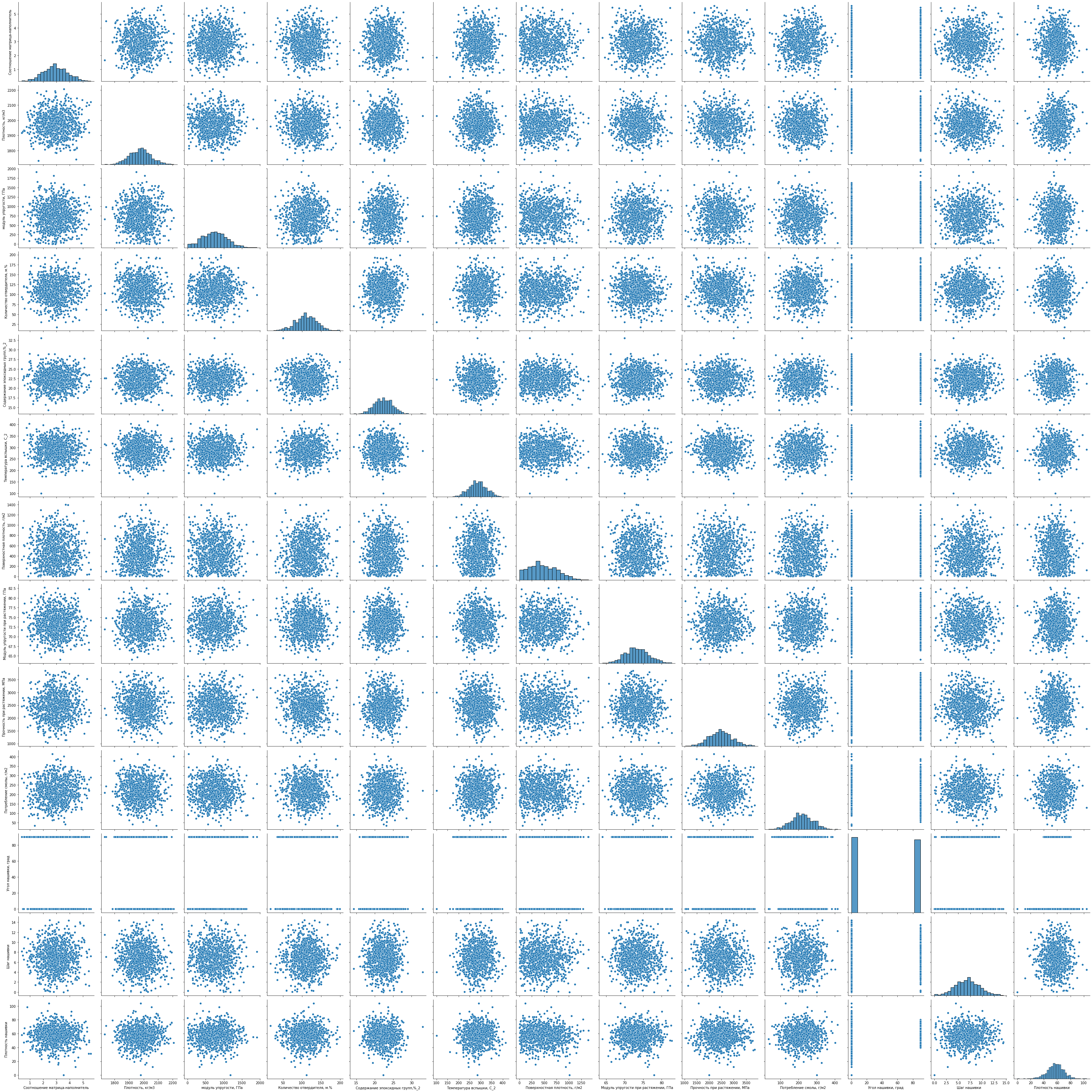


Рисунок 5 – Попарные графики рассеяния точек

1.3.6 Определение взаимосвязей между переменными с помощью тепловой карты:

mask = np.triu(bp\_nup\_df.corr())

f, ax = plt.subplots(figsize=(11, 9))

sns.heatmap(bp\_nup\_df.corr(), mask=mask, annot=True, square=True, cmap='coolwarm')

plt.xticks(rotation=45, ha='right')

plt.show()

Тепловая карта взаимосвязи переменных, представленная на рисунке 6, наглядно демонстрирует коэффициент корреляции между переменными. Самая сильная взаимосвязь у параметров плотность нашивки и угол нашивки – 0,11, но она недостаточно велика и можно считать связь отсутствующей.

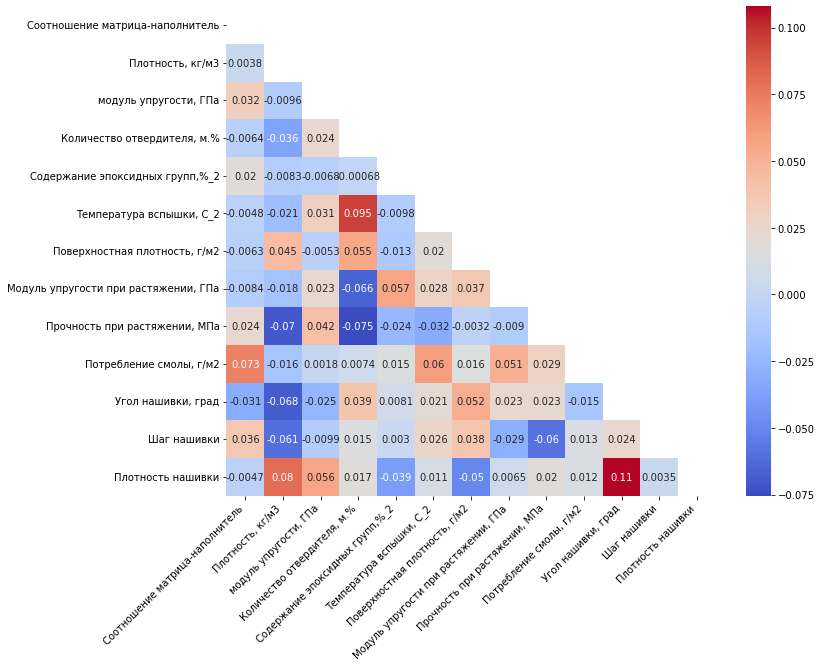


Рисунок 6 – Тепловая карта взаимосвязей переменных

1. Практическая часть

# Предобработка данных

2.1.1 Удаление выбросов

for col in bp\_nup\_df.columns:

    q75,q25 = np.percentile(bp\_nup\_df.loc[:,col],[75,25])

    intr\_qr = q75-q25

    max = q75+(1.5\*intr\_qr)

    min = q25-(1.5\*intr\_qr)

    bp\_nup\_df.loc[bp\_nup\_df[col] < min,col] = np.nan

    bp\_nup\_df.loc[bp\_nup\_df[col] > max,col] = np.nan

clean\_outliers\_bp\_nup\_df = bp\_nup\_df.dropna(axis=0)

clean\_outliers\_bp\_nup\_df.info()

Произведем удаление выбросов по методу 3 сигм и выведем информацию о датасете (рисунок 7).

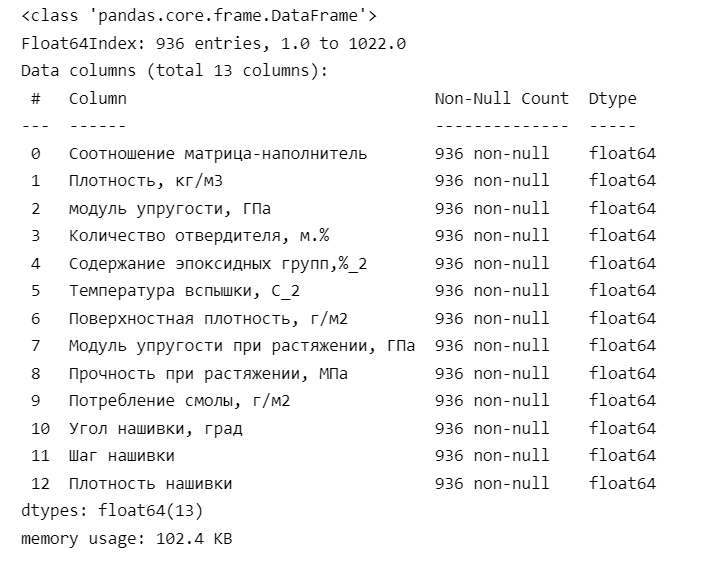


Рисунок 7 – Датасет очищенный от выбросов

Из рисунка 7 видно, что было удалено 87 значений, не попадающий в диапазон 3 сигм.

2.1.2 Нормализация данных

В связи с чувствительностью моделей машинного обучения к дисбалансу данных, необходимо произвести их нормализацию (сжатие их в диапазоне от 0 до 1). Нормализованный датасет приведен на рисунке 8.

min\_max\_scaler = MinMaxScaler()

df\_norm = pd.DataFrame(min\_max\_scaler.fit\_transform(clean\_outliers\_bp\_nup\_df),

                       columns=clean\_outliers\_bp\_nup\_df.columns,

                       index=clean\_outliers\_bp\_nup\_df.index)

df\_norm

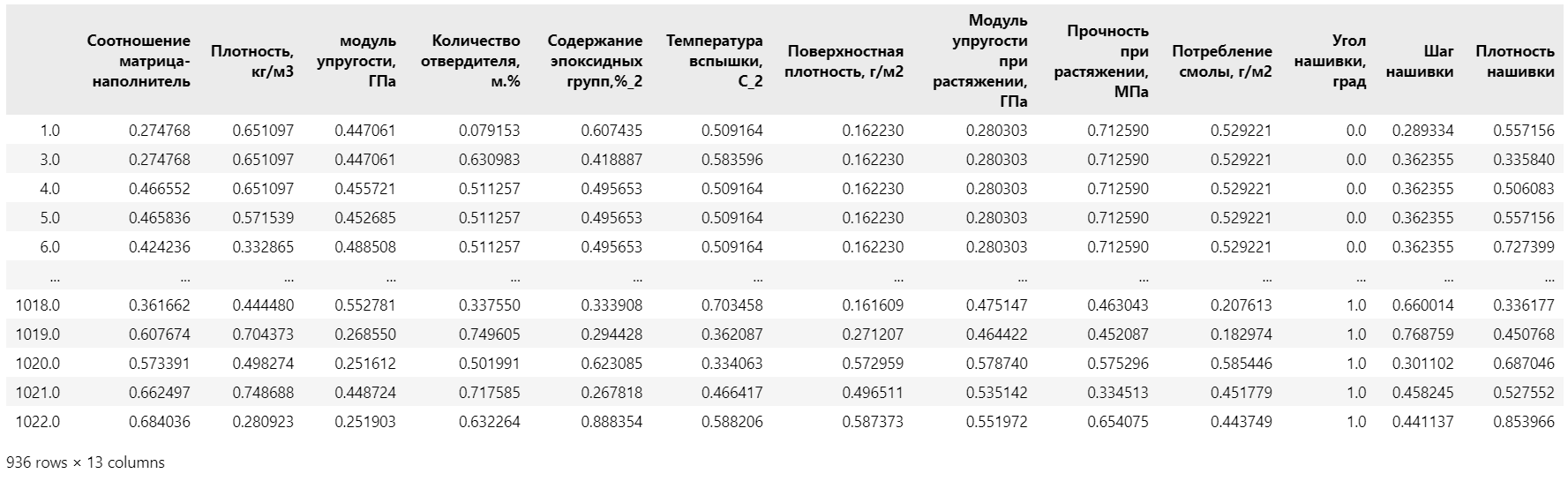


Рисунок 8 – Датасет с нормализованными данными

Статистические данных для нормализованного датасета и его боксплоты приведены на рисунках 9 и 10 соответственно.

df\_norm.describe().round(2).T



Рисунок 9 – Статистические данные нормализованного датасета

boxplot = df\_norm.boxplot(rot=90, figsize=(16, 5), patch\_artist=True, boxprops=dict(facecolor="orange", color="black"))

plt.xticks(rotation=45, ha='right')

plt.show()

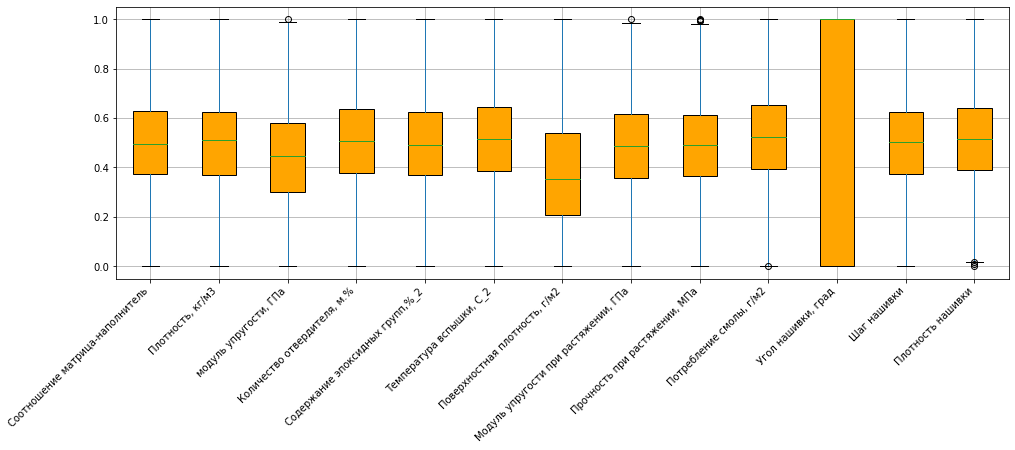


Рисунок 10 – Боксплоты (ящики с усами) нормализованного датасета

# Разработка и обучение моделей

Разработка и обучение моделей, поиск гиперпараметров в данном разделе приведены только для значения модуль упругости при растяжении. Для значения прочность при растяжении все операции выполняются аналогичным образом. В репозитории на GitHub можно ознакомиться с кодом исследования данного параметра.

Для подготовки к разработке и обучению моделей необходимо произвести разбиение данных на обучающую и тестовую выборки:

x\_upr = df\_norm.drop(['Модуль упругости при растяжении, ГПа'], axis=1)

x\_pr = df\_norm.drop(['Прочность при растяжении, МПа'], axis=1)

y\_upr = df\_norm[['Модуль упругости при растяжении, ГПа']]

y\_pr = df\_norm[['Прочность при растяжении, МПа']]

X\_train\_upr, X\_test\_upr, y\_train\_upr, y\_test\_upr = train\_test\_split(x\_upr, y\_upr, test\_size=0.3, random\_state=1)

X\_train\_pr, X\_test\_pr, y\_train\_pr, y\_test\_pr = train\_test\_split(x\_pr, y\_pr,  
test\_size=0.3, random\_state=1)

В данном случае выборки созданы для модуля упругости при растяжении и для прочности при растяжении. Размер тестовой данных составляет 30%

2.2.1 Линейная регрессия

Создаем линейный регрессор и задаем параметры

lr = LinearRegression()

lr\_params = {

    'fit\_intercept' : ['True', 'False']

    }

Производим обучение модели

GSCV\_lr\_upr = GridSearchCV(lr, lr\_params, n\_jobs=-1, cv=10)

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

GSCV\_lr\_upr.best\_params\_

{'fit\_intercept': 'True'}

Если значение fit\_intercept=True, точка пересечения по оси y будет определяться линией наилучшего соответствия.

lr\_upr = GSCV\_lr\_upr.best\_estimator\_

print(f'R2-score LR для модуля упругости при растяжении: {lr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)}')

R2-score LR для модуля упругости при растяжении: -0.021

Полученное значение коэффициента детерминации используется для оценки производительности моделей. Это величина вариации выходного зависимого атрибута, которую можно спрогнозировать на основе входной независимой переменной. Он используется для проверки того, насколько хорошо наблюдаемые результаты воспроизводятся моделью, в зависимости от отношения общего отклонения результатов, описываемых моделью.

Рассчитаем значение абсолютной ошибки:

lr\_upr\_result = pd.DataFrame({

    'Model': 'LinearRegression\_upr',

    'MAE': mean\_absolute\_error(y\_test\_upr, lr\_upr.predict(X\_test\_upr)),

    'R2 score': lr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)

    }, index=['Модуль упругости при растяжении'])

Создадим таблицу с ошибками для каждого метода, и по мере исследования будем добавлять в неё значения (Рисунок 11):

models = pd.DataFrame()

models = pd.concat([models, lr\_upr\_result])

lr\_upr\_result

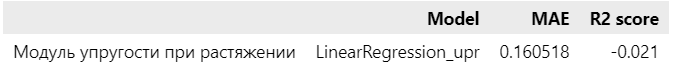


Рисунок 11 – Вывод значений ошибок для модели линейной регрессии

2.2.2 Метод к-ближайших соседей

Создаем регрессию:

knr = KNeighborsRegressor()

knr\_params = {'n\_neighbors' : range(1, 301, 2),

              'weights' : ['uniform', 'distance'],

              'algorithm' : ['auto', 'ball\_tree', 'kd\_tree', 'brute']

             }

Обучаем модель:

GSCV\_knr\_upr = GridSearchCV(knr, knr\_params, n\_jobs=-1, cv=10)

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

GSCV\_knr\_upr.best\_params\_

{'algorithm': 'auto', 'n\_neighbors': 59, 'weights': 'uniform'}

knr\_upr = GSCV\_knr\_upr.best\_estimator\_

print(f'R2-score KNR для модуля упругости при растяжении: {knr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)}')

R2-score KNR для модуля упругости при растяжении: -0.005

knr\_upr\_result = pd.DataFrame({

    'Model': 'KNeighborsRegressor\_upr',

    'MAE': mean\_absolute\_error(y\_test\_upr, knr\_upr.predict(X\_test\_upr)),

    'R2 score': knr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)

    }, index=['Модуль упругости при растяжении'])

models = pd.concat([models, knr\_upr\_result])

knr\_upr\_result



Рисунок 12 – Вывод значений ошибок для модели к=ближайших соседей

2.2.3 Случайный лес

rfr = RandomForestRegressor()

rfr\_params = {

    'n\_estimators' : range(10, 1000, 10),

    'criterion' : ['squared\_error', 'absolute\_error', 'poisson'],

    'max\_depth' : range(1, 7),

    'min\_samples\_split' : range(20, 50, 5),

    'min\_samples\_leaf' : range(2, 8),

    'bootstrap' : ['True', 'False']

    }

Производим обучение модели:

RSCV\_rfr\_upr = RandomizedSearchCV(rfr, rfr\_params, n\_jobs=-1, cv=10, verbose=4)

RSCV\_rfr\_upr.fit(X\_train\_upr, np.ravel(y\_train\_upr))

RSCV\_rfr\_upr.best\_params\_

Fitting 10 folds for each of 10 candidates, totalling 100 fits

{'n\_estimators': 470, 'min\_samples\_split': 45, 'min\_samples\_leaf': 3, 'max\_depth': 2, 'criterion': 'squared\_error', 'bootstrap': 'True'}

rfr\_upr = RSCV\_rfr\_upr.best\_estimator\_

print(f'R2-score RFR для модуля упругости при растяжении: {rfr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)}')

R2-score RFR для модуля упругости при растяжении: -0.019

rfr\_upr\_result = pd.DataFrame({

    'Model': 'RandomForestRegressor\_upr',

    'MAE': mean\_absolute\_error(y\_test\_upr, rfr\_upr.predict(X\_test\_upr)),

    'R2 score': rfr\_upr.score(X\_test\_upr, y\_test\_upr).round(3)

    }, index=['Модуль упругости при растяжении'])

models = pd.concat([models, rfr\_upr\_result])

rfr\_upr\_result



Рисунок 13 – Вывод значений ошибок для модели случайного леса

Исходя из анализа значений абсолютной ошибки и коэффициента детерминации, приведенных на рисунке 14, можно сделать вывод, что выбранные для исследования модели справились с задачей плохо. Модели прогнозируют значения хуже, чем если бы было просто усреднение.

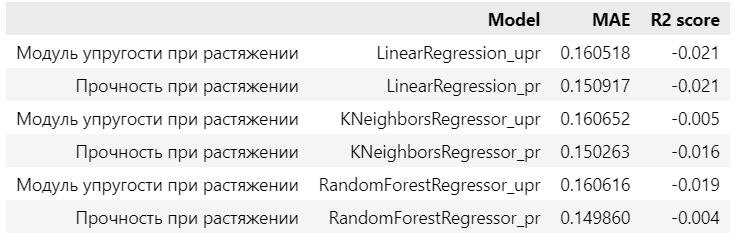


Рисунок 14 – Сравнение ошибок моделей

Произведем поиск гиперпараметров выбранных моделей.

2.2.4 Гиперпараметры для линейной регрессии

lr.fit(X\_train\_upr, y\_train\_upr)

y\_pred\_lr = lr.predict(X\_test\_upr)

plt.figure(figsize=(10, 7))

plt.plot(y\_pred\_lr, "g", label="prediction")

plt.plot(y\_test\_upr.values, label="actual")

plt.grid(True)

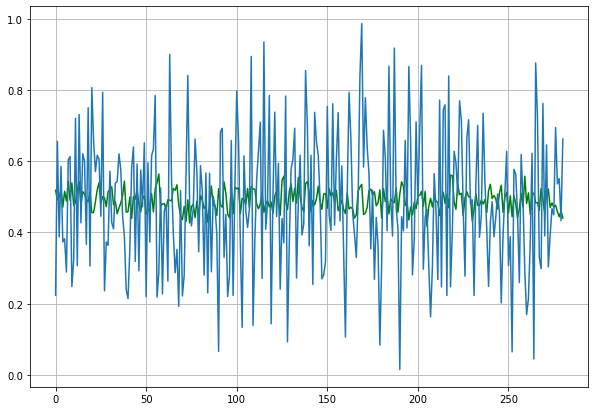


Рисунок 15 – Прогнозное и предсказанные значения линейной регрессии

2.2.4 Гиперпараметры для к-ближайших соседей

knn = KNeighborsRegressor(n\_neighbors=5)

knn.fit(X\_train\_upr, y\_train\_upr)

y\_pred\_knn = knn.predict(X\_test\_upr)

plt.figure(figsize=(10, 7))

plt.plot(y\_pred\_knn, "g", label="prediction")

plt.plot(y\_test\_upr.values, label="actual")

plt.grid(True)

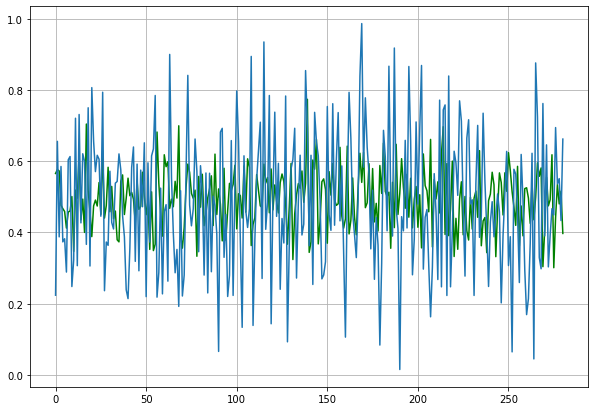


Рисунок 16 – Прогнозное и предсказанные значения к-ближайших соседей

Поиск лучших гиперпараметров

param\_grid = {"n\_neighbors": range(1, 50)}

gs\_upr = GridSearchCV(knn, param\_grid, cv=10, verbose=1, n\_jobs=1)

gs\_upr.fit(X\_train\_upr,y\_train\_upr)

knn = gs\_upr.best\_estimator\_

gs\_upr.best\_params\_

Fitting 10 folds for each of 49 candidates, totalling 490 fits

{'n\_neighbors': 48}

2.2.4 Гиперпараметры для случайного леса

n\_estimators = [5, 7, 9, 11, 15, 20, 40]

max\_depth = [1, 2, 3, 4, 5]

min\_samples\_split = [23, 24, 25, 26]

min\_samples\_leaf = [3, 4, 5, 6]

bootstrap = [True]

param\_grid = {'n\_estimators': n\_estimators,

              'max\_depth': max\_depth,

              'min\_samples\_split': min\_samples\_split,

              'min\_samples\_leaf': min\_samples\_leaf,

              'bootstrap': bootstrap}

gs\_upr = GridSearchCV(RandomForestRegressor(), param\_grid, cv=10, verbose=1, n\_jobs=-1)

gs\_upr.fit(X\_train\_upr, y\_train\_upr)

rfr\_3 = gs\_upr.best\_estimator\_

gs\_upr.best\_params\_

Fitting 10 folds for each of 560 candidates, totalling 5600 fits

c:\anaconda\lib\site-packages\sklearn\model\_selection\\_search.py:926: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel(). self.best\_estimator\_.fit(X, y, \*\*fit\_params)

{'bootstrap': True, 'max\_depth': 4, 'min\_samples\_leaf': 3, 'min\_samples\_split': 24, 'n\_estimators': 20}

Поиск лучших гиперпараметров

rfr = RandomForestRegressor(n\_estimators=100, max\_depth=18)

rfr.fit(X\_train\_upr, y\_train\_upr.values[:, 0])

y\_pred\_upr\_forest = rfr.predict(X\_test\_upr)

plt.figure(figsize=(10, 7))

plt.plot(y\_pred\_upr\_forest, "g", label="prediction")

plt.plot(y\_test\_upr.values, label="actual")

plt.grid(True)

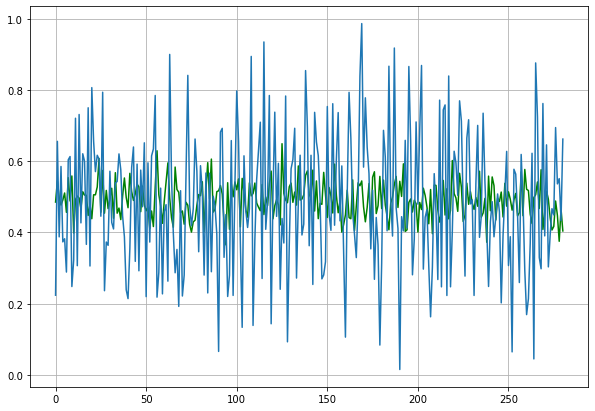


Рисунок 17 – Прогнозное и предсказанные для случайного леса

Наилучшее обучение было у модели случайного леса.

# Нейронная сеть

Построим нейронные сети для упругости при растяжении, плотности при растяжении и параметра матрица-наполнитель.

Создание тренировочной и тестовой выборки:

x\_upr = df\_norm.drop(['Модуль упругости при растяжении, ГПа'], axis=1)

x\_pr = df\_norm.drop(['Прочность при растяжении, МПа'], axis=1)

x\_mn = df\_norm.drop(['Соотношение матрица-наполнитель'], axis=1)

y\_upr = df\_norm[['Модуль упругости при растяжении, ГПа']]

y\_pr = df\_norm[['Прочность при растяжении, МПа']]

y\_mn = df\_norm[['Соотношение матрица-наполнитель']]

X\_train\_upr, X\_test\_upr, y\_train\_upr, y\_test\_upr = train\_test\_split(x\_upr, y\_upr, test\_size=0.3, random\_state=1)

X\_train\_pr, X\_test\_pr, y\_train\_pr, y\_test\_pr = train\_test\_split(x\_pr, y\_pr, test\_size=0.3, random\_state=1)

X\_train\_mn, X\_test\_mn, y\_train\_mn, y\_test\_mn = train\_test\_split(x\_mn, y\_mn, test\_size=0.3, random\_state=1)

Для удобства отображения результатов создадим функции для построения графиков:

а) Потерь модели на тренировочной и тестовой выборках:

def model\_loss\_plot(model\_history):

    plt.figure(figsize=(10, 5))

    plt.plot(model\_history.history['loss'])

    plt.plot(model\_history.history['val\_loss'])

    plt.title('График потерь модели', size=12)

    plt.ylabel('Средняя абсолютная ошибка', size=12)

    plt.xlabel('Эпоха', size=12)

    plt.legend(['loss', 'val\_loss'], loc='best')

    plt.show()

б) Графика оригинального предсказания у

def actual\_and\_predicted\_plot(original\_y, predicted\_y):

    plt.figure(figsize=(10,5))

    plt.title('Тестовые и прогнозные значения', size=12)

    plt.plot(original\_y, color='blue', label = 'Тестовые значения')

    plt.plot(predicted\_y, color='red', label = 'Прогнозные значения')

    plt.legend(loc='best')

    plt.show()

в) Точечного графика оригинального и предсказанного значения у

def actual\_and\_predicted\_scatter(original\_y, predicted\_y):

    plt.figure(figsize=(10,5))

    plt.title('Рассеяние тестовых и прогнозных значений', size=15)

    plt.scatter(original\_y, predicted\_y)

    plt.xlabel('Тестовые значения', size=12)

    plt.ylabel('Прогнозные значения', size=12)

    plt.show()

Нормирование образца

normalizer = tf.keras.layers.Normalization(axis=-1)

X\_train\_mn\_norm = normalizer.adapt(np.array(X\_train\_upr))

Создание нейронной сети для предсказания соотношения матрица-наполнитель:

model\_mn = Sequential(X\_train\_mn\_norm)

model\_mn.add(Dense(128))

model\_mn.add(BatchNormalization())

model\_mn.add(LeakyReLU())

model\_mn.add(Dense(128))

model\_mn.add(BatchNormalization())

model\_mn.add(LeakyReLU())

model\_mn.add(Dense(64))

model\_mn.add(BatchNormalization())

model\_mn.add(LeakyReLU())

model\_mn.add(Dense(64))

model\_mn.add(BatchNormalization())

model\_mn.add(LeakyReLU())

model\_mn.add(Dense(16))

model\_mn.add(BatchNormalization())

model\_mn.add(LeakyReLU())

model\_mn.add(Dense(1))

model\_mn.add(Activation('selu'))

Компилируем модель

model\_mn.compile(

    optimizer=tf.optimizers.SGD(learning\_rate=0.02, momentum=0.5),

    loss='mean\_absolute\_error')

Вводим переменную для остановки обучения:

early\_mn = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=10, verbose=1, mode='auto')

Проводим обучение нейронной сети

%%time

history\_mn = model\_mn.fit(

    X\_train\_mn,

    y\_train\_mn,

    batch\_size = 64,

    epochs=100,

    verbose=1,

    validation\_split = 0.2,

    callbacks = [early\_mn]

    )

Вывод результатов эпох обучения (рисунок 18) показывает, что нейронная сеть перестала обучаться на 41 эпохе.

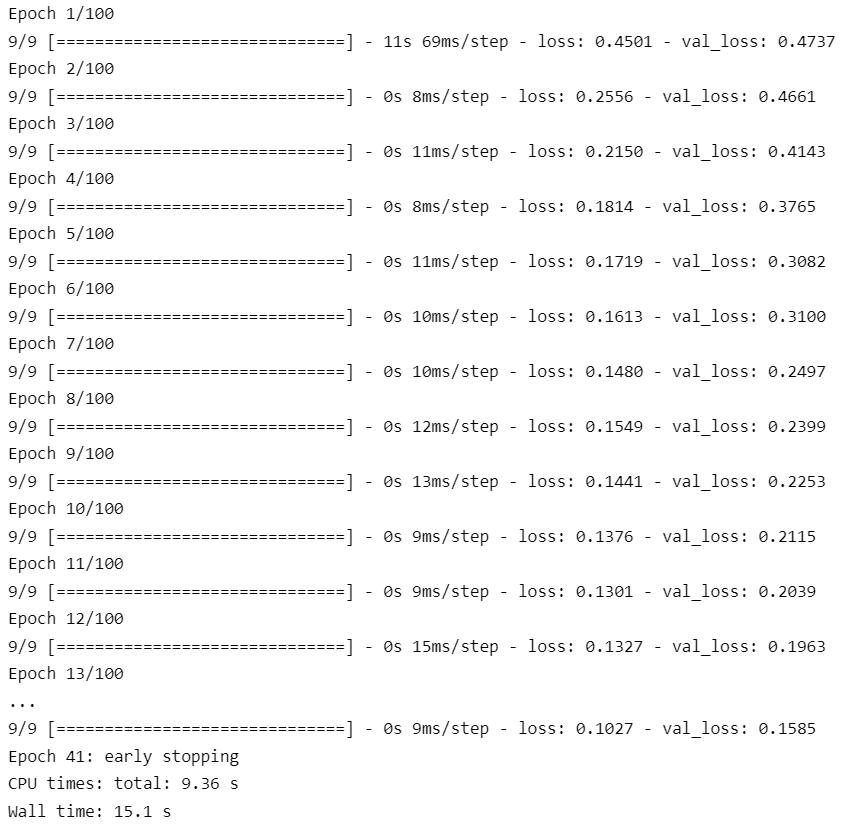


Рисунок 18 – Результаты обучения нейронной сети

Параметры нейронной сети представлены на рисунке 19. Из него можно определить, что всего параметров 33 249, из них 800 – «не обучены», т.е. в процессе обучения их веса не меняются.

model\_mn.summary()

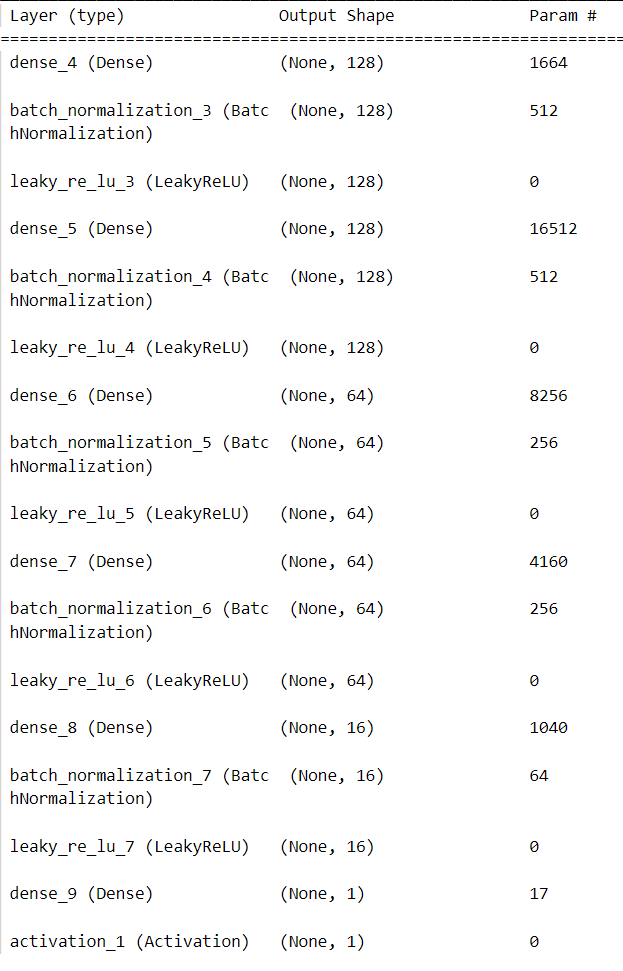




Рисунок 19 – Параметры нейронной сети

Из графика потерь модели (рисунок 20) можно проследить тенденцию уменьшения средней абсолютной ошибки от эпохи к эпохе, и её равномерный (относительно) участок к концу обучения. Нахождение графика данных перекрестной проверки, над графиком обучающих данных, обусловлено использованием необученных нейронов.

model\_loss\_plot(history\_mn)

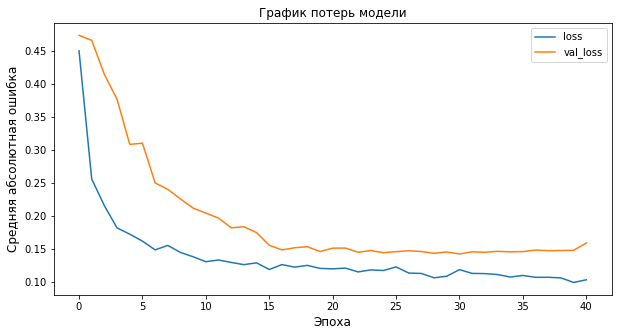


Рисунок 20 – График потерь модели

На рисунке 21 отображены тестовые и прогнозные значения.

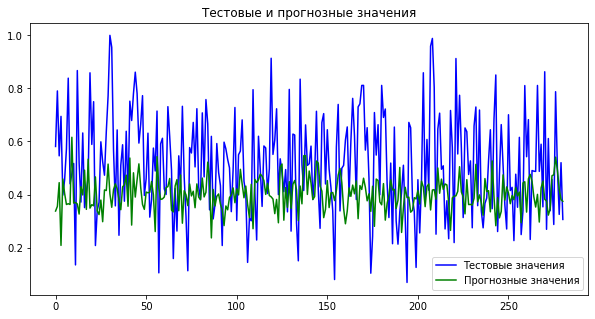


Рисунок 21 – Тестовые и прогнозные значения

Рассеяние тестовых и прогнозных значений указывает на слабую корреляцию тестовых и прогнозируемых значений, что свидетельствует о необходимости доработки нейронной сети, или изменении подхода в решении данной задачи.

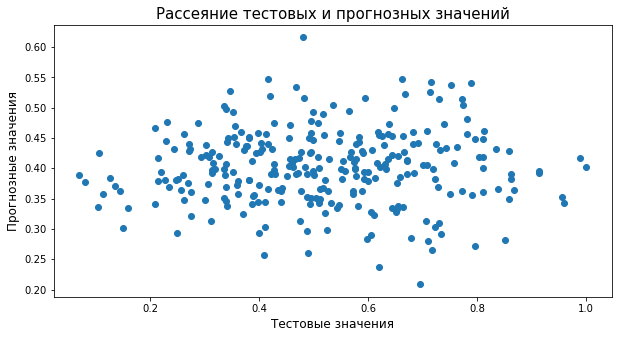


Рисунок 22 – Рассеяние тестовых и прогнозных значений

# Разработка приложения

К сожалению, цель в виде разработки приложения достигнута не была.

# Создание удаленного репозитория и загрузка результатов работы

Удаленный репозиторий находится на GitHub по ссылке:

<https://github.com/KhEvgeniy/VKR>

В нем находится файл Readme.md, а также все актуальные файлы к данной работе. Скринот страницы представлен на рисунке 23.

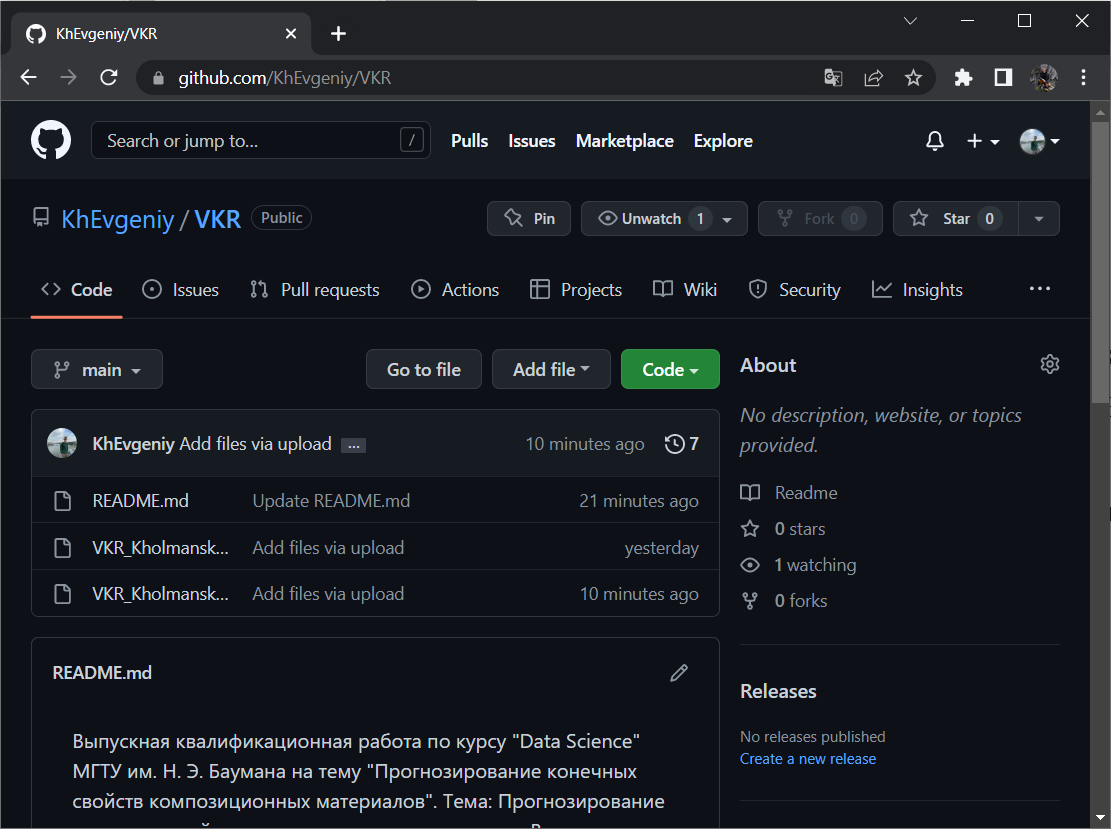


Рисунок 23 – Страница репозитория на GitHub

Заключение

В данной работе был проведен анализ и предобработка данных. Построенные модели показали плохой результат в условиях данного задания. В качестве вариантов решения, предлагаю:

1. повторно провести анализ данных на выявление зависимостей и отклонений;
2. выбрать иные методы и способы решения.

Библиографический список

1. Межгосударственный стандарт ГОСТ 31938-2011 «Арматура композитная полимерная для армирования бетонных конструкций. Общие технические условия» 2011. – с. 6-7.
2. Полимерные композиционные материалы на основе волокон различной химической природы / Ю.А. Кадыкова, А.Н. Леонтьев, О.Г. Васильева, С.Е. Артеменко // Строительные материалы, оборудование, технологии ХХI века. 2002. № 6. С. 10-11.
3. Сравнительные характеристики базальто-, стекло- и углепластиков, сформованных методом поликонденсационного наполнения / Ю.А. Кадыкова, О.Г. Васильева, С.Е. Артеменко, А.Н. Леонтьев // Пластические массы. 2003. № 5. С. 37-38.
4. А.М. Миронов «Машинное обучение часть 1» МГУ – с. 60-65.
5. Джоэл Грас «Data Science Наука о данных с нуля» – 2-е изд. – СПб. 2021. –с.70-80.
6. Box G.E.P., Jenkins G.M., Reinsel G.C., Ljung G.M. Time Series Analysis: Forecasting and Control - 5th Edition. — Wiley, 2015.
7. Nisbet R., Elder J., Miner G. Handbook of Statistical Analysis and Data Mining Applications. – Academic Press, 2009.
8. В. Ш. Берикашвили, С. П.Оськин Статистическая обработка данных, планирование эксперимента и случайные процессы: учебное пособие для вузов – 2-е изд., испр. и доп. – М. Юрайт, 2021. – 163 с.
9. Джулли, Пал: Библиотека Keras – инструмент глубокого обучения / пер. с англ. А. А. Слинкин.– ДМК Пресс, 2017. – 249 с.
10. В.А.Головко. От многослойных персептронов к нейронным сетям глубокого доверия: парадигмы обучения и применение. В сб.: Нейроинформатика-2015. XVII Всероссийская научно-техническая конференция с международным участием. Лекции по нейроинформатике, НИЯУ МИФИ, 2015 –. с.47-84.
11. Перевод: А.Мюллер, С.Гвидо. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. Вильямс, 2017, 480 c. ISBN 978-5-99089-108-1.
12. Перевод: О.Жерон. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем. Вильямс, 2018, 688 c. ISBN 978-5-950-02962-2.
13. Начальное обучение. Машинное обучение в Python [Электронный ресурс] https://scikit-learn.org/stable/index.html (дата обращения 17.06.2022).
14. Stack overflow [Электронный ресурс] https://stackoverflow.com (дата обращения 17.06.2022).