ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

DATA SCIENSE

Слушатель Грудинин Алексей Геннадьевич
МГТУ им.Баумана
2022г.

Начало работы

- ✓ Подробный план работы:
- • Составил подробный план, который помог
- плодотворно двигаться к цели;
- • Изучил теоретические основы, методы
- решения и практические составляющие
- поставленной задачи;
- • Некоторые пункты плана повторял много
- раз, чтобы добиться результата;
- • Использовал 9 разных методов регрессий
- для каждой из моделей;
- • Приложение успешно работает
- ✓ Графики:
- • Строил много графиков;
- • Несколько подобных графиков для одних
- и тех же переменных;
- • Старался делать все графики в одном стиле

Описание используемых методов

– метод опорных векторов (Support Vector Regression)
– случайный лес (Random Forest Regressor)
– линейная регрессия (Linear Regression)
– градиентный бустинг (Gradient Boosting Regressor)
– K-ближайших соседей (K Neighbors Regressor)
– дерево решений (Decision Tree Regressor)
– стохастический градиентный спуск (Stochastic Gradient Descent Regressor)
– многослойный перцептрон (Multi-layer Perceptron regressor)
– Лассо (the Lasso)

ОБЪЕДИНЕНИЕ ФАЙЛОВ И РАЗВЕДОЧНЫЙ АНАЛИЗ

- ✓ Объединение по индексу:
- • Импортируем необходимые библиотеки;
- • Загружаем файлы;
- • Посмотрим размерность;
- • Объединим оба файла по индексу по типу
- объединения INNER
- ✓ Разведочный анализ данных:
- • Посмотрим на начальные и конечные
- строки нашего датасета;
- • Изучим информацию о датасете;
- • Проверим типы данных в каждом
- столбце;
- • Проверим пропуски;
- • Поищем уникальные значения с помощью
- функции nunique

Объединим по индексу, тип объединения INNER, смотрим итоговый датасет

```
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   int64  
 11  Шаг нашивки                           1023 non-null   float64
 12  Плотность нашивки                     1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

ОПИСАТЕЛЬНАЯ СТАТИСТИКА

```
df.describe()
```

	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	У наши
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	482.731833	73.328571	2466.922843	218.423144	0.491
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3.118983	485.628006	59.735931	0.500
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64.054061	1036.856605	33.803026	0.000
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71.245018	2135.850448	179.627520	0.000
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73.268805	2459.524526	219.198882	0.000
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75.356612	2767.193119	257.481724	1.000
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82.682051	3848.436732	414.590628	1.000

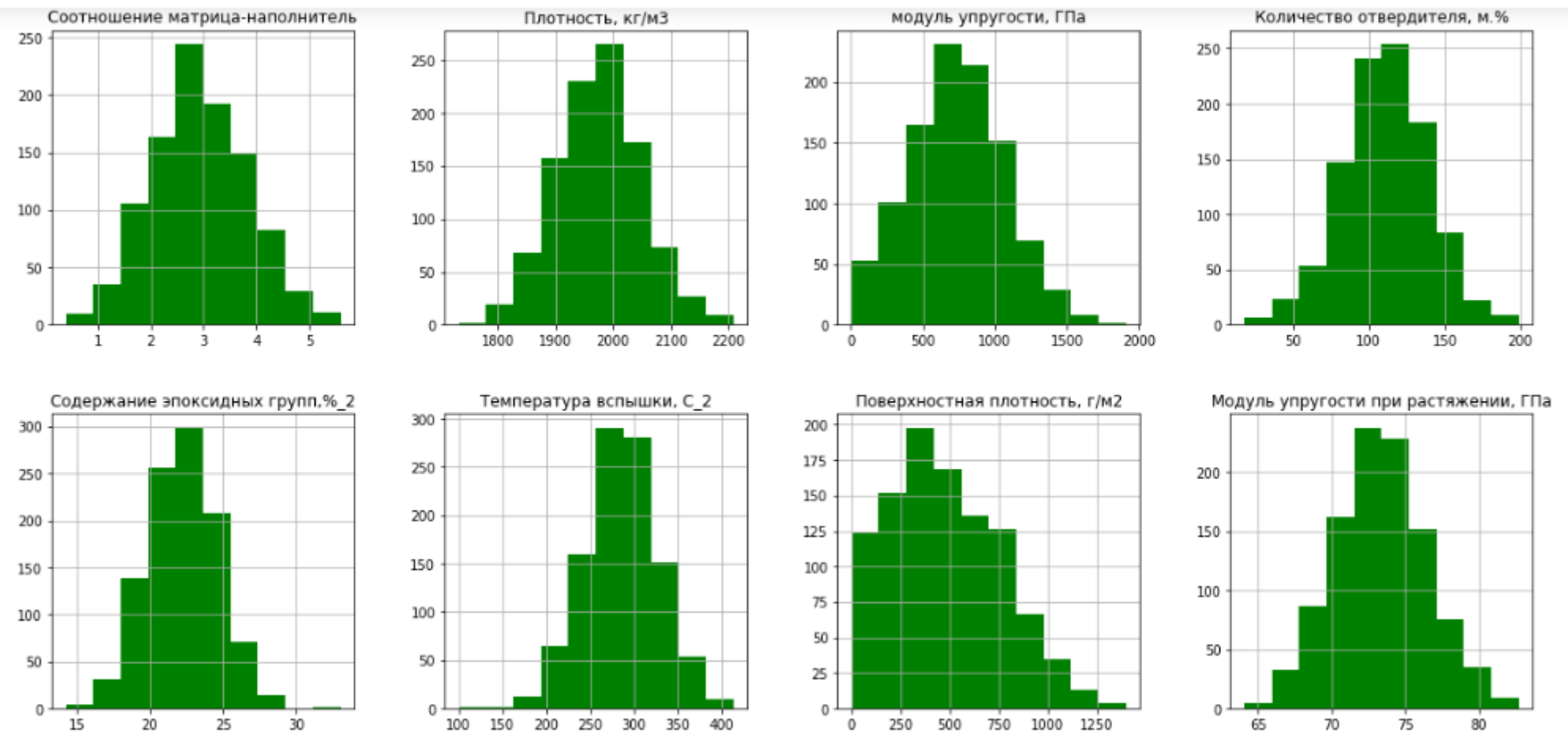
✓ Описательная статистика:

• Изучим описательную статистику данных (максимальное, минимальное, квартили, медиана, стандартное отклонение, среднее значение и т.д.),

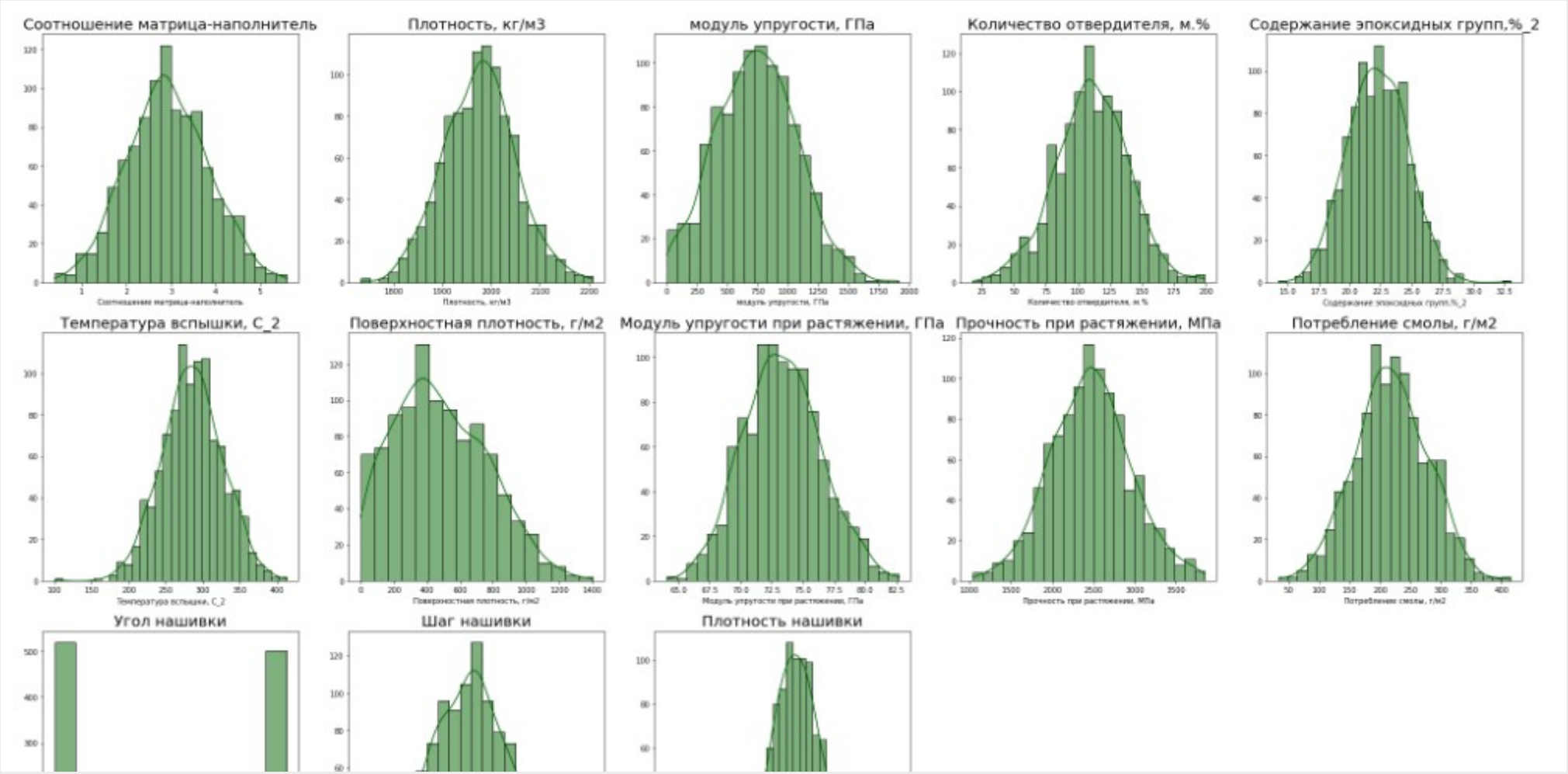
• Посмотрим на основные параметры анализа данных;

• Проверим датасет на пропущенные и дублирующие данные;• Вычислим коэффициенты ранговой корреляции Кендалла и Пирсона

Визуализация «сырых» данных

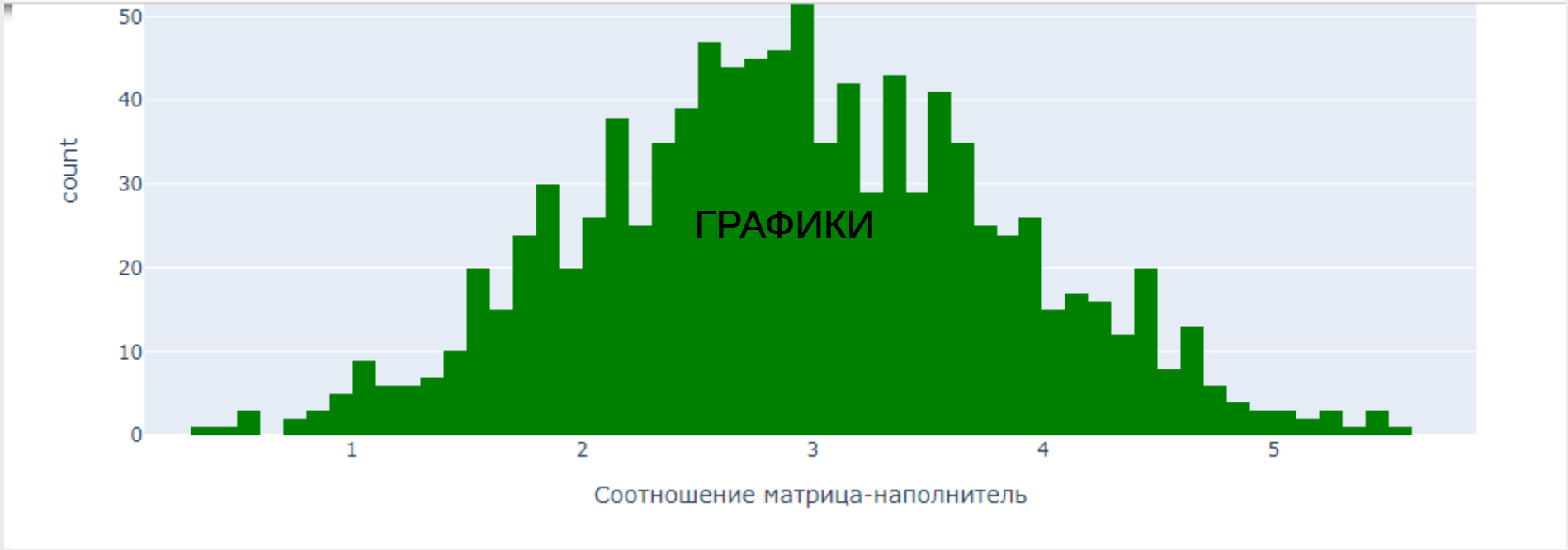


ГРАФИКИ



ГРАФИКИ

```
for column in df.columns:
    fig = px.histogram(df, x = column, color_discrete_sequence = ['green'], nbins = 100, marginal = "box")
    fig.show()
```



Графики без нормализации и исключения шумов :

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана

ПРЕДОБРАБОТКА ДАННЫХ

✓Исключение выбросов:

• Посчитаем количество значений методом 3 сигм и методом межквартильных расстояний;

• Исключим выбросы методом межквартильного расстояния ;

• Проверим результат;

• Построим графики;

• Убедимся, что выбросы ещё остались;

• Повторим удаление выбросов ещё 4 раза до полного удаления;

• Проверим чистоту датасета от выбросов

• Построим все возможные графики «чистого» датасета

Для удаления выбросов существует 2 основных метода - метод 3-х сигм и межквартил

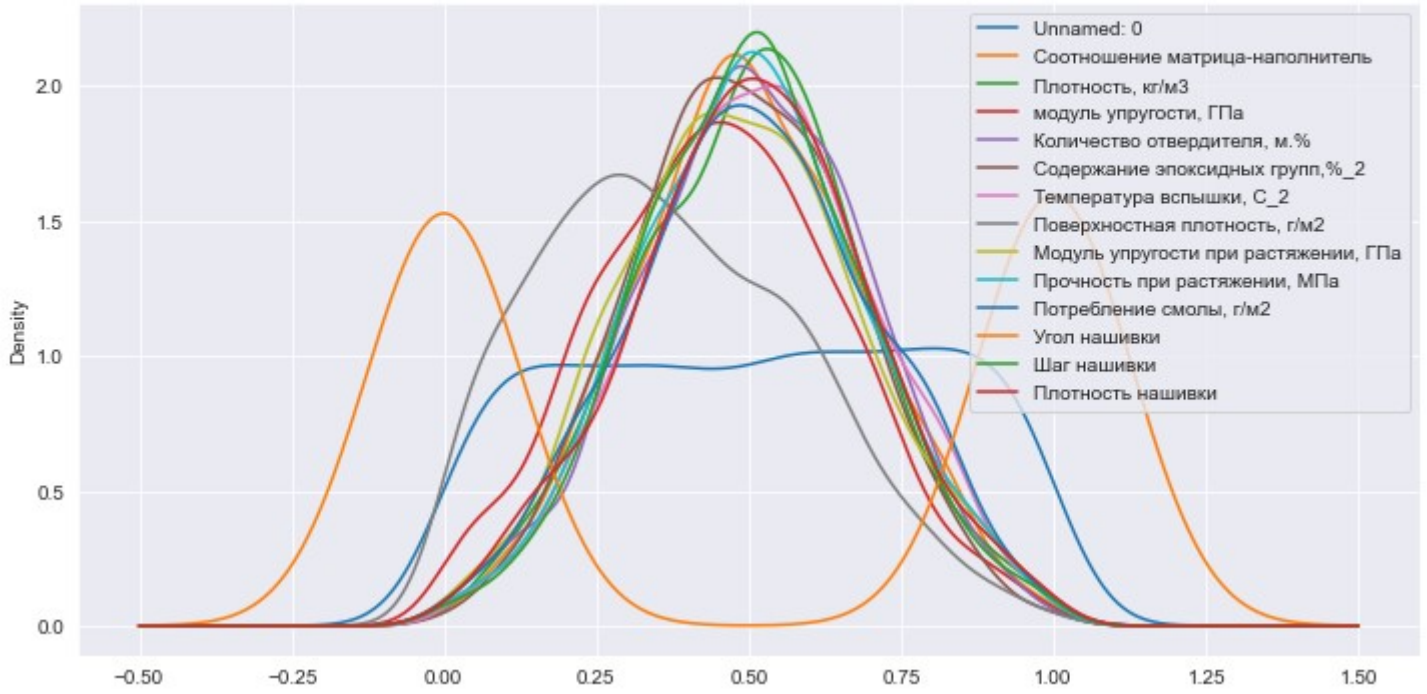
```
metod_3s = 0
metod_iq = 0
count_iq = [] # Список, куда записывается количество выбросов по каждой к
count_3s = [] # Список, куда записывается количество выбросов по каждой к
for column in df:
    d = df.loc[:, [column]]
# методом 3-х сигм
    zscore = (df[column] - df[column].mean()) / df[column].std()
    d['3s'] = zscore.abs() > 3
    metod_3s += d['3s'].sum()
    count_3s.append(d['3s'].sum())
print(column,'3s', ': ', d['3s'].sum())
# методом межквартильных расстояний
q1 = np.quantile(df[column], 0.25)
q3 = np.quantile(df[column], 0.75)
iqr = q3 - q1
lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr
d['iq'] = (df[column] <= lower) | (df[column] >= upper)
metod_iq += d['iq'].sum()
count_iq.append(d['iq'].sum())
print(column, ': ', d['iq'].sum())
print('Метод 3-х сигм, выбросов:', metod_3s)
print('Метод межквартильных расстояний, выбросов:', metod_iq)
```

ПРЕДОБРАБОТКА ДАННЫХ

- ✓ Нормализация данных:
- • Нормализуем данные MinMaxScaler();
- • Построим график плотности ядра;
- • Проверим результат MinMaxScaler();
- • Построим графики MinMaxScaler();
- • Нормализуем данные с помощью
- Normalizer();
- • Проверим результат Normalizer();
- • Построим графики Normalizer();
  ✓ Стандартизация данных:
- • Стандартизируем данные с
- помощью StandardScaler();
- • Проверим результат
- StandardScaler();
- • Построим графики
- StandardScaler();

```
fig, ax = plt.subplots(figsize = (12, 6))
df_minmax_n.plot(kind = 'kde', ax = ax)
```

```
<AxesSubplot:ylabel='Density'>
```

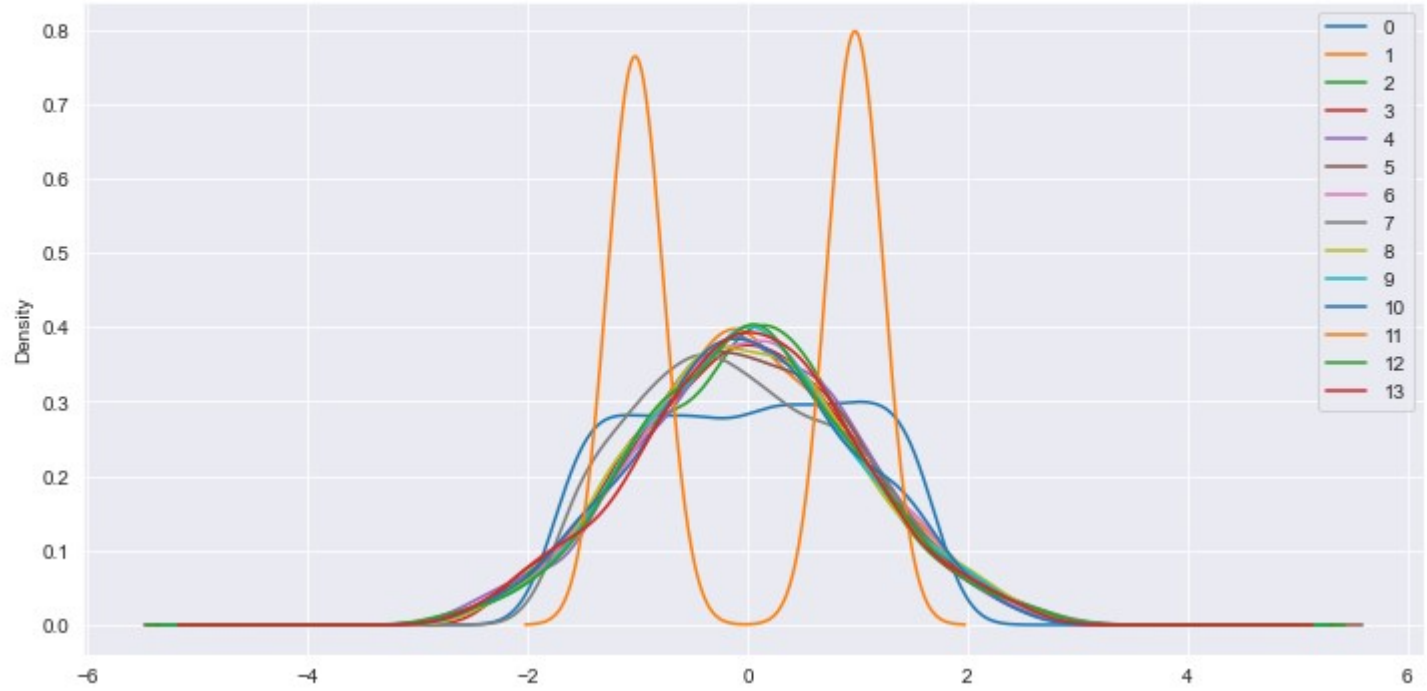


ПРЕДОБРАБОТКА ДАННЫХ

- Визуализация

```
fig, ax = plt.subplots(figsize = (12, 6))
df_standart_1.plot(kind = 'kde', ax = ax)
```

```
<AxesSubplot:ylabel='Density'>
```

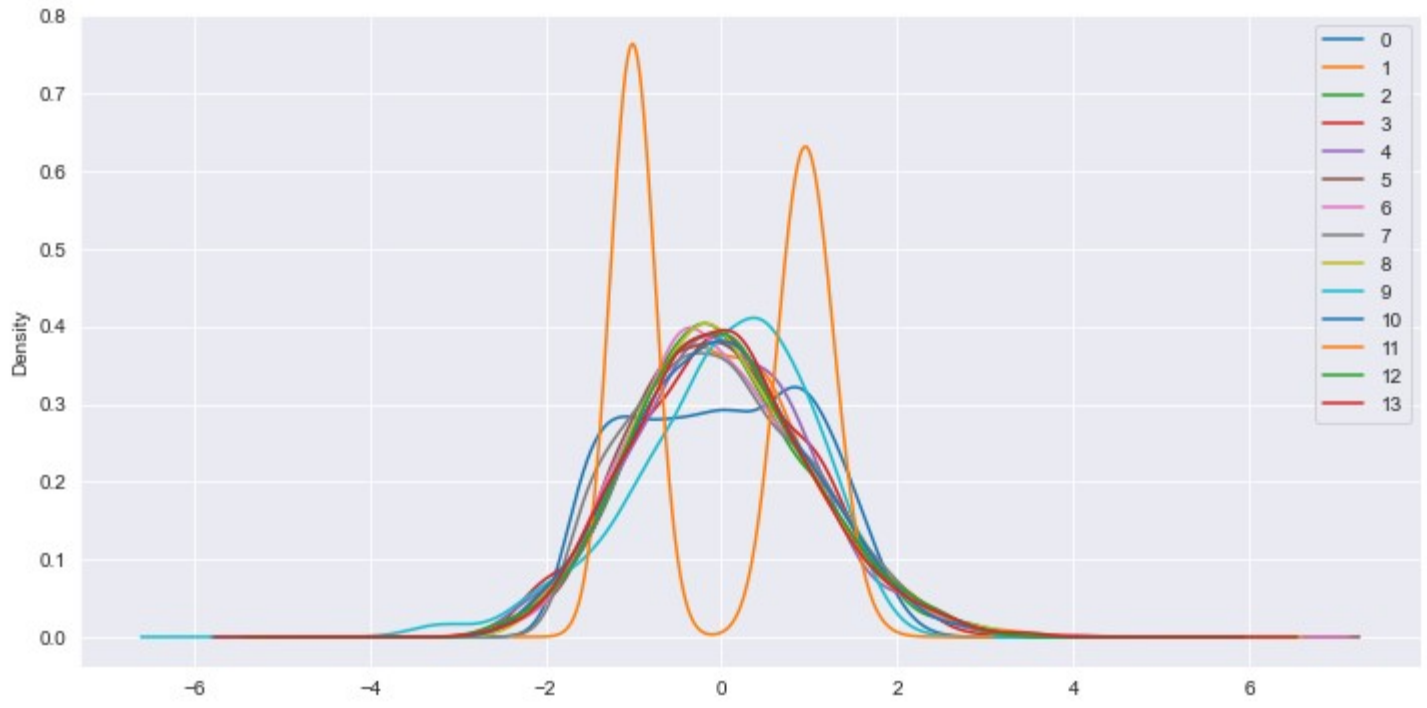


ПРЕДОБРАБОТКА ДАННЫХ

- Визуализация

```
fig, ax = plt.subplots(figsize = (12, 6))
df_standart_2.plot(kind = 'kde', ax=ax)
```

```
<AxesSubplot:ylabel='Density'>
```

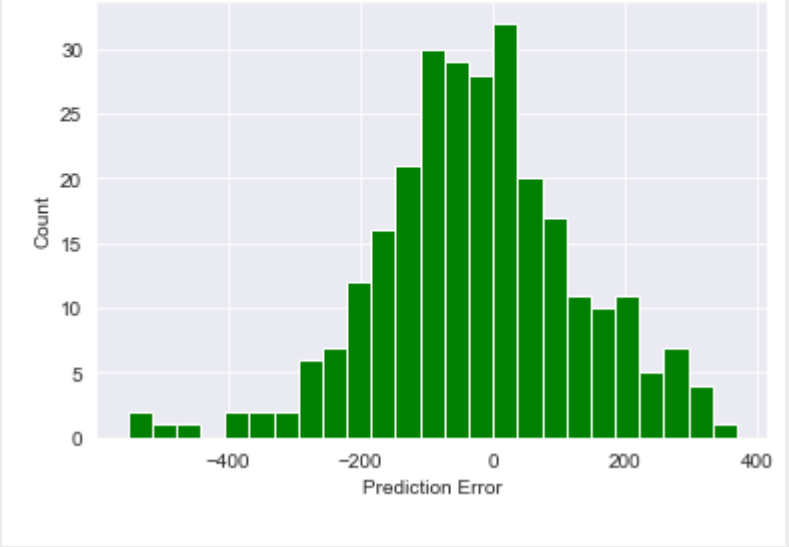


Разработка и обучение моделей для прогноза прочности при растяжении:

- ✓ Методы построения модели
- • Разбиваем данные на тестовую и
- тренировочную выборки;
- • Обучаем модель;
- • Вычисляем коэффициент детерминации;
- • Считаем MAE, MAPE, MSE, RMSE, test
- score train и test score test;
- • Сравниваем с результатами модели,
- выдающей среднее значение;
- • Построим графики для тестовых и
- прогнозных значений;
- • Построим гистограмму распределения
- ошибки

Визуализация гистограммы распределения ошибки

```
error = y_test_1 - y_pred_knn
plt.hist(error, bins = 25, color = "g")
plt.xlabel('Prediction Error')
_ = plt.ylabel('Count')
```



Нейронная сеть для соотношения «матрица-наполнитель»

- • Сформируем входы и выход для модели.
- • Разобьём выборки на обучающую и
- тестовую.
- • Нормализуем данные.
- • Создадим функцию для поиска
- наилучших параметров и слоёв.
- • Построим модель, определим параметры,
- найдем оптимальные параметры
- посмотрим на результаты;
- • Повторим все эти этапы до построения
- окончательной модели;
- • Обучим нейросеть;
- • Посмотрим на потери модели;
- • Построим график потерь на
- тренировочной и тестовой выборках.
- • Построим график результата работы
- модели.

построение окончательной модели

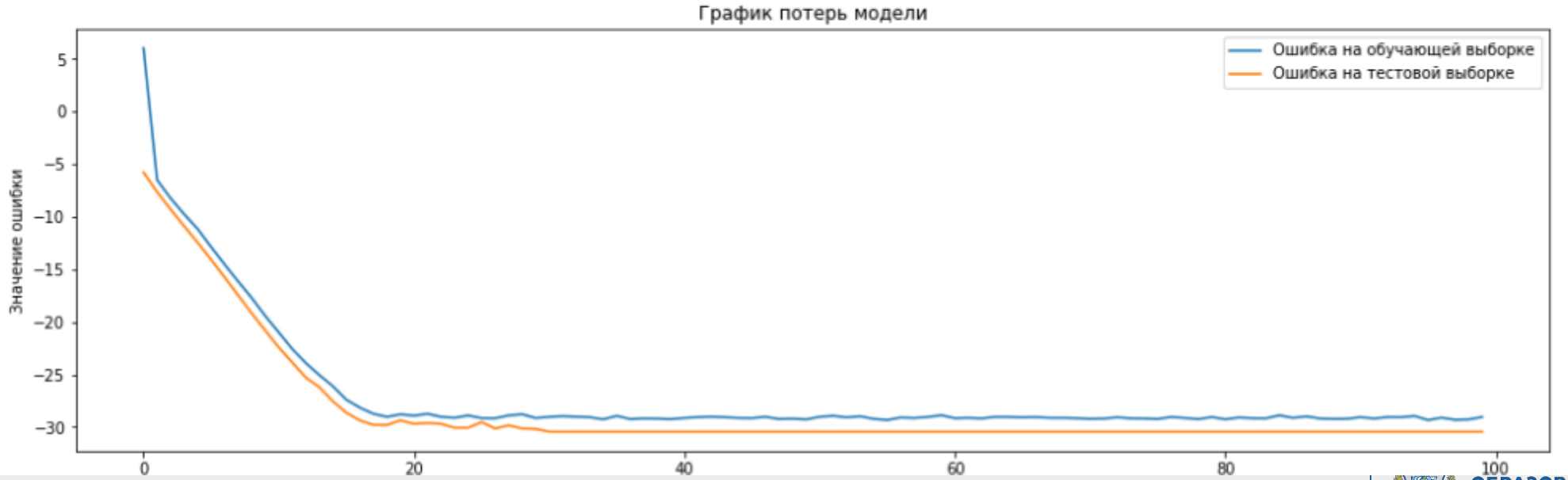
```
model = create_model(lyrs=[128, 64, 16, 3], dr=0.05)
print(model.summary())
```

```
Model: "sequential_195"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_490 (Dense)           (None, 128)               1792

 dense_491 (Dense)           (None, 64)                8256

 dense_492 (Dense)           (None, 16)                1040

 dense_493 (Dense)           (None, 3)                 51

 dropout_195 (Dropout)       (None, 3)                 0

 dense_494 (Dense)           (None, 3)                 12

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

Нейронная сеть для соотношения «матрица-наполнитель»:

Посмотрим на график потерь на тренировочной и тестовой выборках

```
def model_loss_plot(model_hist):
    plt.figure(figsize = (17,5))
    plt.plot(model_hist.history['loss'],
    label = 'ошибка на обучающей выборке')
    plt.plot(model_hist.history['val_loss'],
    label = 'ошибка на тестовой выборке')
    plt.title('График потерь модели')
    plt.ylabel('Значение ошибки')
    plt.xlabel('Эпохи')
    plt.legend(['Ошибка на обучающей выборке', 'Ошибка на тестовой выборке'], loc='best')
    plt.show()
model_loss_plot(model_hist)
```



ПРИЛОЖЕНИЕ

Приложение:

✓ Пользовательское приложение

• Сохранил вторую модель нейронной сети для разработки веб-приложения для прогнозирования соотношения "матрица-наполнитель" в фреймворке Flask;

• 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 «Готово».

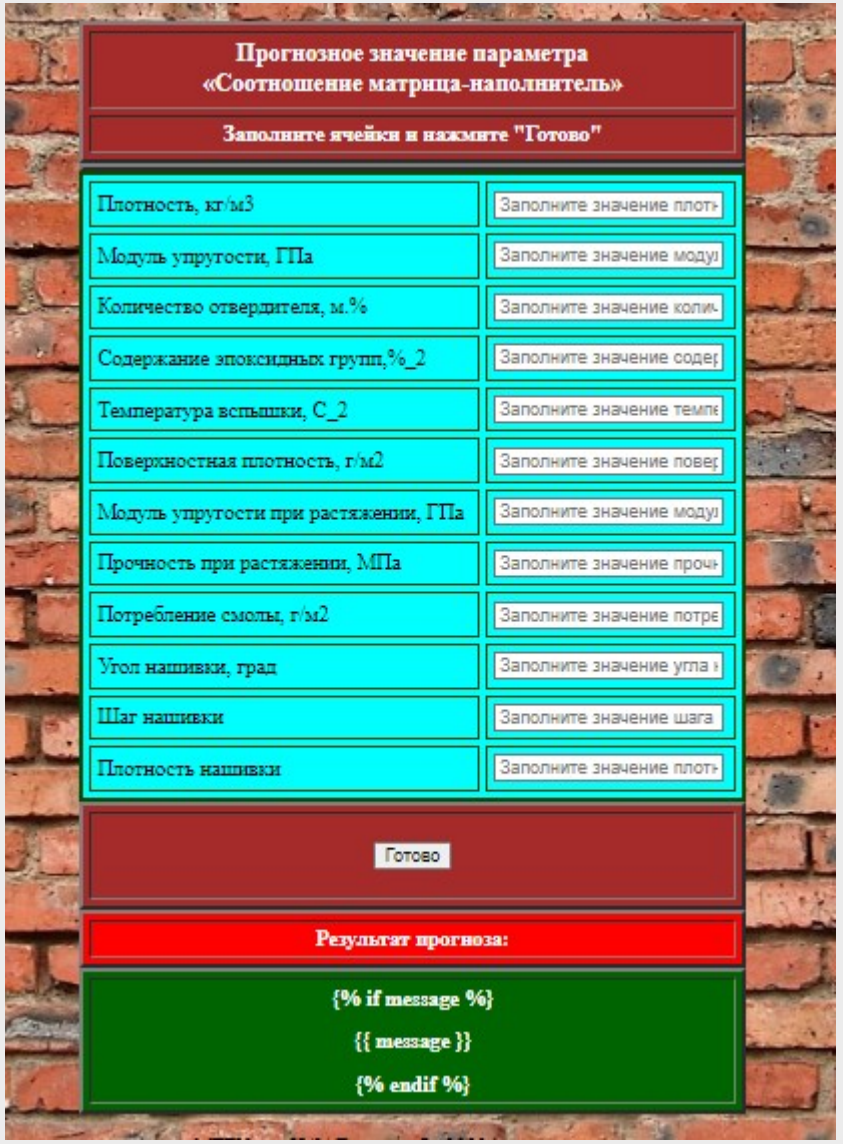
• На выходе пользователь получает результат прогноза для значения парамет-ра «Соотношение «матрица – наполнитель»».

• Приложение успешно работает

✓ Репозиторий на github.com

https://github.com/AlekseyGrudinin/VKR

✓https://colab.research.google.com/drive/1yO8t0m3A8y8rbnKqb0-bpcyidIYmWn61#scrollTo=Fg92VnQIOjXn



ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана

Спасибо за внимание

edu.bmstu.ru

+7 495 182-83-85

edu@bmstu.ru

Москва, Госпитальный переулок , д. 4-6, с.3

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана