Выпускная квалификационная работа Шершневой Елены

слушателя курса "Data Science" Образовательного центра Московского государственного технического университета им. Н.Э. Баумана

Тема исследования: Прогнозирование конечных свойств новых материалов (композиционных материалов). Цель исследования: построение моделей прогнозирования следующих параметров: «модуль упругости при растяжении»; «прочность при растяжении»; «соотношение матрица-наполнитель» Итог работы: Разработка приложения с графическим интерфейсом, которое будет выдавать прогноз параметра «соотношение матрица-наполнитель».

Приложение 1 Подробный план работы: 1. Загружаем и обрабатываем входящие датасеты 1.1. Удаляем неинформативные столбцы 1.2. Объединяем датасеты по методу INNER 2. Проводим разведочный анализ данных: 2.1. Данные в столбце "Угол нашивки» приведём к 0 и 1 2.2. Изучим описательную статистику каждой переменной - среднее, медиана, стандартное отклонение, минимум, максимум, квартили 2.3. Проверим датасет на пропуски и дубликаты данных 2.4. Получим среднее, медианное значение для каждой колонки (по заданию необходимо получить их отдельно, поэтому продублируем их только отдельно) 2.5. Вычислим коэффициенты ранговой корреляции Кендалла 2.6. Вычислим коэффициенты корреляции Пирсона 3. Визуализируем наш разведочный анализ сырых данных (до выбросов и нормализации) 3.1. Построим несколько вариантов гистограмм распределения каждой переменной 3.2. Построим несколько вариантов диаграмм ящиков с усами каждой переменной 3.3. Построим гистограмму распределения и диаграмма "ящик с усами" одновременно вместе с данными по каждому столбцу 3.4. Построим несколько вариантов попарных графиков рассеяния точек (матрицы диаграмм рассеяния) 3.5. Построим графики квантиль-квантиль 3.6. Построим корреляционную матрицу с помощью тепловой карты 4. Проведём предобработку данных (в данном пункте только очистка датасета от выбросов) 4.1. Проверим выбросы по 2 методам: 3-х сигм или межквартильных расстояний 4.2. Посчитаем распределение выбросов по каждому столбцу (с целью предотвращения удаления особенностей признака или допущения ошибки) 4.3. Исключим выбросы методом межквартильного расстояния 4.4. Удалим строки с выбросами 4.5. Визуализируем датасет без выбросов, и убедимся, что выбросы еще есть. 4.6. Для полной очистки датасета от выбросов повторим пункты (4.3 – 4.5) ещё 3 раза. 4.7. Сохраняем идеальный, без выбросов датасет 4.8. Изучим чистые данные по всем параметрам 4.9. Визуализируем «чистый» датасет (без выбросов) 5. Проведём нормализацию и стандартизацию (продолжим предобработку данных) 5.1. Визуализируем плотность ядра 5.2. Нормализуем данные с помощью MinMaxScaler() 5.3. Нормализуем данные с помощью Normalizer() 5.4. Сравним с данными до нормализации 5.5. Проверим перевод данных из нормализованных в исходные 5.6. Рассмотрим несколько вариантов корреляции между параметрами после нормализации 5.7. Стандартизируем данные 5.8. Визуализируем данные корреляции 5.9. Посмотрим на описательную статистику после нормализации и после стандартизации 6. Разработаем и обучим нескольких моделей прогноза прочности при растяжении (с 30% тестовой выборки) 6.1. Определим входы и выходы для моделей 6.2. Разобьём данные на обучающую и тестовую выборки 6.3. Проверим правильность разбивки 6.4. Построим модели и найдём лучшие гиперпараметры (задача по заданию): 6.5. Построим и визуализируем результат работы метода опорных векторов 6.6. Построим и визуализируем результат работы метода случайного леса 6.7. Построим и визуализируем результат работы линейной регрессии 6.8. Построим и визуализируем результат работы метода градиентного бустинга 6.9. Построим и визуализируем результат работы метода К ближайших соседей 6.10. Построим и визуализируем результат работы метода деревья решений 6.11. Построим и визуализируем результат работы стохастического градиентного спуска 6.12. Построим и визуализируем результат работы многослойного перцептрона 6.13. Построим и визуализируем результат работы лассо регрессии 6.14.

Сравним наши модели по метрике MAE 6.15. Найдём лучшие гиперпараметры для случайного леса 6.16. Подставим значения в нашу модель случайного леса 6.17. Найдём лучшие гиперпараметры для K ближайших соседей 6.18. Подставим значения в нашу модель K ближайших соседей 6.19. Найдём лучшие гиперпараметры метода деревья решений 6.20. Подставим значения в нашу модель метода деревья решений 6.21. Проверим все модели и процессинги и выведем лучшую модель и процессинг 7. Разработаем и обучим нескольких моделей прогноза модуля упругости при растяжении (с 30% тестовой выборки) 7.1. Определим входы и выходы для моделей 7.2. Разобьём данные на обучающую и тестовую выборки 7.3. Проверим правильность разбивки 7.4. Построим модели и найдём лучшие гиперпараметры (задача по заданию): 7.5. Построим и визуализируем результат работы метода опорных векторов 7.6. Построим и визуализируем результат работы метода случайного леса 7.7. Построим и визуализируем результат работы линейной регрессии 7.8. Построим и визуализируем результат работы метода градиентного бустинга 7.9. Построим и визуализируем результат работы метода K ближайших соседей 7.10. Построим и визуализируем результат работы метода деревья решений 7.11. Построим и визуализируем результат работы стохастического градиентного спуска 7.12. Построим и визуализируем результат работы многослойного перцептрона 7.13. Построим и визуализируем результат работы лассо регрессии 7.14. Сравним наши модели по метрике MAE 7.15. Найдём лучшие гиперпараметры для случайного леса 7.16. Подставим значения в нашу модель случайного леса 7.17. Найдём лучшие гиперпараметры для K ближайших соседей 7.18. Подставим значения в нашу модель K ближайших соседей 7.19. Найдём лучшие гиперпараметры метода деревья решений 7.20. Подставим значения в нашу модель метода деревья решений 7.21. Проверим все модели и процессинги и выведем лучшую модель и процессинг 8. Нейронная сеть для рекомендации соотношения матрица-наполнитель 8.1. Сформируем входы и выход для модели 8.2. Нормализуем данные 8.3. Построим модель, определим параметры 8.4. Найдем оптимальные параметры для модели 8.5. Посмотрим на результаты 8.6. Повторим шаги 8.4 – 8.5 до построения окончательной модели 8.7. Обучим нейросеть 80/20 8.8. Оценим модель 8.9. Посмотрим на потери модели 8.10. Посмотрим на график результата работы модели 8.11. Посмотрим на график потерь на тренировочной и тестовой выборках 8.12. Сконфигурируем другую модель, зададим слои 8.13. Посмотрим на архитектуру другой модели 8.14. Обучим другую модель 8.15. Посмотрим на потери другой модели 8.16. Посмотрим на график потерь на тренировочной и тестовой выборках 8.17. Зададим функцию для визуализации факт/прогноз для результатов моделей 8.18. Посмотрим на график результата работы модели 8.19. Оценим модель MSE

Импортируем сразу все необходимые библиотеки для нашего исследования

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns
import plotly.express as px
import tensorflow as tf
import sklearn

from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression, SGDRegres
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percenta
from sklearn.model_selection import train_test_split, GridSearchCV, KFold, cross
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn import preprocessing
from sklearn.preprocessing import Normalizer, LabelEncoder, MinMaxScaler, Standa
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from tensorflow import keras as keras
from tensorflow.keras import layers
```

```
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization,
from pandas import read_excel, DataFrame, Series
from keras.wrappers.scikit_learn import KerasClassifier, KerasRegressor
from tensorflow.keras.models import Sequential
from numpy.random import seed
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
```

Загружаем исходные данные из обеих excel таблиц и удаляем колонку с индексом

In [2]:
```
#Загружаем первый датасет (базальтопластик) и посмотрим на названия столбцов
df_bp = pd.read_excel(r"C:\Users\Avona\Desktop\Моя ВКР\Datasets\X_bp.xlsx")
df_bp.shape
```

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

In [3]:
```
#Удаляем первый неинформативный столбец
df_bp.drop(['Unnamed: 0'], axis=1, inplace=True)
#Посмотрим на первые 5 строк первого датасета и убедимся, что первый столбец уда
df_bp.head()
```

Out[3]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверх плотн
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	

In [4]:
```
# Проверим размерность первого файла
df_bp.shape
```

Out[4]:
```
(1023, 10)
```

In [5]:
```
# Загружаем второй датасет (углепластик)
df_nup = pd.read_excel(r"C:\Users\Avona\Desktop\Моя ВКР\Datasets\X_nup.xlsx")
df_nup.shape
```

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

In [6]:
```
#Удаляем первый неинформативный столбец
df_nup.drop(['Unnamed: 0'], axis=1, inplace=True)
#Посмотрим на первые 5 строк второго датасета и убедимся, что и здесь не нужный
df_nup.head()
```

Out[6]:

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0.0	4.0	57.0
1	0.0	4.0	60.0
2	0.0	4.0	70.0
3	0.0	5.0	47.0
4	0.0	5.0	57.0

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

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

Объединим по индексу, тип объединения INNER, смотрим итоговый датасет

In [8]:
```
# Понимаем, что эти два датасета имеют разный объем строк.
# Но наша задача собрать исходные данные файлы в один, единый набор данных.
# По условию задачи объединяем их по типу INNER.
df = df_bp.merge(df_nup, left_index = True, right_index = True, how = 'inner')
df.head().T
```

Out[8]:

	0	1	2	3	4
Соотношение матрица-наполнитель	1.857143	1.857143	1.857143	1.857143	2.771331
Плотность, кг/м3	2030.000000	2030.000000	2030.000000	2030.000000	2030.000000
модуль упругости, ГПа	738.736842	738.736842	738.736842	738.736842	753.000000
Количество отвердителя, м.%	30.000000	50.000000	49.900000	129.000000	111.860000
Содержание эпоксидных групп,%_2	22.267857	23.750000	33.000000	21.250000	22.267857
Температура вспышки, С_2	100.000000	284.615385	284.615385	300.000000	284.615385
Поверхностная плотность, г/м2	210.000000	210.000000	210.000000	210.000000	210.000000
Модуль упругости при растяжении, ГПа	70.000000	70.000000	70.000000	70.000000	70.000000
Прочность при растяжении, МПа	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
Потребление смолы, г/м2	220.000000	220.000000	220.000000	220.000000	220.000000
Угол нашивки, град	0.000000	0.000000	0.000000	0.000000	0.000000
Шаг нашивки	4.000000	4.000000	4.000000	5.000000	5.000000
Плотность нашивки	57.000000	60.000000	70.000000	47.000000	57.000000

Проверяем размеры данных

In [9]:
```
#Посмотрим количество колонок и столбцов
df.shape
# Итоговый датасет имеет 13 столбцов и 1023 строки, 17 строк из таблицы X_nup бы
```

Out[9]:
```
(1023, 13)
```

Познакомимся с датасетом ближе, проведем разведочный анализ.

Знакомство с данными

In [10]:
```
# Посмотрим на начальные и конечные строки нашего датасета на данном этапе работ
df
```

Out[10]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Пов п
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	
...	...	...	...	...	...	...	
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	

1023 rows × 13 columns

In [11]:
```
#Просмотрим информацию о датасете, проверим тип данных в каждом столбце (типы пр
df.info()
# все переменные содержат значения float64, качественные характеристики отсутств
```

```
<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
```

In [12]:
```
#Поиск уникальных значений с помощью функции nunique
df.nunique()
#Видим в основном общее число уникальных значений в каждом столбце, но в столбце
```

Out[12]:
```
Соотношение матрица-наполнитель         1014
Плотность, кг/м3                        1013
модуль упругости, ГПа                   1020
Количество отвердителя, м.%             1005
Содержание эпоксидных групп,%_2         1004
Температура вспышки, С_2                1003
Поверхностная плотность, г/м2           1004
Модуль упругости при растяжении, ГПа    1004
Прочность при растяжении, МПа           1004
Потребление смолы, г/м2                 1003
Угол нашивки, град                         2
Шаг нашивки                              989
Плотность нашивки                        988
dtype: int64
```

In [13]:
```
# Поработаем со столбцом "Угол нашивки"
```

In [14]:
```
df['Угол нашивки, град'].nunique()
#Так как кол-во уникальных значений в колонке Угол нашивки равно 2, можем привес
```

Out[14]:
```
2
```

In [15]:
```
#Проверим кол-во элементов, где Угол нашивки равен 0 градусов
df['Угол нашивки, град'][df['Угол нашивки, град'] == 0.0].count()
```

Out[15]:
```
520
```

In [16]:
```
# Приведем столбец "Угол нашивки" к значениям 0 и 1 и integer
df = df.replace({'Угол нашивки, град': {0.0 : 0, 90.0 : 1}})
df['Угол нашивки, град'] = df['Угол нашивки, град'].astype(int)
```

In [17]:
```
#Переименуем столбец
df = df.rename(columns={'Угол нашивки, град' : 'Угол нашивки'})
```

```
df
```

Out[17]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Пов п
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	
...	...	...	...	...	...	...	
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	

1023 rows × 13 columns

In [18]:
```
#Посчитаем количество элементов, где угол нашивки равен 0 градусов и убедимся, ч
df['Угол нашивки'][df['Угол нашивки'] == 0.0].count()
#После преобразования колонки Угол нашивки к значениям 0 и 1, кол-во элементов,
```

Out[18]: 520

In [19]:
```
# Переведем столбец с нумерацией в integer
df.index = df.index.astype('int')
```

In [20]:
```
# Сохраним итоговый датасет в отдельную папку с данными, чтобы долго не искать
df.to_excel("Itog\itog.xlsx")
```

In [21]:
```
#Изучим описательную статистику наших данных (максимальное, минимальное, квартил
df.describe()
```

Out[21]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	По
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	

In [22]:
```
a = df.describe()
a.T
```

Out[22]:

	count	mean	std	min	25%	50%	
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.5
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.3
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.8
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.7
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.9
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.6
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.2
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.3
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.3
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.4
Угол нашивки	1023.0	0.491691	0.500175	0.000000	0.000000	0.000000	1.0
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.5
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.9

Описательная статистика содержит по каждому столбцу (по каждой переменной):

- count - количество значений
- mean - среднее значение
- std - стандартное отклонение
- min - минимум
- 25% - верхнее значение первого квартиля
- 50% - медиана
- 75% - верхнее значение третьего квартиля

- max - максимум

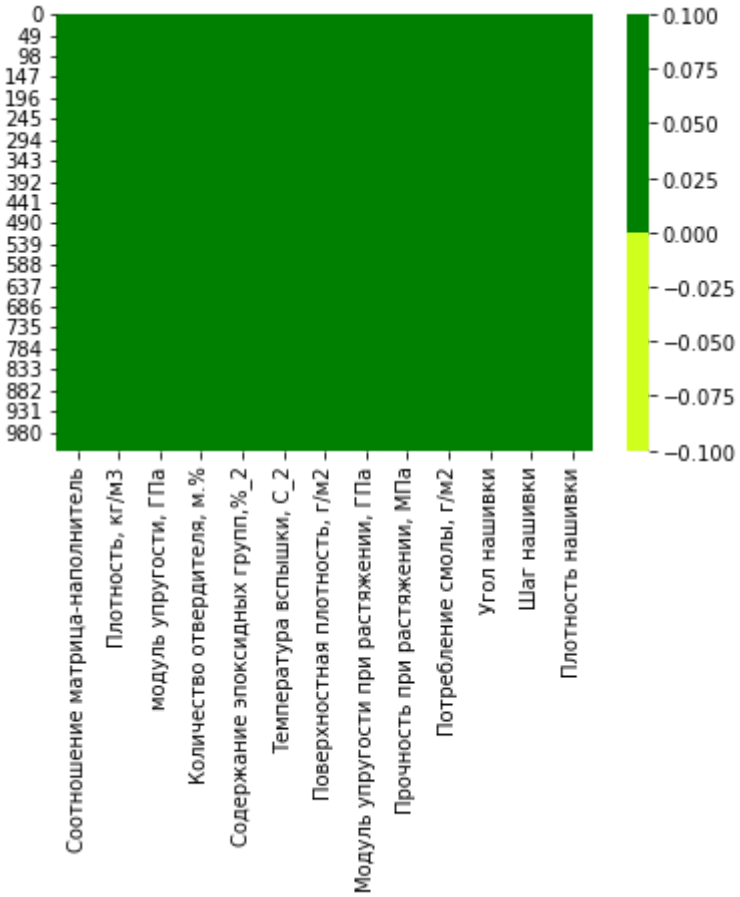
In [23]:
```
# Пропуски данных
```

In [24]:
```
# Проверим на пропущенные данные
df.isnull().sum()
# Пропущенных данных нет = нулевых значений нет, очистка не требуется
```

Out[24]:
```
Соотношение матрица-наполнитель         0
Плотность, кг/м3                        0
модуль упругости, ГПа                   0
Количество отвердителя, м.%             0
Содержание эпоксидных групп,%_2         0
Температура вспышки, С_2                0
Поверхностная плотность, г/м2           0
Модуль упругости при растяжении, ГПа    0
Прочность при растяжении, МПа           0
Потребление смолы, г/м2                 0
Угол нашивки                            0
Шаг нашивки                             0
Плотность нашивки                       0
dtype: int64
```

In [25]:
```
#светло-зеленый - не пропущенные, темнозеленый - пропущенные данные
cols = df.columns
colours = ['#ceff1d', '#008000']
sns.heatmap(df[cols].isnull(), cmap = sns.color_palette(colours))
#Тепловая карта, так же как info() и функция ISNULL() показывает, что пропусков
```

Out[25]: <AxesSubplot:>



In [26]:
```
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
Соотношение матрица-наполнитель - 0%
Плотность, кг/м3 - 0%
модуль упругости, ГПа - 0%
Количество отвердителя, м.% - 0%
Содержание эпоксидных групп,%_2 - 0%
Температура вспышки, С_2 - 0%
Поверхностная плотность, г/м2 - 0%
Модуль упругости при растяжении, ГПа - 0%
Прочность при растяжении, МПа - 0%
Потребление смолы, г/м2 - 0%
Угол нашивки - 0%
Шаг нашивки - 0%
Плотность нашивки - 0%
```

In [27]:
```
#Дубликаты
```

In [28]:
```
# Проверим датасет на дубликаты
df.duplicated().sum()
#Дубликатов нет
```

Out[28]:
```
0
```

In [29]:
```
#По заданию необходимо получить среднее, медианное значение для каждой колонки
#среднее значение
```

In [30]:
```
#получим среднее и медианное значения данных в колонках
mean_and_50 = df.describe()
mean_and_50.loc[['mean', '50%']]
#в целом мы видим близкие друг к другу значения
```

Out[30]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	По г
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	

In [31]:
```
# среднее значение
```

In [32]:
```
df.mean()
```

Out[32]:
```
Соотношение матрица-наполнитель           2.930366
Плотность, кг/м3                       1975.734888
модуль упругости, ГПа                   739.923233
Количество отвердителя, м.%             110.570769
Содержание эпоксидных групп,%_2          22.244390
Температура вспышки, С_2                285.882151
Поверхностная плотность, г/м2           482.731833
Модуль упругости при растяжении, ГПа     73.328571
Прочность при растяжении, МПа          2466.922843
Потребление смолы, г/м2                 218.423144
Угол нашивки                              0.491691
Шаг нашивки                               6.899222
Плотность нашивки                        57.153929
dtype: float64
```

In [33]:
```
# медианное значение
```

In [34]:
```
df.median()
```

Out[34]:
```
Соотношение матрица-наполнитель           2.906878
Плотность, кг/м3                       1977.621657
модуль упругости, ГПа                   739.664328
Количество отвердителя, м.%             110.564840
Содержание эпоксидных групп,%_2          22.230744
Температура вспышки, С_2                285.896812
Поверхностная плотность, г/м2           451.864365
Модуль упругости при растяжении, ГПа     73.268805
Прочность при растяжении, МПа          2459.524526
Потребление смолы, г/м2                 219.198882
Угол нашивки                              0.000000
Шаг нашивки                               6.916144
Плотность нашивки                        57.341920
dtype: float64
```

In [35]:
```
# Вычисляем коэффициенты ранговой корреляции Кендалла. Статистической зависимост
df.corr(method = 'kendall')
```

Out[35]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Темпера вспы
Соотношение матрица-наполнитель	1.000000	-0.003135	0.021247	0.001410	0.010180	-0.00
Плотность, кг/м3	-0.003135	1.000000	-0.008059	-0.021963	-0.007758	-0.01
модуль упругости, ГПа	0.021247	-0.008059	1.000000	0.022382	0.002351	0.02
Количество отвердителя, м.%	0.001410	-0.021963	0.022382	1.000000	0.000010	0.05
Содержание эпоксидных групп,%_2	0.010180	-0.007758	0.002351	0.000010	1.000000	-0.00
Температура вспышки, С_2	-0.009480	-0.019947	0.021028	0.059034	-0.002170	1.00
Поверхностная плотность, г/м2	-0.002060	0.037302	-0.000442	0.033110	-0.006859	0.01
Модуль упругости при растяжении, ГПа	-0.004157	-0.021151	0.005458	-0.043140	0.041994	0.01
Прочность при растяжении, МПа	0.011614	-0.047426	0.022959	-0.046507	-0.013441	-0.01
Потребление смолы, г/м2	0.035145	-0.017079	0.005169	-0.003677	0.009756	0.03
Угол нашивки	-0.021395	-0.051525	-0.031695	0.024690	0.004668	0.01
Шаг нашивки	0.022723	-0.031220	-0.008305	0.006232	-0.004539	0.02
Плотность нашивки	0.002788	0.052935	0.049347	0.016607	-0.021968	0.00

In [36]:
```
#Вычисляем коэффициенты корреляции Пирсона. Статистической зависимости не наблюд
df.corr(method ='pearson')
```

Out[36]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Темпера вспы
Соотношение матрица-наполнитель	1.000000	0.003841	0.031700	-0.006445	0.019766	-0.00
Плотность, кг/м3	0.003841	1.000000	-0.009647	-0.035911	-0.008278	-0.02
модуль упругости, ГПа	0.031700	-0.009647	1.000000	0.024049	-0.006804	0.03
Количество отвердителя, м.%	-0.006445	-0.035911	0.024049	1.000000	-0.000684	0.09
Содержание эпоксидных групп,%_2	0.019766	-0.008278	-0.006804	-0.000684	1.000000	-0.00
Температура вспышки, С_2	-0.004776	-0.020695	0.031174	0.095193	-0.009769	1.00
Поверхностная плотность, г/м2	-0.006272	0.044930	-0.005306	0.055198	-0.012940	0.02
Модуль упругости при растяжении, ГПа	-0.008411	-0.017602	0.023267	-0.065929	0.056828	0.02
Прочность при растяжении, МПа	0.024148	-0.069981	0.041868	-0.075375	-0.023899	-0.03
Потребление смолы, г/м2	0.072531	-0.015937	0.001840	0.007446	0.015165	0.05
Угол нашивки	-0.031073	-0.068474	-0.025417	0.038570	0.008052	0.02
Шаг нашивки	0.036437	-0.061015	-0.009875	0.014887	0.003022	0.02
Плотность нашивки	-0.004652	0.080304	0.056346	0.017248	-0.039073	0.01

In [37]:
```
#Создадим переменную для названия всех столбцов. Это нам пригодится при построен
df.columns
#column_names = ["Соотношение матрица-наполнитель","Плотность, кг/м3","модуль уп
#          "Содержание эпоксидных групп,%_2","Температура вспышки, С_2","Поверхно
#          "Модуль упругости при растяжении, ГПа","Прочность при растяжении, МПа"
#         "Угол нашивки, град","Шаг нашивки","Плотность нашивки"]
column_names = df.columns
```

Визуализируем сырые данные и проведем анализ

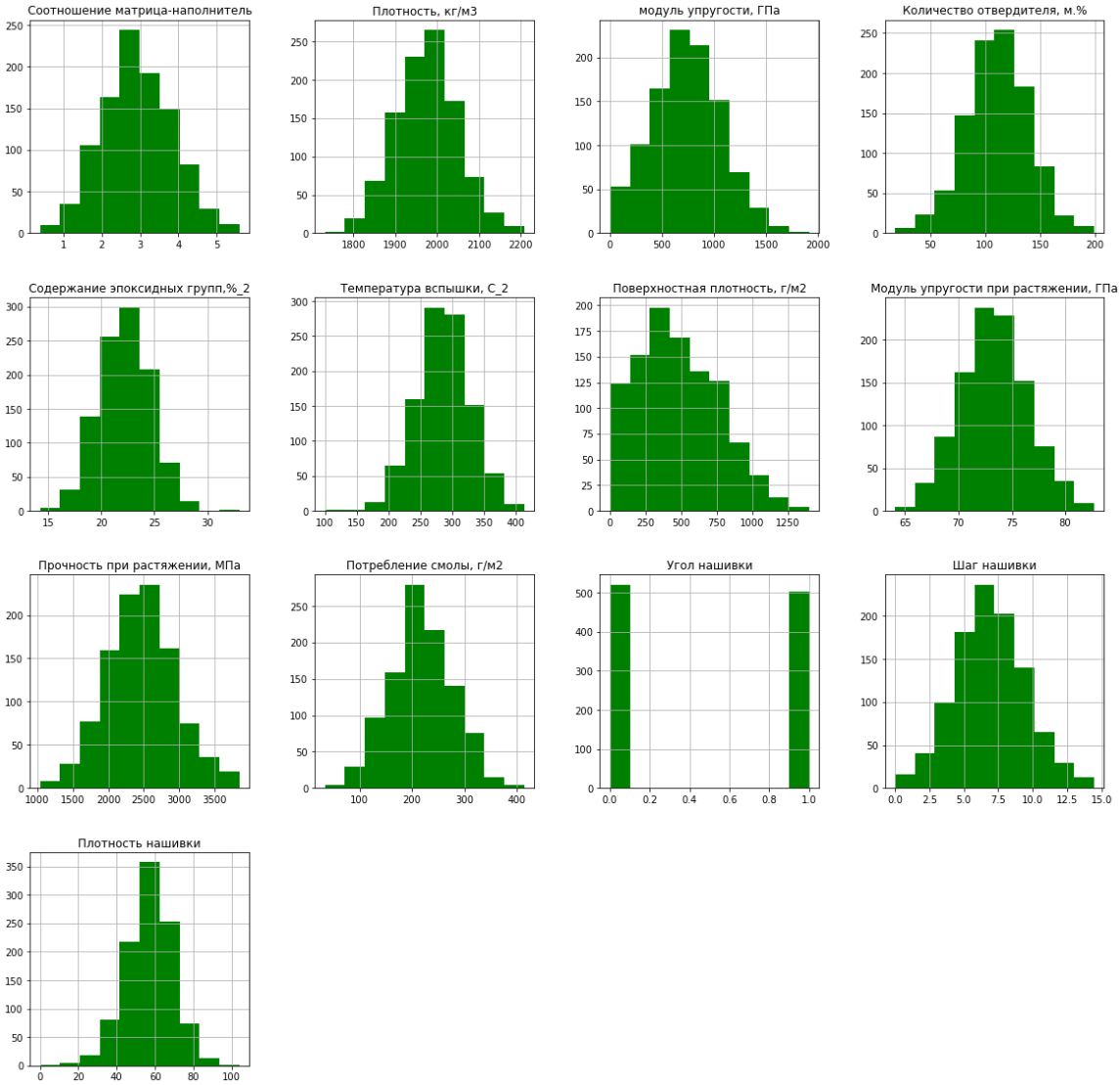
- Построим гистограммы распределения каждой из переменных и боксплоты (несколько разных способов визуализации),
- диаграммы "ящиков с усами" (несколько вариантов),
- попарные графики рассеяния точек (несколько вариантов)
- графики квантиль-квантиль

без нормализации и исключения шумов

Т.к. беглый взгляд на общий файл и дополнительный анализ в excel не дал каких-то явных и бросающихся в глаза закономерностей, то используем разные варианты визуализации в надежде, что получится увидеть какую-то корреляцию. И разные варианты одного и того же типа визуализации используются для отображения результата, потому что какие-то графики отображаются в jupiter, но не работают в colab, какие-то не работают в Github

Показатели описательной статистики и визуализация гистограмм и/или диа-грамм размаха («ящик с усами») позволяют получить наглядное представление о характерах распределений переменных. Такое частотное распределение показывает, какие именно конкретные значения или диапазоны значений исследуемой переменной встречаются наиболее часто, насколько различаются эти значения, расположено ли большинство наблюдений около среднего значения, является распределение симметричным или асимметричным, многомодальным (т.е. имеет две или более вершины) или одномодальным и т.д. По форме распределения можно судить о природе исследуемой переменной (например, бимодальное распределение позволяет предположить, что выборка не является однородной и содержит наблюдения, принадлежащие двум различным множествам, которые в свою очередь нормально распределены).

In [38]:
```
# Построим гистограммы распределения каждой из переменных без нормализации и иск
df.hist(figsize = (20,20), color = "g")
plt.show()
```

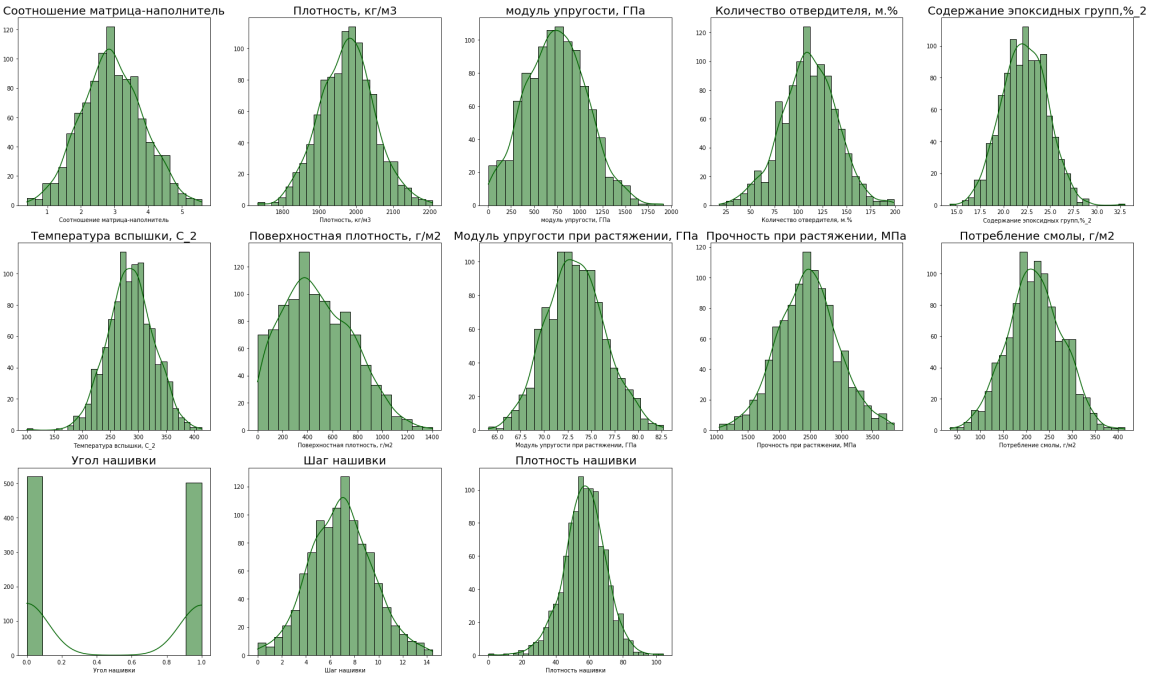


При проведении анализа выявлены параметры близкие к нормальному: Соотношение матрица-наполнитель; Плотность, кг/м3; Модуль упругости, Гпа; Количество отвердителя, м.%; Содержание эпоксидных групп,%_2; Температура вспышки, С_2; Поверхностная плотность, г/м2; Модуль упругости при растяжении, Гпа; Прочность при растяжении, Мпа; Потребление смолы, г/м2; Шаг нашивки; Плотность нашивки. Преимущественно данные стремятся к нормальному распределению. Угол нашивки, как и отражено в датасете, имеет только два значения 90 градусов и 0 градусов, что отражает общий подход к проведению нашивки материалов, а также может быть использовано при обработке данных. Учитывая отсутствие иных показателей для угла нашивки, предлагаем в прогнозе использовать категориальный, а не непрерывный подход при анализе данного параметра.

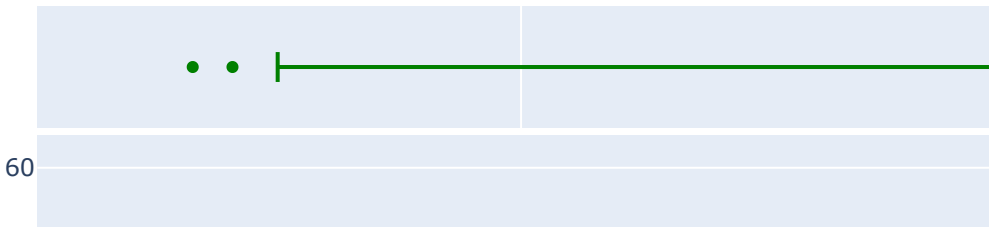
In [39]:
```
# Гистограмма распределения (второй вариант)
a = 5 # количество строк
b = 5 # количество столцбцов
c = 1 # инициализация plot counter
plt.figure(figsize = (35,35))
plt.suptitle('Гистограммы переменных', fontsize = 30)
for col in df.columns:
    plt.subplot(a, b, c)
    #plt.figure(figsize=(7,5))
```

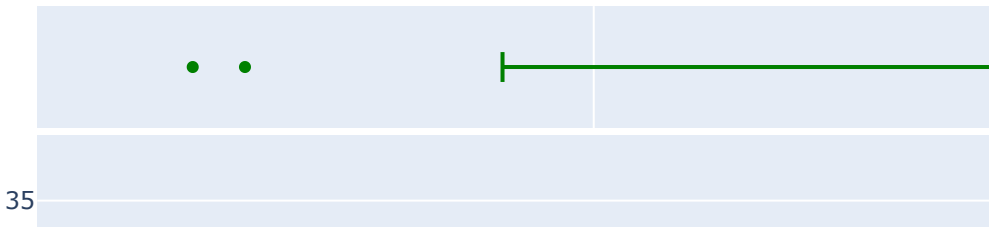
```
    sns.histplot(data = df[col], kde=True, color = "darkgreen")
    plt.ylabel(None)
    plt.title(col, size = 20)
    #plt.show()
    c += 1
#Гистограммы показывают ярковыраженные выбросы в столбцах: плотность, содержание
#Данные стремятся к нормальному распределению практически везде, кроме угла наши
```

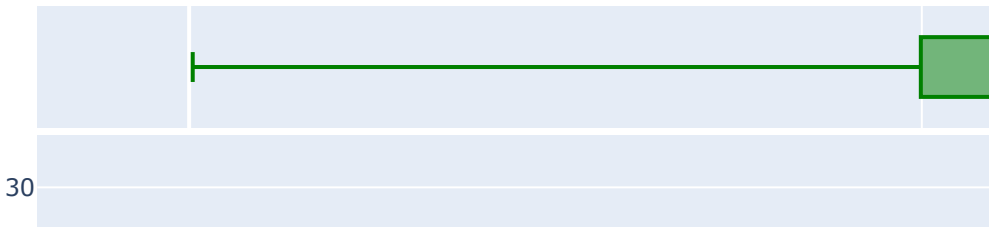
Гистограммы переменных

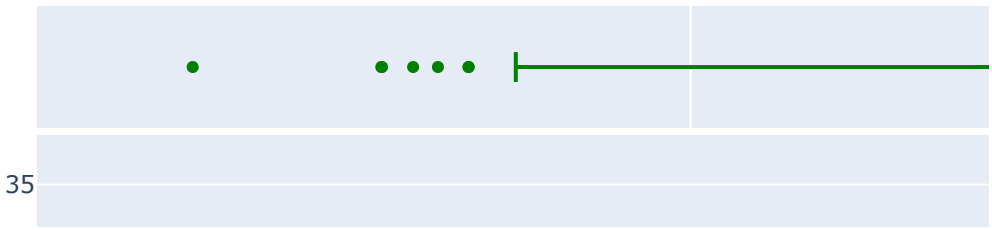


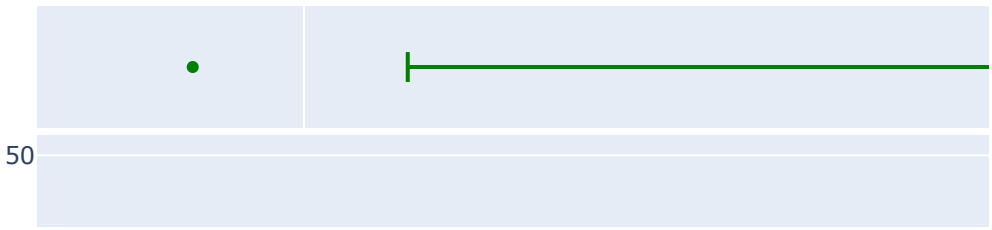
In [40]:
```
# гистограмма распределения и боксплоты (третий вариант)

for column in df.columns:
    fig = px.histogram(df, x = column, color_discrete_sequence = ['green'], nbir
    fig.show()
```

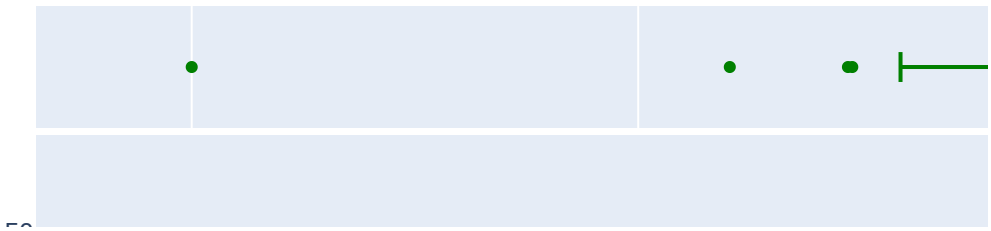

60

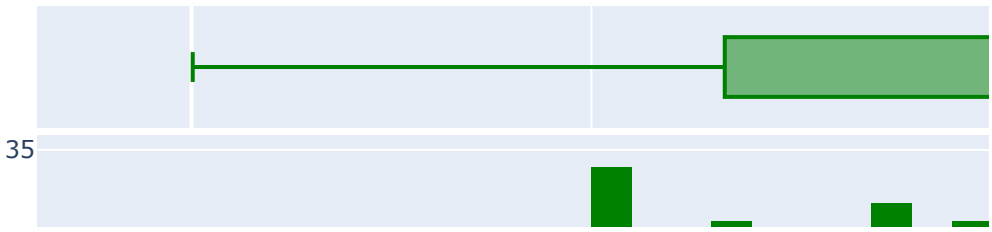

35

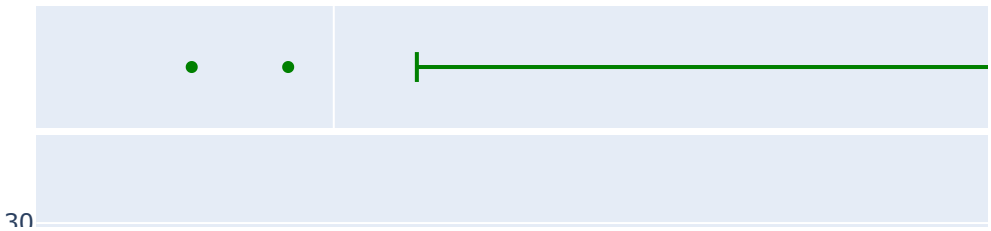

30

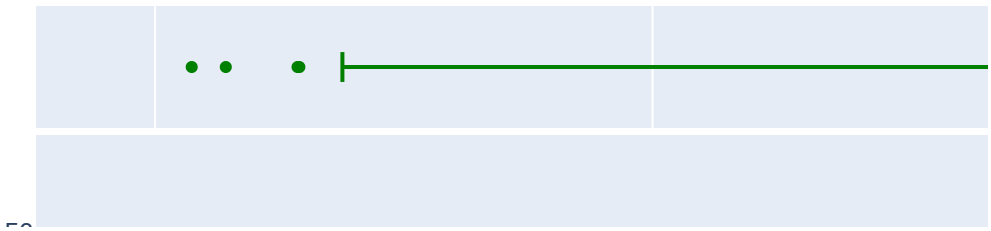

35


50

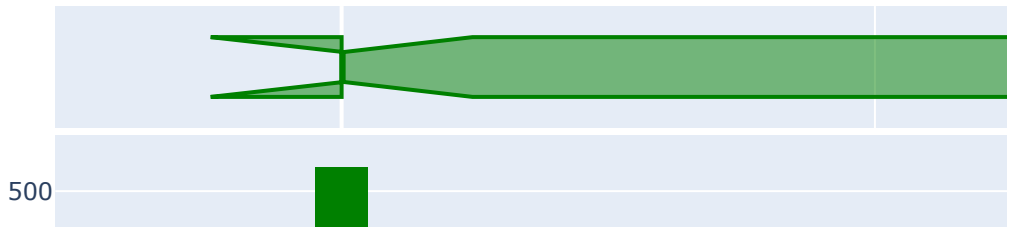


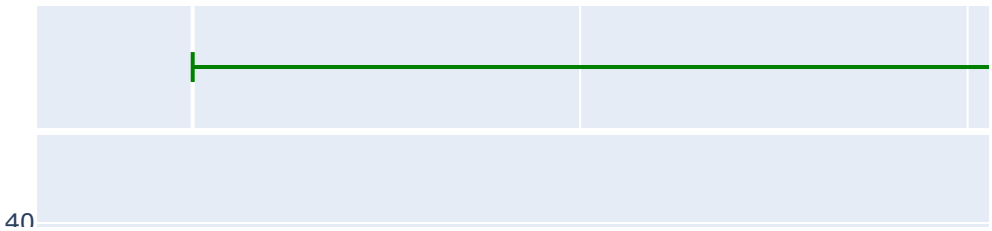

35


30

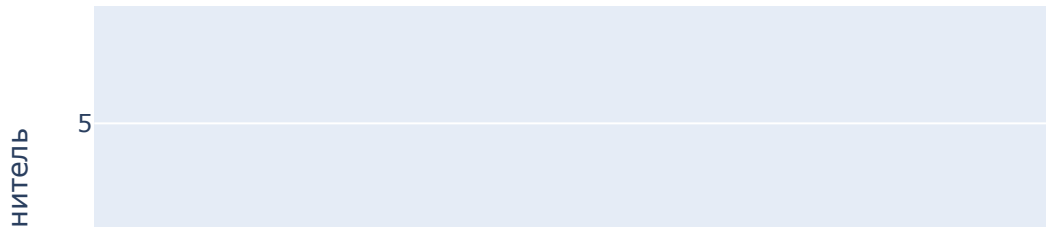



50

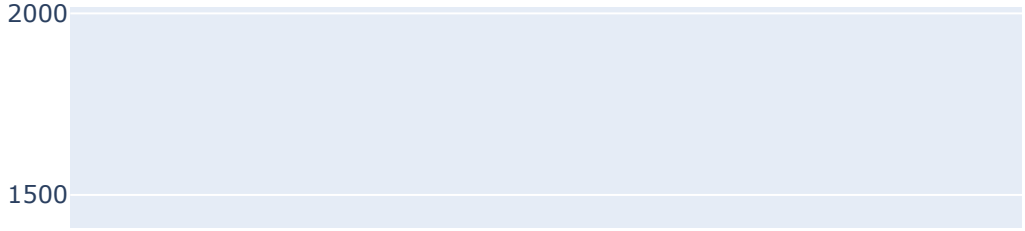

500

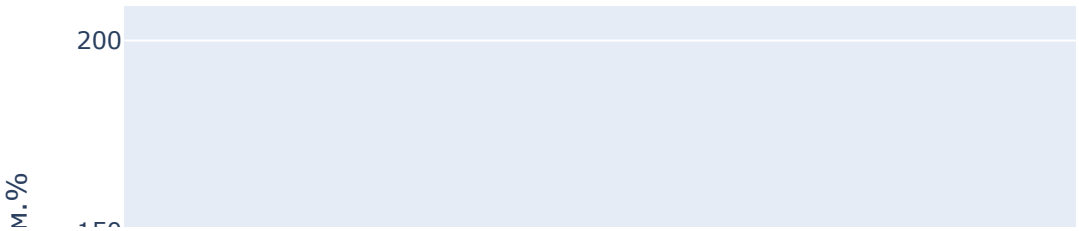

40

In [41]:
```
for column in df.columns:
    fig = px.box(df, y = column)
    fig.show()
```

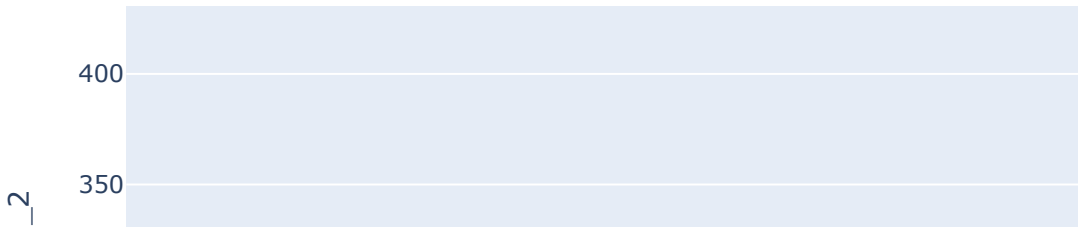

нитель
5

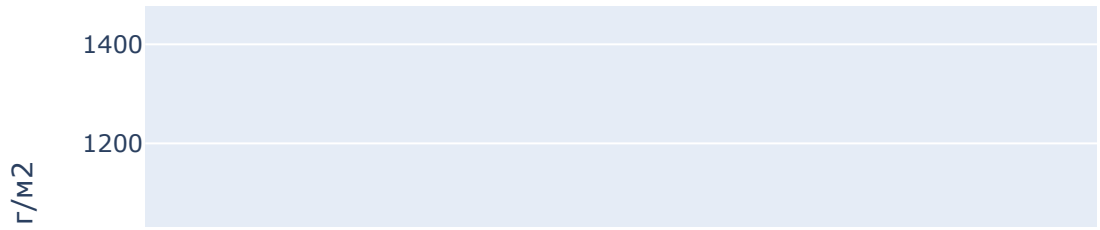

2200
2100

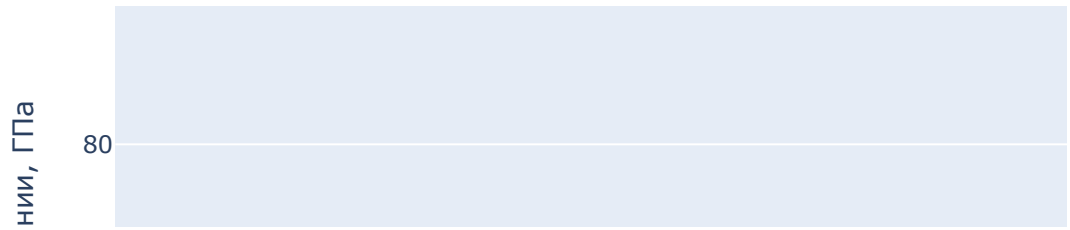

2000
1500

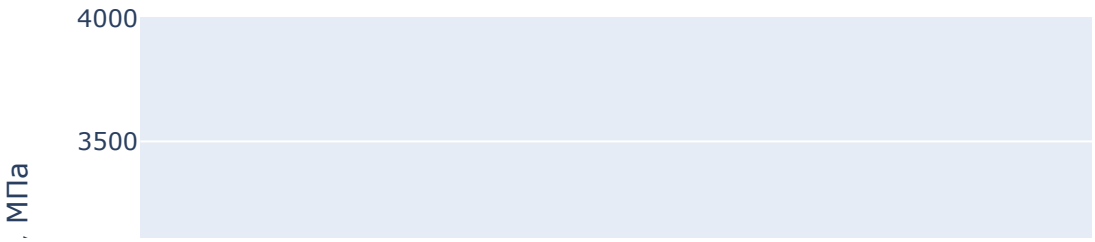

200
м.%

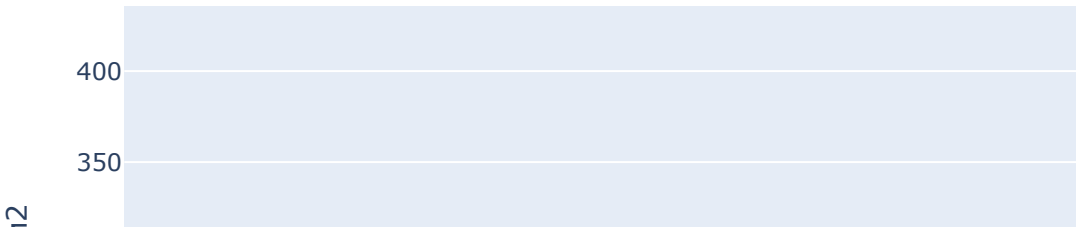
30
ип,%_2

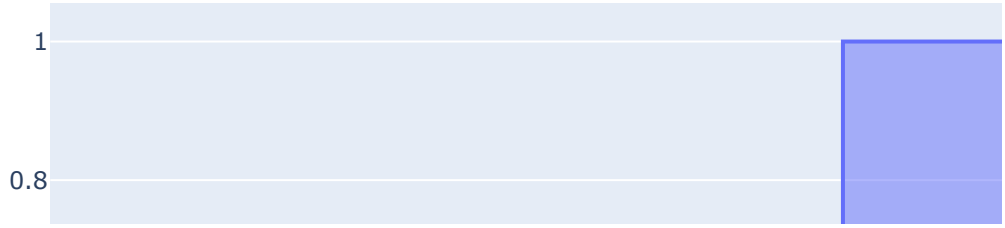

400
350
2

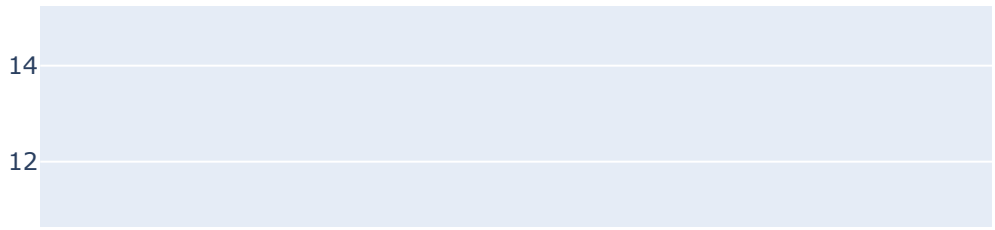

1400
1200
г/м2


нии, ГПа
80

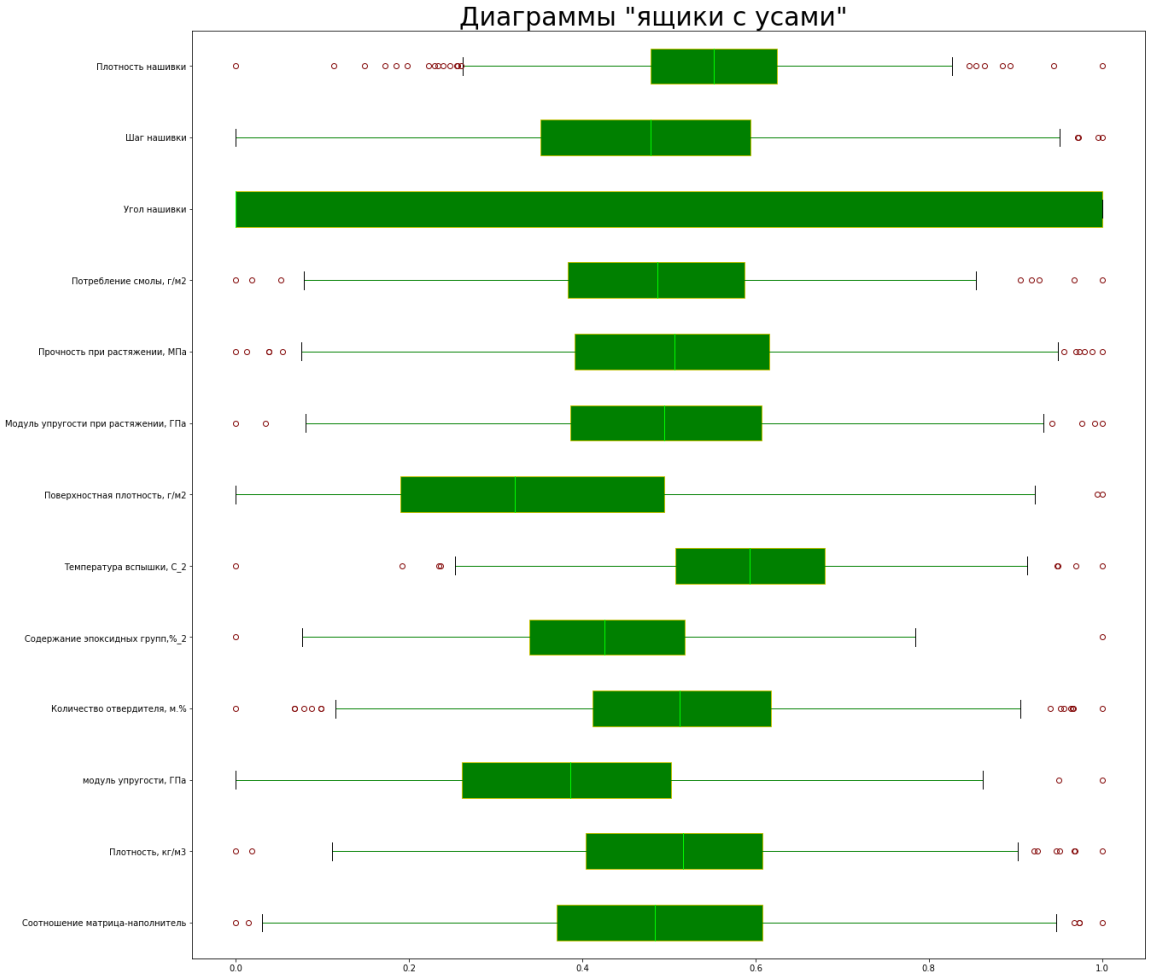

4000
3500
МПа


400
350
I2


1
0.8


14
12

In [42]:
```
# "Ящики с усами"(боксплоты) (первый вариант)
scaler = MinMaxScaler()
scaler.fit(df)
plt.figure(figsize = (20, 20))
plt.suptitle('Диаграммы "ящики с усами"', y = 0.9 ,
             fontsize = 30)
plt.boxplot(pd.DataFrame(scaler.transform(df)), labels = df.columns,patch_artist
plt.show()
```



Многие алгоритмы машинного обучения чувствительны к разбросу и распределению значений признаков обрабатываемых объектов. Соответственно, выбросы во входных данных могут исказить и ввести в заблуждение процесс обучения алгоритмов машинного обучения, что приводит к увеличению времени обучения, снижению точности моделей и, в конечном итоге, к снижению результатов.

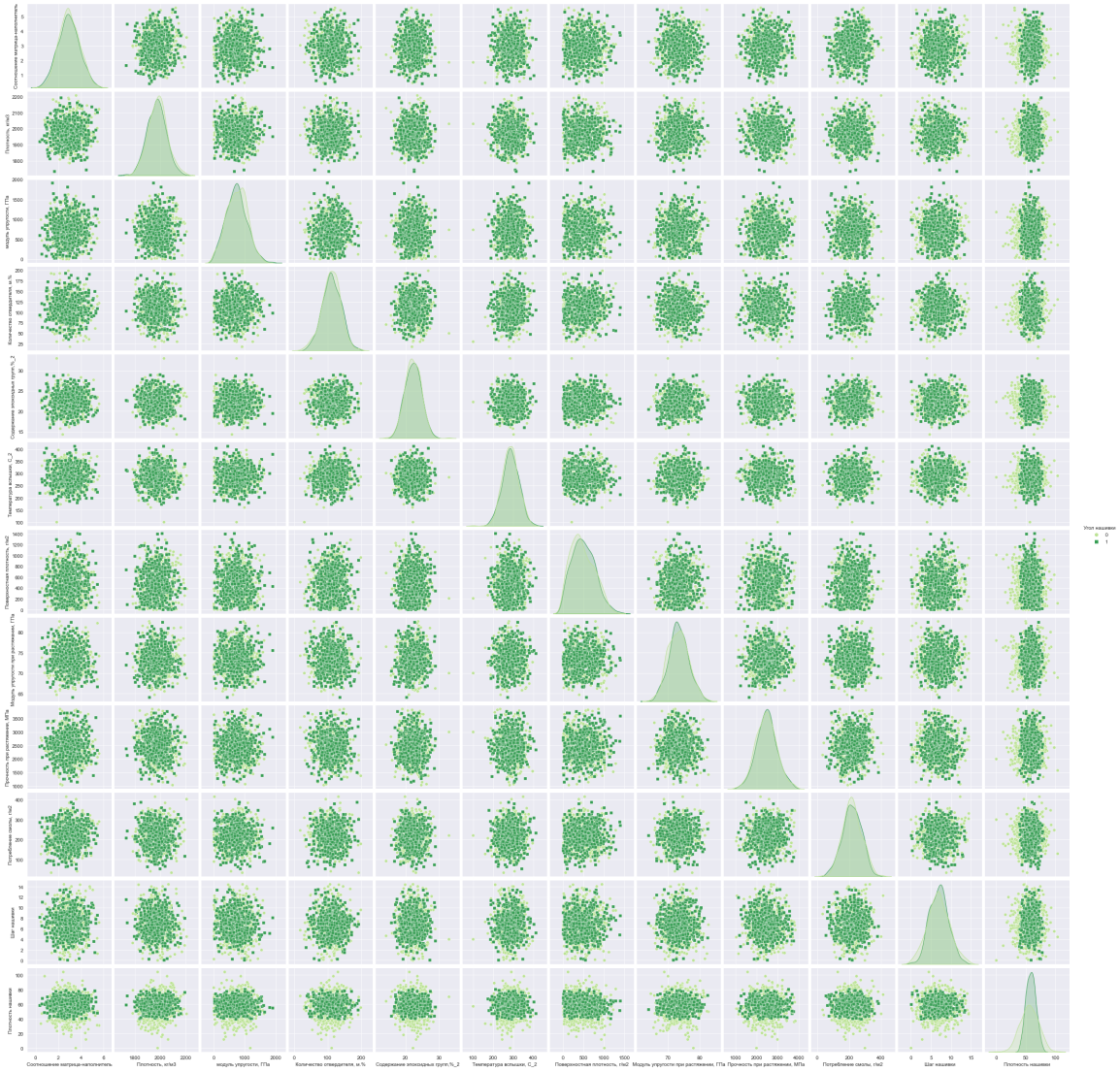
In [ ]:
```
# Ящики с усами (второй вариант)
a = 5 # количество строк
b = 5 # количество столцбцов
c = 1 # инициализация plot counter

plt.figure(figsize = (35,35))
plt.suptitle('Диаграммы "ящики с усами"', y = 0.9 ,
             fontsize = 30)
for col in df.columns:
    plt.subplot(a, b, c)
    #plt.figure(figsize=(7,5))
    sns.boxplot(data = df, y = df[col], fliersize = 15, linewidth = 5, boxprops
    plt.ylabel(None)
    plt.title(col, size = 20)
    #plt.show()
    c += 1
# "Ящики с усами" показывают наличие выбросов во всех столбцах, кроме углов наши
```

In [ ]:
```
# Гистограмма распределения и диаграмма "ящик с усами" вместе с данными по каждо
for column_name in column_names:
    print(column_name)
```

```
    #Гистограмма распределения
    gis = df[column_name]
    sns.set_style("whitegrid")
    sns.kdeplot(data = gis, shade = True, palette ='colorblind', color = "g")
    plt.show()

    #Диаграмма "Ящик с усами"
    sns.boxplot(x=gis, color = "g");
    plt.show()

    #Значения (мин макс ср)
    print("Минимальное значение: ", end = " ")
    print(np.min(gis))
    print("Максимальное значение: ", end=" ")
    print(np.max(gis))
    print("Среднее значение: ", end = " ")
    print(np.mean(gis))

    print("Медианное значение: ", end = " ")
    print(np.median(gis))
    print("\n\n")
# Кроме "Угол нашивки, град" и "Поверхностная плотность, г/м2" остальные перемен
```

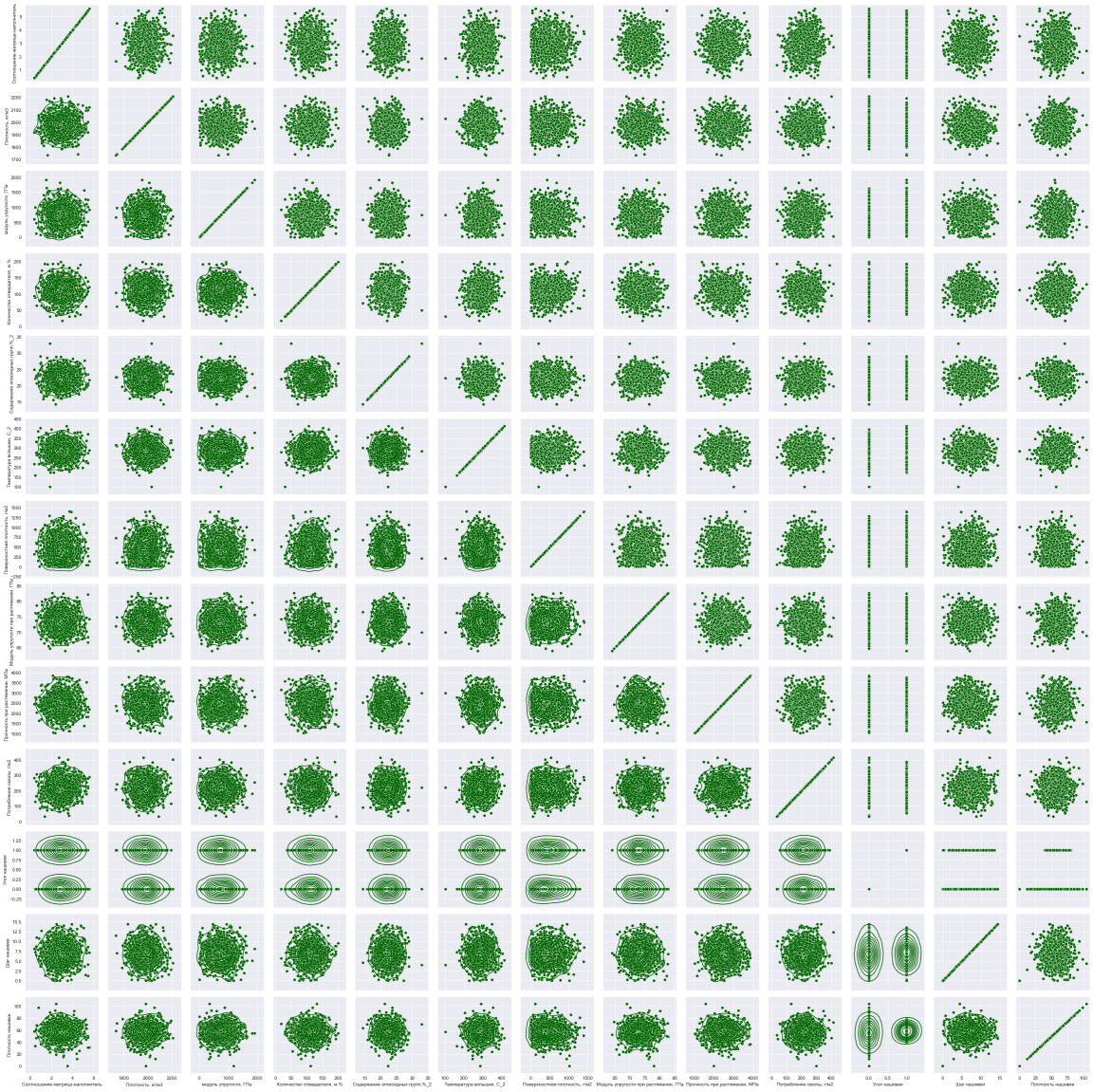
In [45]:
```
# Попарные графики рассеяния точек (матрица диаграмм рассеяния) (первый вариант)
sns.set_style('darkgrid')
sns.pairplot(df, hue = 'Угол нашивки', markers = ["o", "s"], diag_kind = 'auto',
# Попарные графики рассеяния точек так же не показывают какой-либо зависимости м
# из графиков можно наблюдать выбросы, потому что некоторые точки располагаются
# Отсутствие линейной корреляции наверняка подтвердится при построении регрессии
```

Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x1c3db5688e0>
```

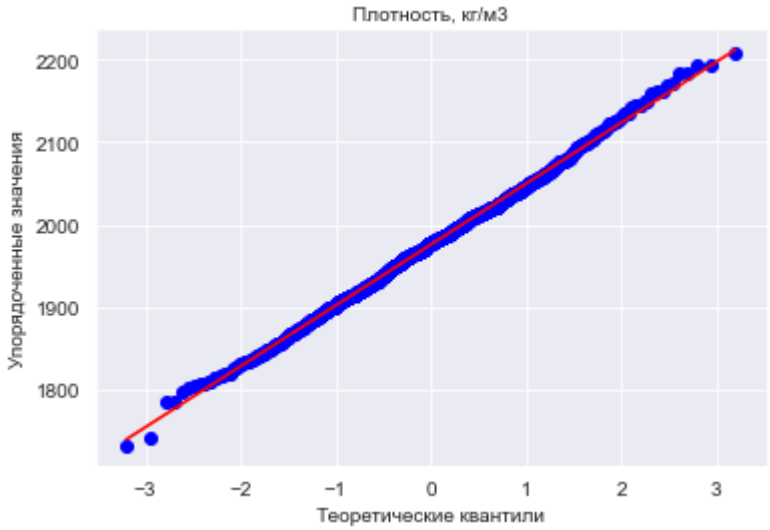


In [46]:
```
# Попарные графики рассеяния точек - скаттерплоты (второй вариант)
g = sns.PairGrid(df[df.columns])
g.map(sns.scatterplot, color = 'darkgreen')
g.map_upper(sns.scatterplot, color = 'darkgreen')
g.map_lower(sns.kdeplot, color = 'darkgreen')
plt.show
# Корреляции нет
```

Out[46]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



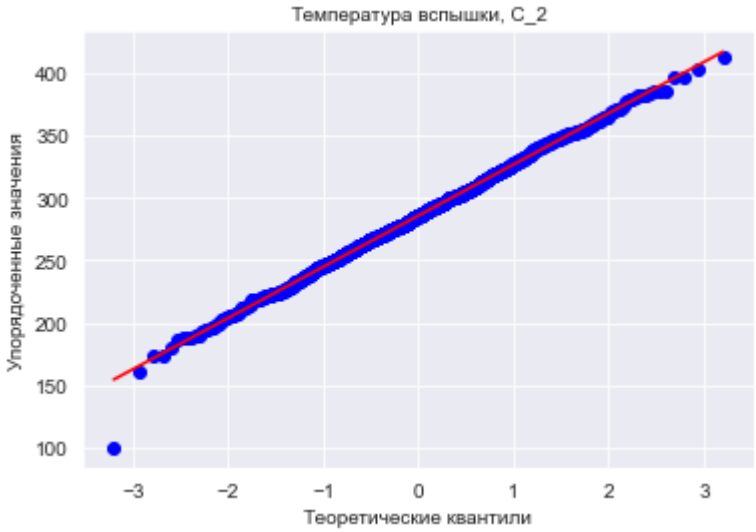
In [47]:
```
# график qq
for i in df.columns:
    plt.figure(figsize = (6, 4))
    res = stats.probplot(df[i], plot = plt)
    plt.title(i, fontsize = 10)
    plt.xlabel("Теоретические квантили", fontsize = 10)
    plt.ylabel("Упорядоченные значения", fontsize = 10)
    plt.show()
```


Плотность, кг/м3
Упорядоченные значения
Теоретические квантили
2200
2100
2000
1900
1800
−3
−2
−1
0
1
2
3


модуль упругости, ГПа
Упорядоченные значения
Теоретические квантили
2000
1500
1000
500
0
−3
−2
−1
0
1
2
3

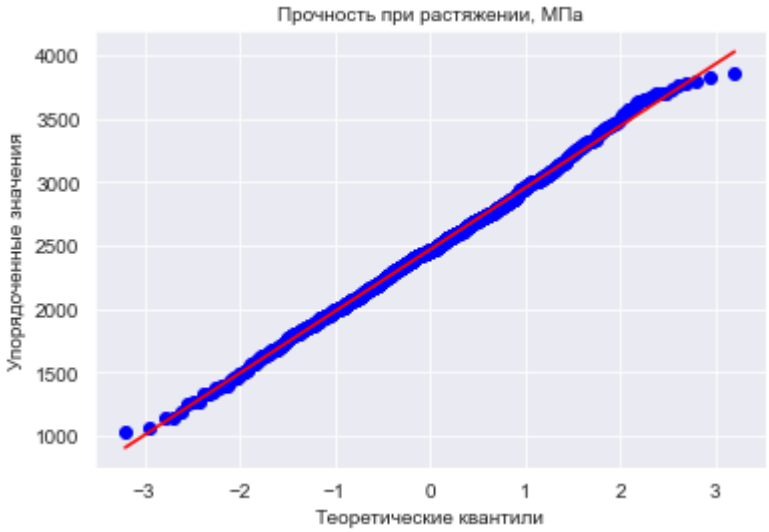

Количество отвердителя, м.%
Упорядоченные значения
Теоретические квантили
200
175
150
125
100
75
50
25
−3
−2
−1
0
1
2
3

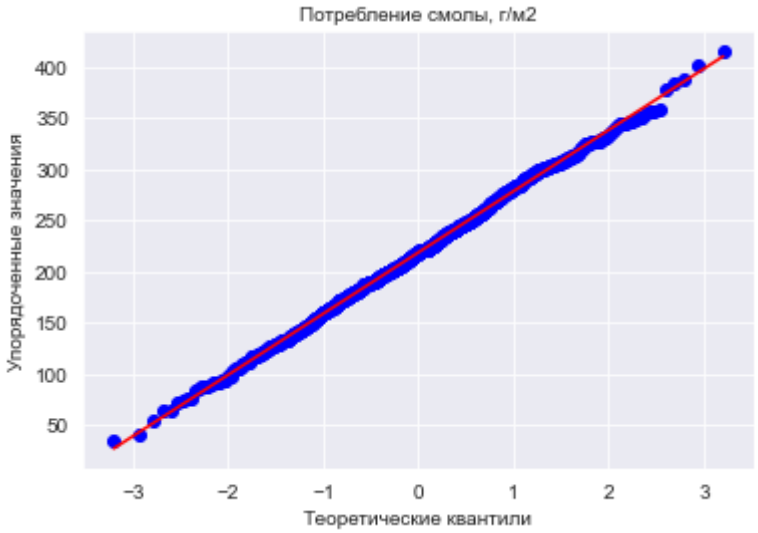

Содержание эпоксидных групп,%_2
Упорядоченные значения
Теоретические квантили

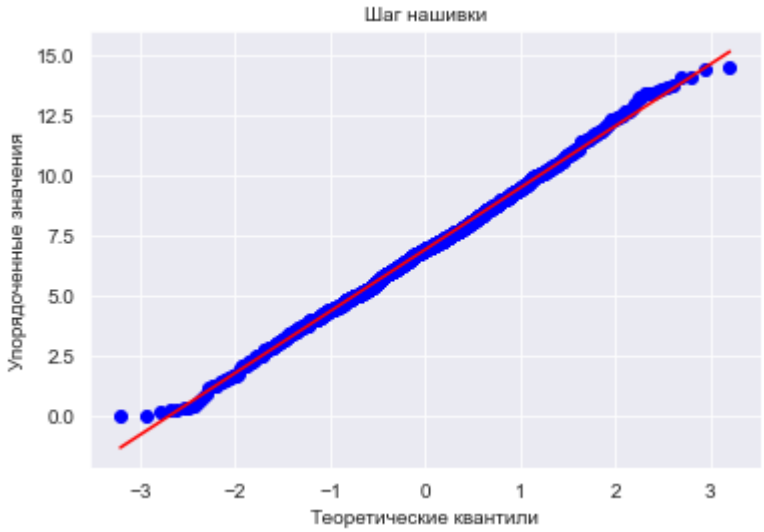

Температура вспышки, С_2
Упорядоченные значения
Теоретические квантили

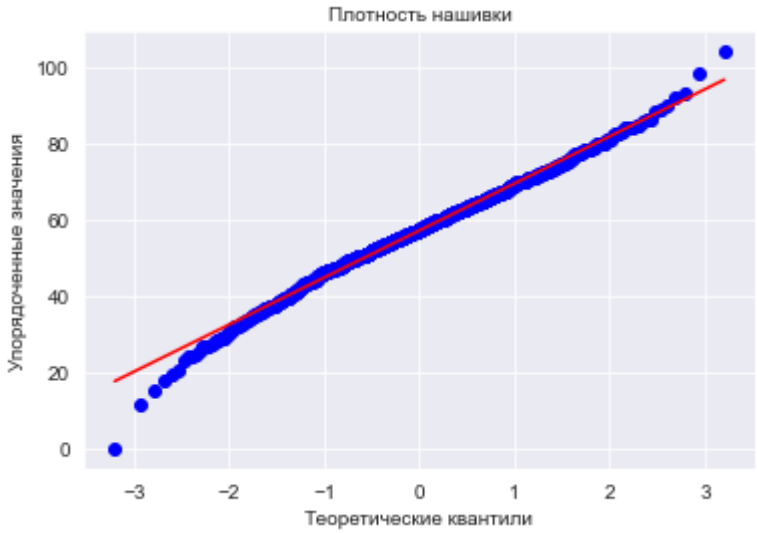

Поверхностная плотность, г/м2
Упорядоченные значения
Теоретические квантили


Модуль упругости при растяжении, ГПа
Упорядоченные значения
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
−3
−2
−1
0
1
2
3
Теоретические квантили

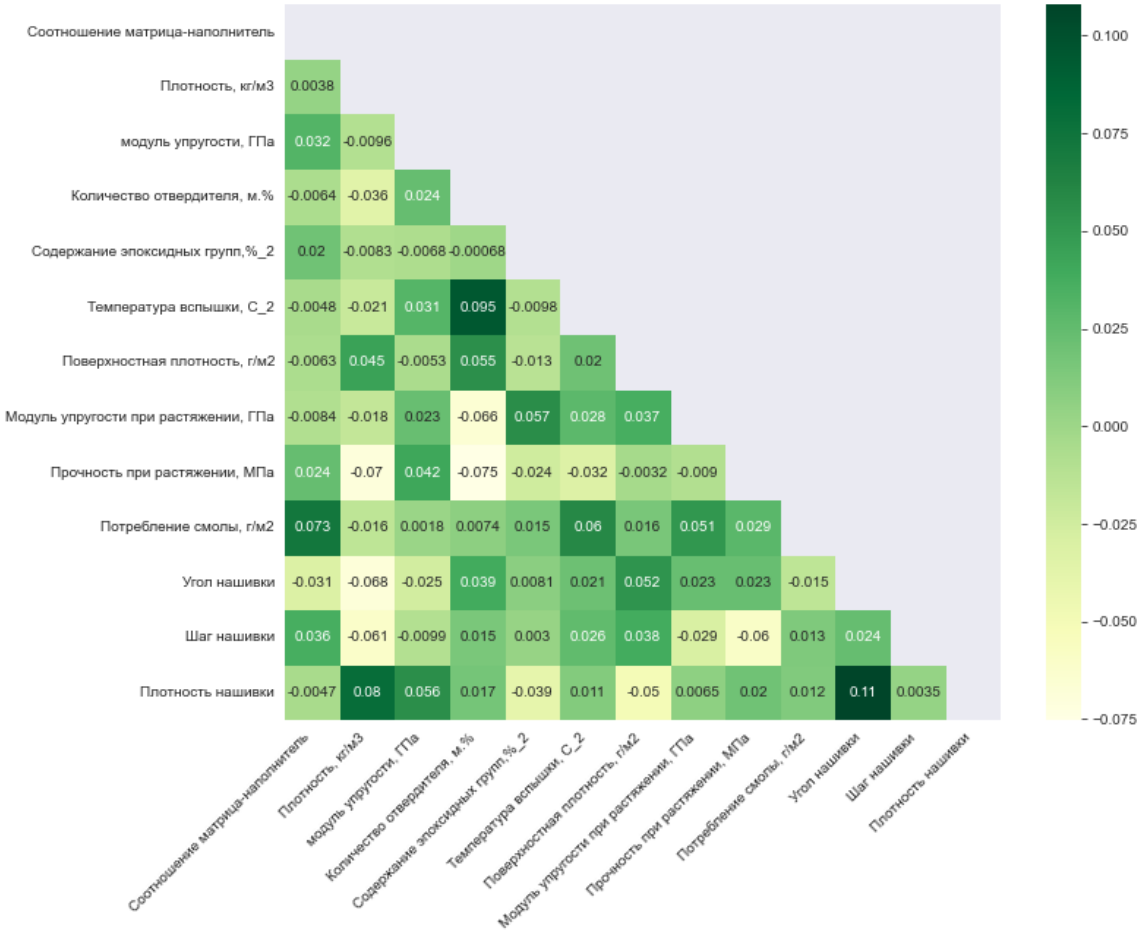

Прочность при растяжении, МПа
Упорядоченные значения
4000
3500
3000
2500
2000
1500
1000
−3
−2
−1
0
1
2
3
Теоретические квантили


Потребление смолы, г/м2
Упорядоченные значения
400
350
300
250
200
150
100
50
−3
−2
−1
0
1
2
3
Теоретические квантили





In [48]:
```
#Визуализация корреляционной матрицы с помощью тепловой карты
mask = np.triu(df.corr())
# Создаем полотно для отображения большого графика
f, ax = plt.subplots(figsize = (11, 9))
# # Визуализируем данные кореляции и создаем цветовую палитру
sns.heatmap(df.corr(), mask = mask, annot = True, square = True, cmap = 'YlGn')
plt.xticks(rotation = 45, ha='right')
plt.show()
# Максимальная корреляция между Плотностью нашивки и углом нашивки и составляет
# Корреляция между всеми параметрами очень близка к 0, что говорит об отсутствии
```



In [49]:
```
# График корреляции подтверждает данные теории композитных материалов. Мы видим,
```

Вывод на данном этапе работы: На наших "сырых" данных мы наблюдаем выбросы в каждом столбце, кроме столбца "Угол нашивки" и корреляция входных переменных очень слабая.