ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу
«Data Science»

Тема: «Прогнозирование конечных свойств новых материалов (композиционных материалов)»

Слушатель: Боркунов Юрий Александрович

Москва, 2023 год

Задание: Прогнозирование конечных свойств новых материалов (композиционных материалов).
На входе: данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.).
На выходе: необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов.

Алгоритм выполнения работы:

- проведение анализа и предобработки данных;
- описание методов, которые предполагается использовать для решения поставленной задачи;
- разработка, обучение и тестирование моделей;
- написание нейронной сети, которая будет рекомендовать соотношение матрица-наполнитель;
- разработка приложения;
- создание репозитория в GitHub / GitLab с кодом исследования.

Произведено объединение двух Excel-таблиц с данными в один датасет.

В параметре "угол нашивки,Град" только 2 значения, поэтому для удобства мы определили переменные как 0 и 1 и переименовали в “угол нашивки”.

	1	3	4	5	6	7	8	9	10	11	...	1013	1014	1015	1016	1017	1018	1019	1020	1021
Соотношение матрица-наполнитель	1.857143	1.857143	2.771331	2.767918	2.569620	2.561475	3.557018	3.532338	2.919678	2.877358	...	2.310394	1.646235	2.806563	3.745862	2.758727	2.271346	3.444022	3.280604	3.705351
Плотность, кг/м3	2030.000000	2030.000000	2030.000000	2000.000000	1910.000000	1900.000000	1930.000000	2100.000000	2160.000000	1990.000000	...	1931.146887	2014.772547	1872.864660	1914.629424	2000.506141	1952.087902	2050.089171	1972.372865	2066.799773
модуль упругости, ГПа	738.736842	738.736842	753.000000	748.000000	807.000000	535.000000	889.000000	1421.000000	933.000000	1628.000000	...	554.010341	841.064806	996.018683	680.683701	934.564388	912.855545	444.732634	416.836524	741.475517
Количество отвердителя, м.%	50.000000	129.000000	111.860000	111.860000	111.860000	111.860000	129.000000	129.000000	129.000000	129.000000	...	96.749782	102.979906	146.199194	110.979100	143.021859	86.992183	145.981978	110.533477	141.397963
Содержание эпоксидных групп,%_2	23.750000	21.250000	22.267857	22.267857	22.267857	22.267857	21.250000	21.250000	21.250000	21.250000	...	22.146487	21.073367	21.559290	25.922635	21.379518	20.123249	19.599769	23.957502	19.246945
Температура вспышки, С_2	284.615385	300.000000	284.615385	284.615385	284.615385	284.615385	300.000000	300.000000	300.000000	300.000000	...	214.827727	271.490843	313.900486	309.796388	273.852679	324.774576	254.215401	248.423047	275.779840
Поверхностная плотность, г/м2	210.000000	210.000000	210.000000	210.000000	210.000000	380.000000	380.000000	1010.000000	1010.000000	1010.000000	...	56.242761	615.168127	799.634090	628.364550	65.105965	209.198700	350.660830	740.142791	641.468152
Модуль упругости при растяжении, ГПа	70.000000	70.000000	70.000000	70.000000	70.000000	75.000000	75.000000	78.000000	78.000000	78.000000	...	78.143609	79.154469	72.815552	76.030555	67.633752	73.090961	72.920827	74.734344	74.042708
Прочность при растяжении, МПа	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	1800.000000	1800.000000	2000.000000	2000.000000	2000.000000	...	1939.307550	2518.516089	2443.482888	2466.925422	3102.539548	2387.292495	2360.392784	2662.906040	2071.715856
Потребление смолы, г/м2	220.000000	220.000000	220.000000	220.000000	220.000000	120.000000	120.000000	300.000000	300.000000	300.000000	...	87.270139	232.428214	307.265172	152.184720	229.780372	125.007669	117.730099	236.606764	197.126067
Угол нашивки	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
Шаг нашивки	4.000000	5.000000	5.000000	5.000000	5.000000	7.000000	7.000000	7.000000	7.000000	9.000000	...	7.683346	5.048503	5.240448	8.057020	8.736592	9.076380	10.565614	4.161154	6.313201
Плотность нашивки	60.000000	47.000000	57.000000	60.000000	70.000000	47.000000	57.000000	60.000000	70.000000	47.000000	...	62.785021	59.837798	52.044507	47.067229	60.277805	47.019770	53.750790	67.629684	58.261074

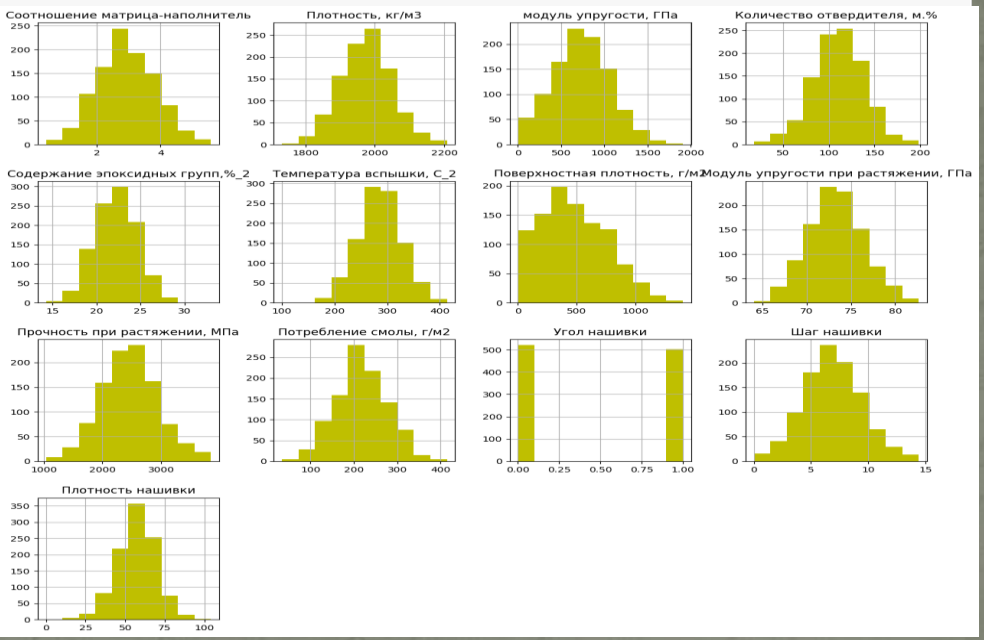
Среднее и медианное значения для каждого столбца

```
df.mean()
```

```
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
```

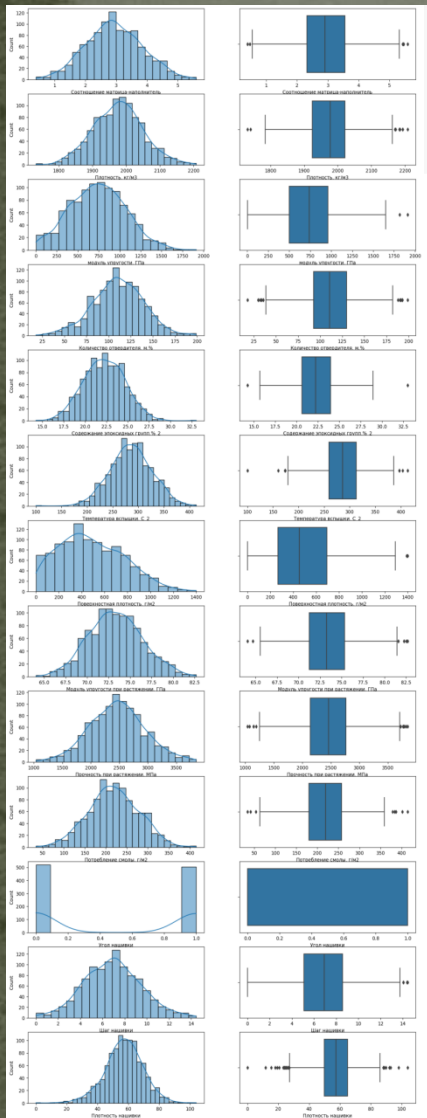
```
[25] df.median()
```

```
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
```



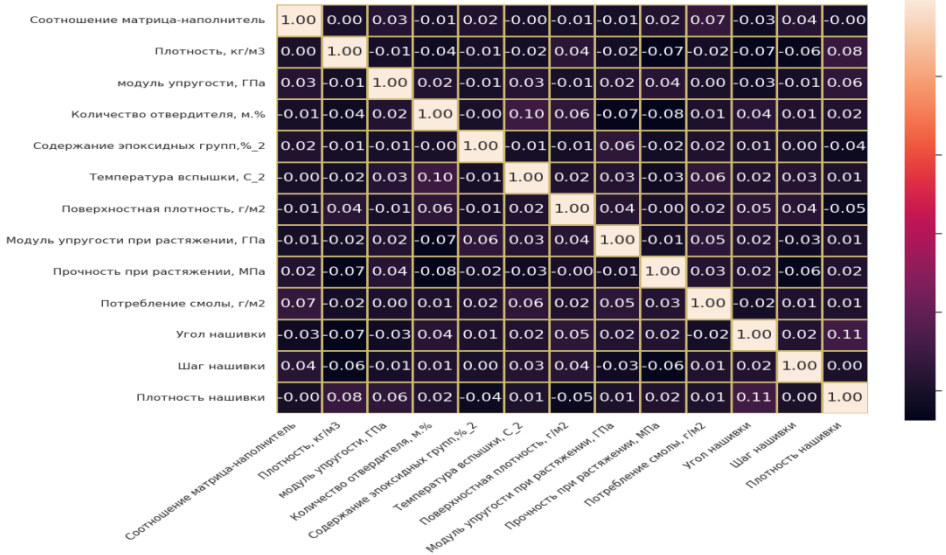
Все признаки, кроме "Угол нашивки" имеют нормальное распределение.

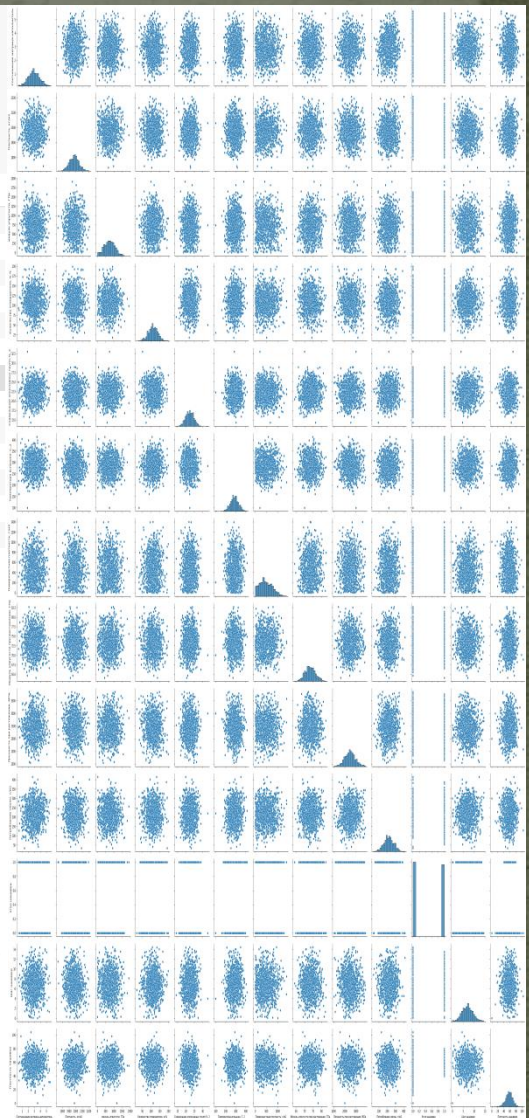
Разведочный анализ данных



```
#Описательная статистика датасета
# 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
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки	1023.0	0.491691	0.500175	0.000000	0.000000	0.000000	1.000000	1.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901





Гистограммы распределения параметров переменных и "ящик с усами"

Корреляционная матрица, представляющую собой тепловую карту, в которой измеряется линейная зависимость между парой признаков

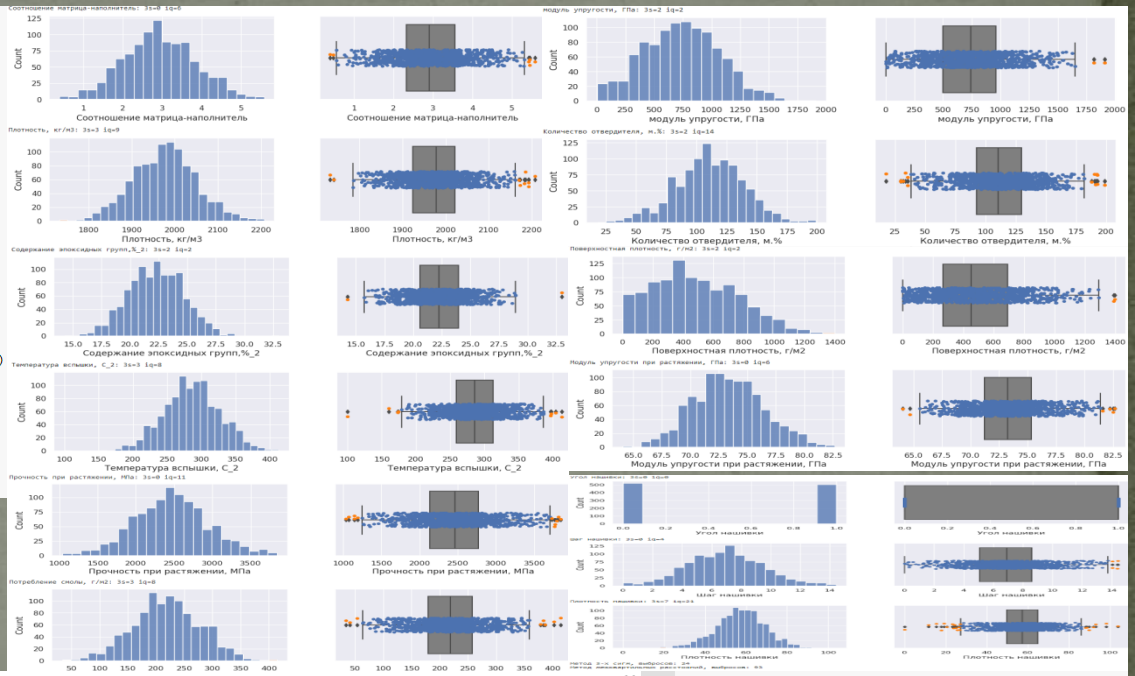
Попарные графики рассеяния точек

Все признаки, кроме "Угол нашивки" имеют нормальное распределение. "Ящики с усами" показывают наличие выбросов во всех столбцах, кроме углов нашивки.

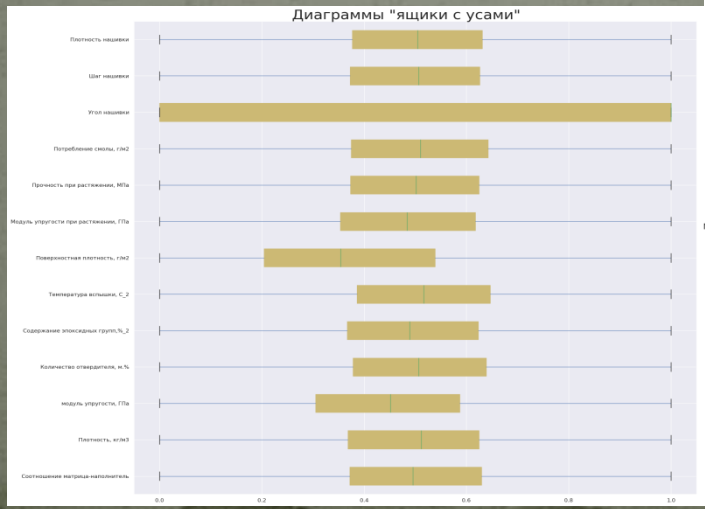
Произведено обнаружение выбросов методом трех сигм и межквартильного расстояния. Правило трех сигм (3-sigma rule) - правило, утверждающее, что вероятность того, что случайная величина отклонится от своего математического ожидания более чем на три среднеквадратических отклонения. Межквартильный диапазон набора данных, часто сокращенно IQR, представляет собой разницу между первым квартилем (25-й процентиль) и третьим квартилем (75-й процентиль) набора данных.

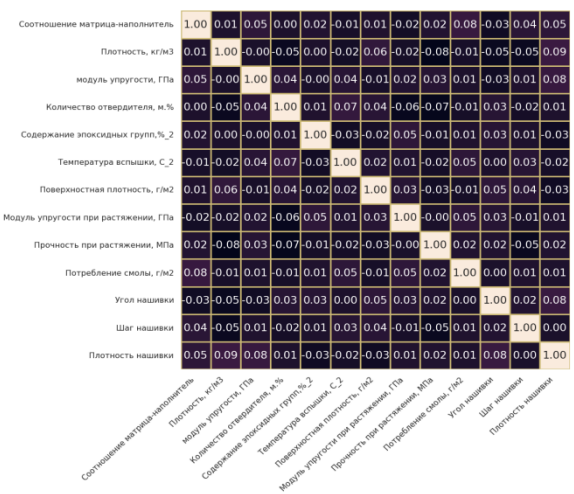
```
count_3s = 0
count_iq = 0
for column in df:
    d = df.loc[:, [column]]
    # методом 3-х сигм
    zscore = (df[column] - df[column].mean()) / df[column].std()
    d['3s'] = zscore.abs() > 3
    count_3s += d['3s'].sum()
    # методом межквартильных расстояний
    q1 = np.quantile(df[column], 0.25)
    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)
    count_iq += d['iq'].sum()
    # визуализация выбросов
    print('{}: 3s={} iq={}'.format(column, d['3s'].sum(), d['iq'].sum()))
    fig, axes = plt.subplots(1, 2, figsize=(12, 2.5))
    sns.histplot(data=d, x=column, hue='3s', multiple='stack', legend=False, ax=axes[0])
    sns.boxplot(data=d, x=column, color='tab:gray', ax=axes[1])
    sns.stripplot(data=d[d['iq']==False], x=column, ax=axes[1])
    sns.stripplot(data=d[d['iq']==True], x=column, color='tab:orange', ax=axes[1])
    plt.show()

print('Метод 3-х сигм, выбросов:', count_3s)
print('Метод межквартильных расстояний, выбросов:', count_iq)
```



Далее осуществлена очистка данных от выбросов методом межквартильного расстояния.






```
# Средние и медианные знчения датасета после исключения выбросов
mean_and_50 = df.describe()
mean_and_50.loc[['mean', '50%']].T
```

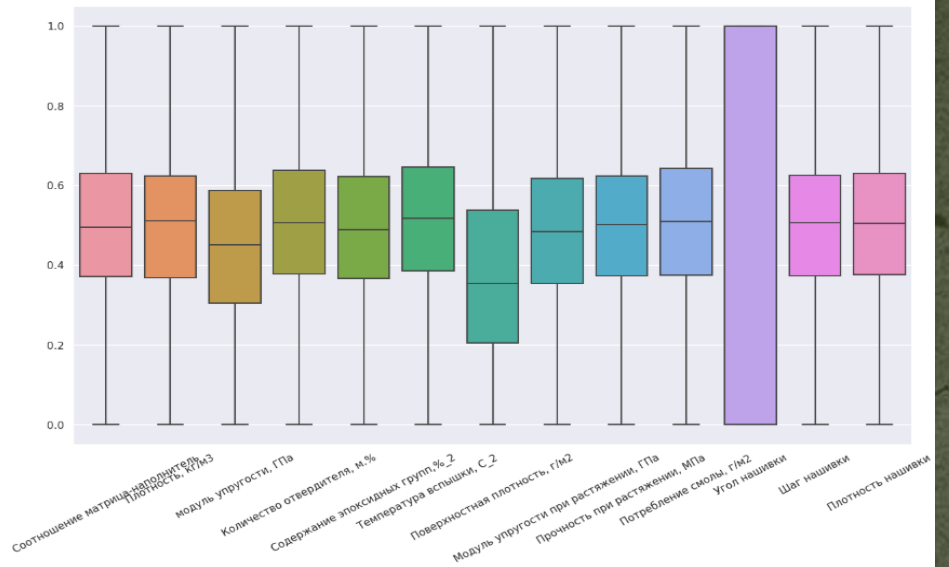
	mean	50%
Соотношение матрица-наполнитель	2.927964	2.907832
Плотность, кг/м3	1974.118744	1977.321002
модуль упругости, ГПа	736.119982	736.178435
Количество отвердителя, м.%	111.136066	111.162090
Содержание эпоксидных групп,%_2	22.200570	22.177681
Температура вспышки, С_2	286.181128	286.220763
Поверхностная плотность, г/м2	482.429070	457.732246
Модуль упругости при растяжении, ГПа	73.303464	73.247594
Прочность при растяжении, МПа	2461.491315	2455.974462
Потребление смолы, г/м2	218.048059	218.697660
Угол нашивки	0.510846	1.000000
Шаг нашивки	6.931939	6.972862
Плотность нашивки	57.562887	57.584225

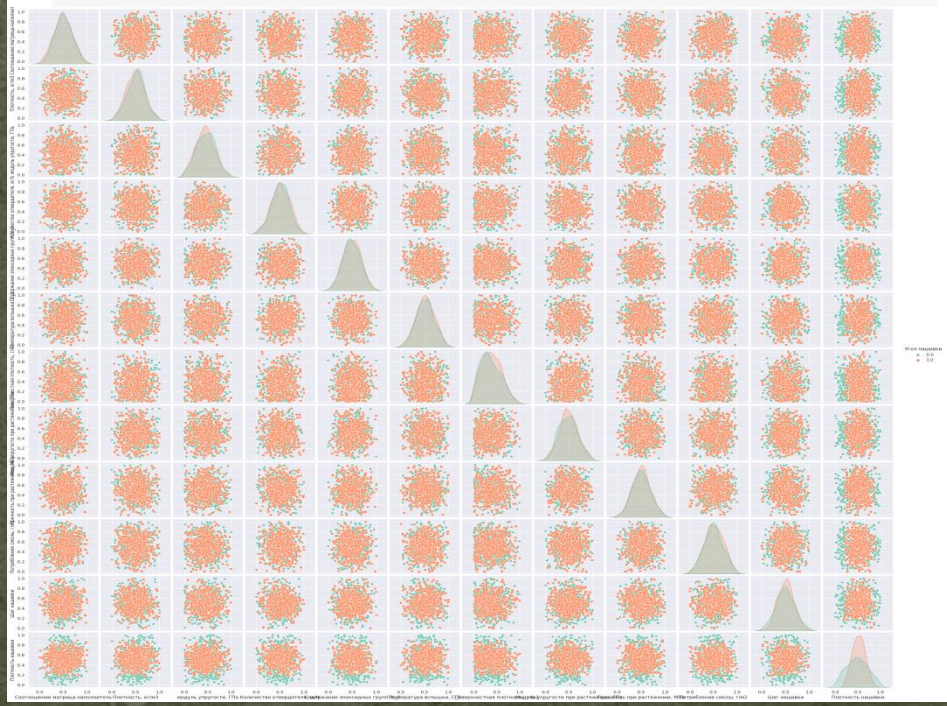
Нормализация данных методом MinMaxScaller()

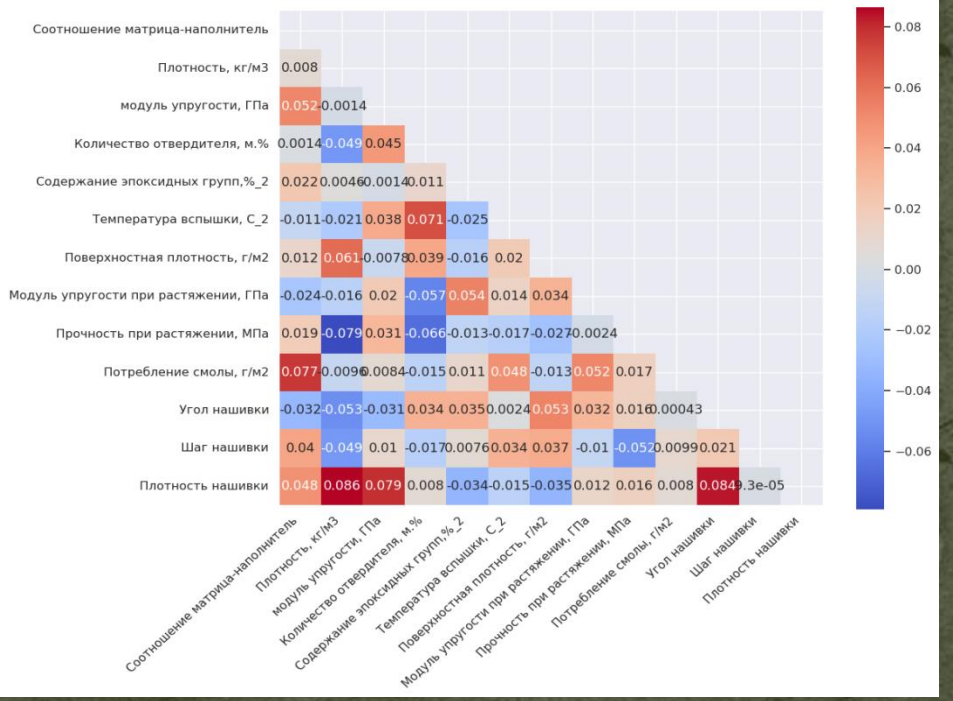
```
min_max_Scaler = preprocessing.MinMaxScaler()
col = df.columns
result = min_max_Scaler.fit_transform(df)
min_max_Scaler_df = pd.DataFrame(result, columns = col)
min_max_Scaler_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	922.0	0.499412	0.187858	0.0	0.371909	0.495189	0.629774	1.0
Плотность, кг/м3	922.0	0.502904	0.188395	0.0	0.368184	0.511396	0.624719	1.0
модуль упругости, ГПа	922.0	0.451341	0.201534	0.0	0.305188	0.451377	0.587193	1.0
Количество отвердителя, м.%	922.0	0.506200	0.186876	0.0	0.378514	0.506382	0.638735	1.0
Содержание эпоксидных групп,%_2	922.0	0.490578	0.180548	0.0	0.366571	0.488852	0.623046	1.0
Температура вспышки, С_2	922.0	0.516739	0.190721	0.0	0.386228	0.516931	0.646553	1.0
Поверхностная плотность, г/м2	922.0	0.373295	0.217269	0.0	0.204335	0.354161	0.538397	1.0
Модуль упругости при растяжении, ГПа	922.0	0.487343	0.196366	0.0	0.353512	0.483718	0.617568	1.0
Прочность при растяжении, МПа	922.0	0.503776	0.188668	0.0	0.373447	0.501481	0.624299	1.0
Потребление смолы, г/м2	922.0	0.507876	0.199418	0.0	0.374647	0.510143	0.642511	1.0
Угол нашивки	922.0	0.510846	0.500154	0.0	0.000000	1.000000	1.000000	1.0
Шаг нашивки	922.0	0.503426	0.183587	0.0	0.372844	0.506414	0.626112	1.0
Плотность нашивки	922.0	0.503938	0.193933	0.0	0.376869	0.504310	0.630842	1.0

```
plt.figure(figsize = (15,9))
ax = sns.boxplot(data = min_max_Scaler_df)
ax.set_xticklabels(ax.get_xticklabels(),rotation=30);
```







Разработка и обучение модели

Перед нами стоит задача регрессии – прогноз на основе выборки объектов с различными признаками. Построим модели для прогноза модуля упругости при растяжении и прочности при растяжении.

Разбиваем данные на обучающую и тестовую выборки

```
X_u = df_norm.drop(['Модуль упругости при растяжении, ГПа'], axis=1)
X_p = df_norm.drop(['Прочность при растяжении, МПа'], axis=1)
y_u = df_norm[['Модуль упругости при растяжении, ГПа']]
y_p = df_norm[['Прочность при растяжении, МПа']]

X_train_u, X_test_u, y_train_u, y_test_u = train_test_split(X_u, y_u, test_size=0.3, random_state=42)
X_train_p, X_test_p, y_train_p, y_test_p = train_test_split(X_p, y_p, test_size=0.3, random_state=42)
```

Метод линейной регрессии.

Определим несколько параметров:

- R2(или Коэффициент детерминации)- это статистическая мера, которая показывает степень вариации зависимой переменной из-за независимой переменной;
- Средняя квадратическая ошибка;
- Средняя абсолютная ошибка.

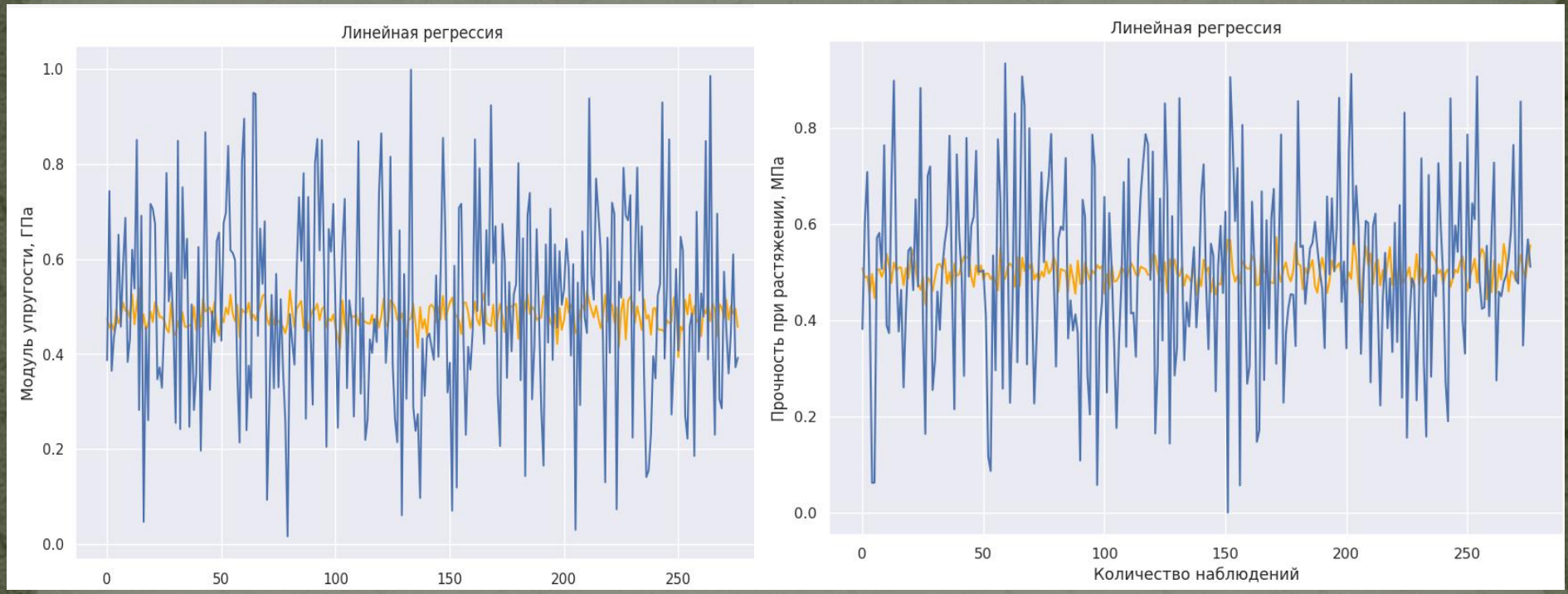
```
model = LinearRegression()
model.fit(X_train_u,y_train_u)
predictions = model.predict(X_test_u)
r_sq = model.score(X_u, y_u)
print('coefficient of determination:', r_sq) # коэффициент детерминации. R2 (или Коэффициент детерминации)
print(
  'mean_squared_error : ', mean_squared_error(y_test_u, predictions)) #Средняя квадратическая ошибка
print(
  'mean_absolute_error : ', mean_absolute_error(y_test_u, predictions)) #Средняя абсолютная ошибка
```

```
coefficient of determination: 0.0054214924703653855
mean_squared_error :  0.042832542319332874
mean_absolute_error :  0.16952044227222318
```

```
model = LinearRegression()
model.fit(X_train_p,y_train_p)
predictions = model.predict(X_test_p)
r_sq = model.score(X_p, y_p)
print('coefficient of determination:', r_sq) # коэффициент детерминации
print(
  'mean_squared_error : ', mean_squared_error(y_test_p, predictions)) #средняя квадратическая ошибка
print(
  'mean_absolute_error : ', mean_absolute_error(y_test_p, predictions)) #Средняя абсолютная ошибка
```

```
coefficient of determination: 0.01040217342102201
mean_squared_error :  0.037238059902490565
mean_absolute_error :  0.1551905647370025
```

Получим прогнозы и выведем соответствующие графики.



графики прогноза и тестовых данных

Метод k-ближайших соседей.

```
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train_u, y_train_u)
y_pred_u_knn = knn.predict(X_test_u)
mae_knr = mean_absolute_error(y_pred_u_knn, y_test_u)
mse_knn_elast = mean_squared_error(y_test_u, y_pred_u_knn)

print('K Neighbors Regressor  Results Train:')
print("Test score: {:.2f}".format(knn.score(X_train_u, y_train_u)))# Оценка тренировочной выборки
print('K Neighbors Regressor  Results:') # результаты регрессора соседей
print('KNN_MAE: ', round(mean_absolute_error(y_test_u, y_pred_u_knn))) # Средняя абсолютная ошибка
print('KNN_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test_u, y_pred_u_knn))) # Средняя абсолютная ошибка в процентах
print('KNN_MSE: {:.2f}'.format(mse_knn_elast)) # средняя квадратическая ошибка
print("KNN_RMSE: {:.2f}".format (np.sqrt(mse_knn_elast))) # Среднеквадратическая ошибка
print("Test score: {:.2f}".format(knn.score(X_test_u, y_test_u)))# Оценка тестовой выборки
```

```
K Neighbors Regressor  Results Train:
Test score: 0.18
K Neighbors Regressor  Results:
KNN_MAE:  0
KNN_MAPE: 0.68
KNN_MSE: 0.05
KNN_RMSE: 0.23
Test score: -0.22
```

Здесь будет применена функция GridSearch — поиск лучших параметров в фиксированной сетке возможных значений.
CV – перекрёстная проверка (кросс-валидация, Cross-validation), метод, который показывает, что модель не переобучилась.
Результаты:
R2-score Модуль упругости при растяжении: -0.014
R2-score Прочность при растяжении: -0.002
R2<0-разработанная модель даёт прогноз даже хуже, чем простое усреднение.

Случайный лес.

Здесь будет применена функция RandomizedSearchCV. Она реализует метод «подгонки» и «оценки». В отличие от GridSearchCV, проверяются не все значения параметров, а из указанных распределений выбирается фиксированное количество значений параметров.

оценочные метрики для модуля упругости при растяжении

```
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train_u, y_train_u)
y_pred_u_knn = knn.predict(X_test_u)
mae_knr = mean_absolute_error(y_pred_u_knn, y_test_u)
mse_knn_elast = mean_squared_error(y_test_u, y_pred_u_knn)

print('K Neighbors Regressor  Results Train:')
print("Test score: {:.2f}".format(knn.score(X_train_u, y_train_u)))# Оценка тренировочной выборки
print('K Neighbors Regressor  Results:') # результаты регрессора соседей
print('KNN_MAE: ', round(mean_absolute_error(y_test_u, y_pred_u_knn))) # Средняя абсолютная ошибка
print('KNN_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test_u, y_pred_u_knn))) # Средняя абсолютная ошибка в процентах
print('KNN_MSE: {:.2f}'.format(mse_knn_elast)) # средняя квадратическая ошибка
print("KNN_RMSE: {:.2f}".format (np.sqrt(mse_knn_elast))) # Среднеквадратическая ошибка
print("Test score: {:.2f}".format(knn.score(X_test_u, y_test_u)))# Оценка тестовой выборки
```

```
K Neighbors Regressor  Results Train:
Test score: 0.18
K Neighbors Regressor  Results:
KNN_MAE:  0
KNN_MAPE: 0.68
KNN_MSE: 0.05
KNN_RMSE: 0.23
Test score: -0.22
```

оценочные метрики для прочности при растяжении

```
rfr = RandomForestRegressor(n_estimators=15, max_depth=7, random_state=33)
rfr.fit(X_train_p, y_train_p.values)
y_pred_p_forest = rfr.predict(X_test_p)
mae_rfr = mean_absolute_error(y_pred_p_forest, y_test_p)
mse_rfr_elast = mean_squared_error(y_test_p, y_pred_p_forest)

print('Random Forest Regressor Results Train:')
print("Test score: {:.2f}".format(rfr.score(X_train_p, y_train_p)))
print('Random Forest Regressor Results:')
print('RF_MAE: ', round(mean_absolute_error(y_test_p, y_pred_p_forest)))
print('RF_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test_p, y_pred_p_forest)))
print('RF_MSE: {:.2f}'.format(mse_rfr_elast))
print("RF_RMSE: {:.2f}".format (np.sqrt(mse_rfr_elast)))
print("Test score: {:.2f}".format(rfr.score(X_test_p, y_test_p)))
```

```
Random Forest Regressor Results Train:
Test score: 0.48
Random Forest Regressor Results:
RF_MAE:  0
RF_MAPE: 8769553567468.01
RF_MSE: 0.04
RF_RMSE: 0.20
```

R2-score Модуль упругости при растяжении: -0.013
R2-score Прочность при растяжении: -0.002

Написание нейронной сети, которая будет рекомендовать соотношение матрица-наполнитель.

```
# Определяем входы и выходы
X_MN = df.drop(['Соотношение матрица-наполнитель'], axis=1)
y_MN = df[['Соотношение матрица-наполнитель']]
# Разбиваем выборки на обучающую и тестовую
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)
```

```
x_array = np.array(df['Соотношение матрица-наполнитель'])
normalized_arr = preprocessing.normalize([x_array])
print(normalized_arr)
```

Применим функцию активации SELU,она сочетает в себе оба преимущества классического RELU со свойствами самонормализации.

```
modelMN = Sequential()
modelMN.add(Dense(128))
modelMN.add(BatchNormalization())
modelMN.add(ReLU())
modelMN.add(Dense(128, activation='selu'))
modelMN.add(BatchNormalization())
modelMN.add(Dense(64, activation='selu'))
modelMN.add(BatchNormalization())
modelMN.add(Dense(32, activation='selu'))
modelMN.add(BatchNormalization())
modelMN.add(ReLU())
modelMN.add(Dense(16, activation='selu'))
modelMN.add(BatchNormalization())
modelMN.add(Dense(1))
modelMN.add(Activation('selu'))
```

Построение модели и определение её параметров.
SGD-стохастический градиентный спуск с мини-пакетами — вариант, при котором коэффициенты меняются после обсчета N элементов выборки, то есть для каждой тренировочной итерации алгоритм выбирает случайное подмножество набора данных. Частота обновления параметров выше, меньше требуется оперативной памяти, эффективность вычислений высокая.

```
modelMN.compile(optimizer=tf.optimizers.SGD(learning_rate=0.02, momentum=0.5),
    loss='mean_absolute_error',metrics=['mse', "mape"])
```

```
tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    min_delta=0,
    patience=10,
    verbose=1,
    mode='auto',
    baseline=None,
    restore_best_weights=False,
    start_from_epoch=0
)# Параметры ранней остановки для претотвращения "переоснащения" модели.
```

```
<keras.callbacks.EarlyStopping at 0x7feaf83fc070>
```

```
 # Минимизируемая функция потерь
loss=keras.losses.SparseCategoricalCrossentropy()
```

```
historyMN=modelMN.fit(
    X_train_MN,
    y_train_MN,
    batch_size = 64,
    epochs=100,
    verbose=1,
    validation_split = 0.2,
        )
```

Оценка модели

```
modelMN.evaluate(X_test_MN, y_test_MN)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 0.7727 - mse: 0.9130 - mape: 33.5275
[0.7727224230766296, 0.9130035638809204, 33.527462005615234]
```

```
scores = modelMN.evaluate(X_train_MN,y_train_MN)
print("\n%s: %.2f%%" % (modelMN.metrics_names[1], scores[1]*100))
```

```
21/21 [==============================] - 0s 1ms/step - loss: 0.7020 - mse: 0.7867 - mape: 31.4950

mse: 78.67%
```

Средняя абсолютная ошибка.
9/9 [==============================] - 0s 2ms/step –
loss: 0.7727 - mse: 0.9130 - mape: 33.5275
Model MAE: [0.7727224230766296, 0.9130035638809204, 33.527462005615234]
MAE среднего значения: Соотношение матрица-наполнитель 0.741552

Разработка приложения

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
    return "WELCOME!!! This is the home page"
if __name__ == "__main__":
    app.run()
```

http://127.0.0.1:5000/

Создание репозитория

https://github.com/YuriKoss/YuriKoss.git