Построение модели нейронной сети которая будет рекомендовать соотношение матрица-наполнитель

Нейро́нная сеть (также иску́сственная нейро́нная сеть, ИНС, или просто нейросе́ть) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации нервных сетей (биологических нейронных сетей) — сетей нервных клеток (нейронов) живого организма.

Импортируем необходимые библиотеки

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle


import warnings

warnings.filterwarnings("ignore")
print(tf.__version__)
```

```
2.19.0
```

Загружаем итоговый набор данных

```
df = pd.read_excel("db_itog_bez_norm.xlsx")
df.drop("Unnamed: 0", axis=1, inplace=True)
df.head()
```

```
   Соотношение матрица-наполнитель  Плотность, кг/м3  модуль
упругости, ГПа  \
0                         1.857143            2030.0
738.736842
1                         1.857143            2030.0
738.736842
2                         1.857143            2030.0
738.736842
3                         2.771331            2030.0
753.000000
4                         2.767918            2000.0
748.000000

   Количество отвердителя, м.%  Поверхностная плотность, г/м2  \
0                        50.00                          210.0
1                        49.90                          210.0
2                       129.00                          210.0
```

```
3                         111.86                                210.0
4                         111.86                                210.0

   Модуль упругости при растяжении, ГПа  Прочность при растяжении, МПа  \
0                                  70.0                          3000.0

1                                  70.0                          3000.0

2                                  70.0                          3000.0

3                                  70.0                          3000.0

4                                  70.0                          3000.0


   Шаг нашивки  Плотность нашивки  Угол нашивки, град_0  \
0          4.0               60.0                     1
1          4.0               70.0                     1
2          5.0               47.0                     1
3          5.0               57.0                     1
4          5.0               60.0                     1

   Угол нашивки, град_90  Содержание эпоксидных групп, г  \
0                      0                       52.250000
1                      0                       72.600000
2                      0                       46.750000
3                      0                       48.989286
4                      0                       48.989286

   Потребление смолы, г/м2 без ЭГ
0                       167.750000
1                       147.400000
2                       173.250000
3                       171.010714
4                       171.010714
```

Подготавливаем датафреймы для создания обучающей и тестовой выборки, а именно исключаем и выносим в отдельный датафрейм переменные в отношении которых необходимо подготовить прогнозирующую модель

```
df_x_smn = df.drop(["Соотношение матрица-наполнитель"], axis=1)
df_y_smn = df[["Соотношение матрица-наполнитель"]]
```

Создание обучающей и тестирующей выборки для прогноза по переменной "Соотношение матрица-наполнитель". Разделение по принципу 70% данных относится к обучающей выборки и 30% относится к тестирующей выборки.

```
X_train_smn, X_test_smn, y_train_smn, y_test_smn =
train_test_split(df_x_smn, df_y_smn, test_size=0.3, random_state=1
)

# Проверяем, что разделение прошло успешно
X_train_smn.shape
```

```
(647, 12)
```

```
# Проверяем, что разделение прошло успешно
X_train_smn.head()
```

```
     Плотность, кг/м3  модуль упругости, ГПа  Количество отвердителя,
м.%  \
529       1862.679792             341.672132
75.068600
638       1856.196526             465.688532
119.594890
421       2097.110886             764.246548
103.418855
682       1976.868448             630.925295
136.600341
245       2005.412966             414.375364
131.173357

     Поверхностная плотность, г/м2  Модуль упругости при растяжении,
ГПа  \
529                       9.046203
73.006935
638                     141.948041
74.222008
421                     711.229254
74.070615
682                     590.865535
73.559943
245                     485.516113
70.838307

     Прочность при растяжении, МПа  Шаг нашивки  Плотность нашивки  \
529                    1399.118555    10.264664          73.738350
638                    2257.615117    10.594981          64.509447
421                    2751.750346     7.927846          37.327435
682                    3033.723583     3.980068          43.107490
245                    2482.710947     9.059405          31.674189

     Угол нашивки, град_0  Угол нашивки, град_90  \
529                     0                      1
638                     0                      1
421                     1                      0
682                     0                      1
```

```
245                         1                           0

     Содержание эпоксидных групп, г  Потребление смолы, г/м2 без ЭГ
529                          41.665671                      118.853356
638                          39.055114                      144.781121
421                          26.837728                       90.462225
682                          58.489555                      175.215897
245                          69.896408                      213.471301
```

Создаем нормализационный слой. Это предварительной обработки, который нормализует непрерывные признаки . Этот слой будет сдвигать и масштабировать входные данные в распределение, центрированное вокруг 0 со стандартным отклонением 1.

```
normalizer = layers.Normalization(axis=-1)
normalizer.adapt(np.array(X_train_smn))
```

Строим Последовательную модель нейросети. Модель строится на основе класса Sequential из библиотеки tensorflow.keras.

```
model_smn = tf.keras.Sequential(
    [
        normalizer,
        layers.Dense(16, activation="tanh"),
        layers.Dense(8, activation="relu"),
        layers.Dense(8, activation="relu"),
        layers.Dense(1),
    ]
)

model_smn.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
    loss="mean_squared_error",
    metrics=[tf.keras.metrics.RootMeanSquaredError()],
)
```

Архитектура нейросети

```
model_smn.summary()

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━
━━━━━┩
│ normalization (Normalization)    │ (None, 12)             │
25 │
```

```
├──────────────────────────────────┼─────────────────────────┼───────
───────┤
│ dense (Dense)                    │ (None, 16)              │
208 │
├──────────────────────────────────┼─────────────────────────┼───────
───────┤
│ dense_1 (Dense)                  │ (None, 8)               │
136 │
├──────────────────────────────────┼─────────────────────────┼───────
───────┤
│ dense_2 (Dense)                  │ (None, 8)               │
72 │
├──────────────────────────────────┼─────────────────────────┼───────
───────┤
│ dense_3 (Dense)                  │ (None, 1)               │
9 │
└──────────────────────────────────┴─────────────────────────┴───────
───────┘

 Total params: 1,302 (5.09 KB)

 Trainable params: 425 (1.66 KB)

 Non-trainable params: 25 (104.00 B)

 Optimizer params: 852 (3.33 KB)

# Обучим модель
model_history = model_smn.fit(X_train_smn, y_train_smn, epochs=100,
verbose=1, validation_split=0.2)

Epoch 1/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 9.6223 -
root_mean_squared_error: 3.1016 - val_loss: 8.4052 -
val_root_mean_squared_error: 2.8992
Epoch 2/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 8.6482 -
root_mean_squared_error: 2.9403 - val_loss: 7.5477 -
val_root_mean_squared_error: 2.7473
Epoch 3/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.5147 -
root_mean_squared_error: 2.7412 - val_loss: 6.7121 -
val_root_mean_squared_error: 2.5908
Epoch 4/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 6.6751 -
root_mean_squared_error: 2.5833 - val_loss: 5.8069 -
val_root_mean_squared_error: 2.4097
Epoch 5/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.0624 -
root_mean_squared_error: 2.4614 - val_loss: 4.8033 -
val_root_mean_squared_error: 2.1917
```

```
Epoch 6/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.2216 -
root_mean_squared_error: 2.2808 - val_loss: 3.7722 -
val_root_mean_squared_error: 1.9422
Epoch 7/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.7897 -
root_mean_squared_error: 1.9461 - val_loss: 2.8226 -
val_root_mean_squared_error: 1.6801
Epoch 8/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 2.8949 -
root_mean_squared_error: 1.6992 - val_loss: 2.0711 -
val_root_mean_squared_error: 1.4391
Epoch 9/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.0722 -
root_mean_squared_error: 1.4389 - val_loss: 1.6016 -
val_root_mean_squared_error: 1.2655
Epoch 10/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.6748 -
root_mean_squared_error: 1.2932 - val_loss: 1.3546 -
val_root_mean_squared_error: 1.1639
Epoch 11/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 1.4157 -
root_mean_squared_error: 1.1895 - val_loss: 1.2556 -
val_root_mean_squared_error: 1.1205
Epoch 12/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.3422 -
root_mean_squared_error: 1.1578 - val_loss: 1.1940 -
val_root_mean_squared_error: 1.0927
Epoch 13/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 1.2278 -
root_mean_squared_error: 1.1078 - val_loss: 1.1527 -
val_root_mean_squared_error: 1.0737
Epoch 14/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.2874 -
root_mean_squared_error: 1.1337 - val_loss: 1.1204 -
val_root_mean_squared_error: 1.0585
Epoch 15/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.2136 -
root_mean_squared_error: 1.1010 - val_loss: 1.0893 -
val_root_mean_squared_error: 1.0437
Epoch 16/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.0045 -
root_mean_squared_error: 1.0016 - val_loss: 1.0672 -
val_root_mean_squared_error: 1.0331
Epoch 17/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.0200 -
root_mean_squared_error: 1.0098 - val_loss: 1.0386 -
val_root_mean_squared_error: 1.0191
Epoch 18/100
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.0500 -
root_mean_squared_error: 1.0244 - val_loss: 1.0099 -
val_root_mean_squared_error: 1.0049
Epoch 19/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.9587 -
root_mean_squared_error: 0.9790 - val_loss: 0.9891 -
val_root_mean_squared_error: 0.9945
Epoch 20/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 1.0023 -
root_mean_squared_error: 1.0002 - val_loss: 0.9674 -
val_root_mean_squared_error: 0.9836
Epoch 21/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 1.0113 -
root_mean_squared_error: 1.0049 - val_loss: 0.9471 -
val_root_mean_squared_error: 0.9732
Epoch 22/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.8907 -
root_mean_squared_error: 0.9436 - val_loss: 0.9334 -
val_root_mean_squared_error: 0.9661
Epoch 23/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.9014 -
root_mean_squared_error: 0.9492 - val_loss: 0.9216 -
val_root_mean_squared_error: 0.9600
Epoch 24/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.9317 -
root_mean_squared_error: 0.9647 - val_loss: 0.9126 -
val_root_mean_squared_error: 0.9553
Epoch 25/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.9735 -
root_mean_squared_error: 0.9838 - val_loss: 0.8993 -
val_root_mean_squared_error: 0.9483
Epoch 26/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.8329 -
root_mean_squared_error: 0.9123 - val_loss: 0.8898 -
val_root_mean_squared_error: 0.9433
Epoch 27/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.8040 -
root_mean_squared_error: 0.8963 - val_loss: 0.8870 -
val_root_mean_squared_error: 0.9418
Epoch 28/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.8573 -
root_mean_squared_error: 0.9245 - val_loss: 0.8825 -
val_root_mean_squared_error: 0.9394
Epoch 29/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.8434 -
root_mean_squared_error: 0.9169 - val_loss: 0.8775 -
val_root_mean_squared_error: 0.9368
Epoch 30/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.7436 -
```

```
root_mean_squared_error: 0.8613 - val_loss: 0.8694 -
val_root_mean_squared_error: 0.9324
Epoch 31/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7464 -
root_mean_squared_error: 0.8638 - val_loss: 0.8703 -
val_root_mean_squared_error: 0.9329
Epoch 32/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7625 -
root_mean_squared_error: 0.8727 - val_loss: 0.8639 -
val_root_mean_squared_error: 0.9295
Epoch 33/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7377 -
root_mean_squared_error: 0.8588 - val_loss: 0.8704 -
val_root_mean_squared_error: 0.9330
Epoch 34/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.7128 -
root_mean_squared_error: 0.8441 - val_loss: 0.8637 -
val_root_mean_squared_error: 0.9293
Epoch 35/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.7760 -
root_mean_squared_error: 0.8803 - val_loss: 0.8650 -
val_root_mean_squared_error: 0.9301
Epoch 36/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7004 -
root_mean_squared_error: 0.8358 - val_loss: 0.8661 -
val_root_mean_squared_error: 0.9306
Epoch 37/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7225 -
root_mean_squared_error: 0.8500 - val_loss: 0.8604 -
val_root_mean_squared_error: 0.9276
Epoch 38/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7000 -
root_mean_squared_error: 0.8365 - val_loss: 0.8538 -
val_root_mean_squared_error: 0.9240
Epoch 39/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6887 -
root_mean_squared_error: 0.8298 - val_loss: 0.8591 -
val_root_mean_squared_error: 0.9269
Epoch 40/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7338 -
root_mean_squared_error: 0.8564 - val_loss: 0.8562 -
val_root_mean_squared_error: 0.9253
Epoch 41/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6743 -
root_mean_squared_error: 0.8209 - val_loss: 0.8557 -
val_root_mean_squared_error: 0.9250
Epoch 42/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6835 -
root_mean_squared_error: 0.8266 - val_loss: 0.8608 -
```

```
val_root_mean_squared_error: 0.9278
Epoch 43/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7389 -
root_mean_squared_error: 0.8586 - val_loss: 0.8587 -
val_root_mean_squared_error: 0.9266
Epoch 44/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.6819 -
root_mean_squared_error: 0.8254 - val_loss: 0.8639 -
val_root_mean_squared_error: 0.9295
Epoch 45/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.7053 -
root_mean_squared_error: 0.8384 - val_loss: 0.8694 -
val_root_mean_squared_error: 0.9324
Epoch 46/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6449 -
root_mean_squared_error: 0.8029 - val_loss: 0.8624 -
val_root_mean_squared_error: 0.9286
Epoch 47/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6626 -
root_mean_squared_error: 0.8134 - val_loss: 0.8602 -
val_root_mean_squared_error: 0.9275
Epoch 48/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6578 -
root_mean_squared_error: 0.8106 - val_loss: 0.8599 -
val_root_mean_squared_error: 0.9273
Epoch 49/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.7244 -
root_mean_squared_error: 0.8496 - val_loss: 0.8638 -
val_root_mean_squared_error: 0.9294
Epoch 50/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6400 -
root_mean_squared_error: 0.7996 - val_loss: 0.8658 -
val_root_mean_squared_error: 0.9305
Epoch 51/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6540 -
root_mean_squared_error: 0.8086 - val_loss: 0.8677 -
val_root_mean_squared_error: 0.9315
Epoch 52/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6780 -
root_mean_squared_error: 0.8232 - val_loss: 0.8672 -
val_root_mean_squared_error: 0.9313
Epoch 53/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6431 -
root_mean_squared_error: 0.8017 - val_loss: 0.8678 -
val_root_mean_squared_error: 0.9315
Epoch 54/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6036 -
root_mean_squared_error: 0.7761 - val_loss: 0.8633 -
val_root_mean_squared_error: 0.9291
```

```
Epoch 55/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6181 -
root_mean_squared_error: 0.7859 - val_loss: 0.8780 -
val_root_mean_squared_error: 0.9370
Epoch 56/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6718 -
root_mean_squared_error: 0.8194 - val_loss: 0.8735 -
val_root_mean_squared_error: 0.9346
Epoch 57/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6750 -
root_mean_squared_error: 0.8213 - val_loss: 0.8710 -
val_root_mean_squared_error: 0.9333
Epoch 58/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6887 -
root_mean_squared_error: 0.8292 - val_loss: 0.8731 -
val_root_mean_squared_error: 0.9344
Epoch 59/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6377 -
root_mean_squared_error: 0.7984 - val_loss: 0.8696 -
val_root_mean_squared_error: 0.9325
Epoch 60/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6216 -
root_mean_squared_error: 0.7883 - val_loss: 0.8870 -
val_root_mean_squared_error: 0.9418
Epoch 61/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6296 -
root_mean_squared_error: 0.7929 - val_loss: 0.8793 -
val_root_mean_squared_error: 0.9377
Epoch 62/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6100 -
root_mean_squared_error: 0.7807 - val_loss: 0.8767 -
val_root_mean_squared_error: 0.9363
Epoch 63/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6149 -
root_mean_squared_error: 0.7840 - val_loss: 0.8897 -
val_root_mean_squared_error: 0.9432
Epoch 64/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6106 -
root_mean_squared_error: 0.7811 - val_loss: 0.8779 -
val_root_mean_squared_error: 0.9369
Epoch 65/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6766 -
root_mean_squared_error: 0.8220 - val_loss: 0.8882 -
val_root_mean_squared_error: 0.9424
Epoch 66/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6453 -
root_mean_squared_error: 0.8031 - val_loss: 0.8890 -
val_root_mean_squared_error: 0.9429
Epoch 67/100
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5926 -
root_mean_squared_error: 0.7692 - val_loss: 0.8792 -
val_root_mean_squared_error: 0.9377
Epoch 68/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6150 -
root_mean_squared_error: 0.7835 - val_loss: 0.8751 -
val_root_mean_squared_error: 0.9355
Epoch 69/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6141 -
root_mean_squared_error: 0.7833 - val_loss: 0.8848 -
val_root_mean_squared_error: 0.9406
Epoch 70/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6014 -
root_mean_squared_error: 0.7752 - val_loss: 0.8916 -
val_root_mean_squared_error: 0.9443
Epoch 71/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6131 -
root_mean_squared_error: 0.7829 - val_loss: 0.8875 -
val_root_mean_squared_error: 0.9421
Epoch 72/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6243 -
root_mean_squared_error: 0.7899 - val_loss: 0.8924 -
val_root_mean_squared_error: 0.9447
Epoch 73/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5664 -
root_mean_squared_error: 0.7521 - val_loss: 0.8866 -
val_root_mean_squared_error: 0.9416
Epoch 74/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6039 -
root_mean_squared_error: 0.7767 - val_loss: 0.8856 -
val_root_mean_squared_error: 0.9411
Epoch 75/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6375 -
root_mean_squared_error: 0.7982 - val_loss: 0.8991 -
val_root_mean_squared_error: 0.9482
Epoch 76/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5776 -
root_mean_squared_error: 0.7597 - val_loss: 0.8944 -
val_root_mean_squared_error: 0.9457
Epoch 77/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5976 -
root_mean_squared_error: 0.7726 - val_loss: 0.8990 -
val_root_mean_squared_error: 0.9482
Epoch 78/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5550 -
root_mean_squared_error: 0.7448 - val_loss: 0.9030 -
val_root_mean_squared_error: 0.9503
Epoch 79/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5868 -
```

```
root_mean_squared_error: 0.7658 - val_loss: 0.8950 -
val_root_mean_squared_error: 0.9460
Epoch 80/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6097 -
root_mean_squared_error: 0.7801 - val_loss: 0.9046 -
val_root_mean_squared_error: 0.9511
Epoch 81/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5554 -
root_mean_squared_error: 0.7449 - val_loss: 0.9033 -
val_root_mean_squared_error: 0.9504
Epoch 82/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.5888 -
root_mean_squared_error: 0.7666 - val_loss: 0.8968 -
val_root_mean_squared_error: 0.9470
Epoch 83/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.6055 -
root_mean_squared_error: 0.7779 - val_loss: 0.9129 -
val_root_mean_squared_error: 0.9555
Epoch 84/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5671 -
root_mean_squared_error: 0.7528 - val_loss: 0.9040 -
val_root_mean_squared_error: 0.9508
Epoch 85/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.5571 -
root_mean_squared_error: 0.7461 - val_loss: 0.8937 -
val_root_mean_squared_error: 0.9454
Epoch 86/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5729 -
root_mean_squared_error: 0.7568 - val_loss: 0.9120 -
val_root_mean_squared_error: 0.9550
Epoch 87/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.6042 -
root_mean_squared_error: 0.7769 - val_loss: 0.9203 -
val_root_mean_squared_error: 0.9593
Epoch 88/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5571 -
root_mean_squared_error: 0.7463 - val_loss: 0.9120 -
val_root_mean_squared_error: 0.9550
Epoch 89/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5692 -
root_mean_squared_error: 0.7543 - val_loss: 0.9136 -
val_root_mean_squared_error: 0.9558
Epoch 90/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.5564 -
root_mean_squared_error: 0.7458 - val_loss: 0.9107 -
val_root_mean_squared_error: 0.9543
Epoch 91/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.5458 -
root_mean_squared_error: 0.7381 - val_loss: 0.9088 -
```

```
val_root_mean_squared_error: 0.9533
Epoch 92/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5445 -
root_mean_squared_error: 0.7376 - val_loss: 0.9251 -
val_root_mean_squared_error: 0.9618
Epoch 93/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5727 -
root_mean_squared_error: 0.7567 - val_loss: 0.9135 -
val_root_mean_squared_error: 0.9558
Epoch 94/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.5746 -
root_mean_squared_error: 0.7576 - val_loss: 0.9194 -
val_root_mean_squared_error: 0.9589
Epoch 95/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5068 -
root_mean_squared_error: 0.7099 - val_loss: 0.9276 -
val_root_mean_squared_error: 0.9631
Epoch 96/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5214 -
root_mean_squared_error: 0.7213 - val_loss: 0.9254 -
val_root_mean_squared_error: 0.9620
Epoch 97/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5602 -
root_mean_squared_error: 0.7482 - val_loss: 0.9140 -
val_root_mean_squared_error: 0.9560
Epoch 98/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5611 -
root_mean_squared_error: 0.7482 - val_loss: 0.9263 -
val_root_mean_squared_error: 0.9625
Epoch 99/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5845 -
root_mean_squared_error: 0.7642 - val_loss: 0.9183 -
val_root_mean_squared_error: 0.9583
Epoch 100/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5751 -
root_mean_squared_error: 0.7574 - val_loss: 0.9206 -
val_root_mean_squared_error: 0.9595
```

```python
# Посмотрим на потери модели
model_history.history
```

```
{'loss': [9.261905670166016,
  8.327838897705078,
  7.466700077056885,
  6.5802435874938965,
  5.62439489364624,
  4.583411693572998,
  3.5566720962524414,
  2.6491286754608154,
  1.9848363399505615,
```

```
  1.6090381145477295,
  1.4014173746109009,
  1.309525489807129,
  1.2357524633407593,
  1.1786445379257202,
  1.1326812505722046,
  1.0866512060165405,
  1.0482310056686401,
  1.0095247030258179,
  0.9753322005271912,
  0.9480133056640625,
  0.920687198638916,
  0.8970826268196106,
  0.8729844093322754,
  0.8521742224693298,
  0.8322960734367371,
  0.8157261610031128,
  0.8040559887886047,
  0.7891897559165955,
  0.7747093439102173,
  0.7621809244155884,
  0.7554405331611633,
  0.7438270449638367,
  0.7337678074836731,
  0.7286124229431152,
  0.7212194800376892,
  0.7161349058151245,
  0.7080916166305542,
  0.7058077454566956,
  0.7012231945991516,
  0.6958186030387878,
  0.6912018656730652,
  0.6886048316955566,
  0.6839228868484497,
  0.6787930130958557,
  0.6779356598854065,
  0.6721822619438171,
  0.6678324341773987,
  0.6644562482833862,
  0.6605589389801025,
  0.6595175266265869,
  0.6550401449203491,
  0.6529672741889954,
  0.6478049755096436,
  0.6470639109611511,
  0.6440303325653076,
  0.6399769186973572,
  0.6370531320571899,
  0.6349833011627197,
```

```
  0.6328302025794983,
  0.6287987232208252,
  0.6286587119102478,
  0.624546170234680‌2,
  0.6225168108940125,
  0.6180999279022217,
  0.6160736680030823,
  0.612827420234680‌2,
  0.6097449660301208,
  0.6121860146522522,
  0.607446551322937,
  0.6051326394081116,
  0.6027856469154358,
  0.5995573997497559,
  0.5964089632034302,
  0.5990961790084839,
  0.5961523056030273,
  0.5931375622749329,
  0.590187668800354,
  0.587803304195404,
  0.589832067489624,
  0.5833613276481628,
  0.5821890234947205,
  0.5804935693740845,
  0.5772608518600464,
  0.5763463377952576,
  0.582434535026550‌3,
  0.5720189213752747,
  0.5750992894172668,
  0.571302056312561,
  0.5653330683708191,
  0.564086377620697,
  0.5641186833381653,
  0.5587901473045349,
  0.557967483997345,
  0.5568670034408569,
  0.553088366985321,
  0.5546130537986755,
  0.5562941431999207,
  0.5525319576263428,
  0.5477803945541382,
  0.5452204942703247],
 'root_mean_squared_error': [3.043337821960449,
  2.8857994079589844,
  2.7325263023376465,
  2.5651986598968506,
  2.3715806007385254,
  2.14089035987854,
  1.8859140872955322,
```

```
    1.6276143789291382,
    1.4088422060012817,
    1.2684786319732666,
    1.1838147640228271,
    1.1443450450897217,
    1.1116440296173096,
    1.0856540203094482,
    1.0642750263214111,
    1.0424256324768066,
    1.0238314867019653,
    1.0047510862350464,
    0.9875890612602234,
    0.9736597537994385,
    0.9595244526863098,
    0.9471444487571716,
    0.9343363642692566,
    0.9231328368186951,
    0.9123026132583618,
    0.9031755924224854,
    0.8966916799545288,
    0.8883635401725769,
    0.880175769329071,
    0.8730297088623047,
    0.8691608309745789,
    0.8624540567398071,
    0.8566024899482727,
    0.8535879850387573,
    0.8492464423179626,
    0.8462475538253784,
    0.8414818048477173,
    0.8401236534118652,
    0.8373907208442688,
    0.834157407283783,
    0.8313854932785034,
    0.8298221826553345,
    0.8269963264465332,
    0.8238889575004578,
    0.823368489742279,
    0.8198671936988831,
    0.8172101378440857,
    0.8151418566703796,
    0.8127477765083313,
    0.8121068477630615,
    0.8093454837799072,
    0.8080639243125916,
    0.8048633337020874,
    0.8044028282165527,
    0.8025150299072266,
    0.7999855875968933,
```

```
  0.7981560826301575,
  0.7968583703041077,
  0.7955062389373779,
  0.7929682731628418,
  0.7928799986839294,
  0.7902823090553284,
  0.7889973521232605,
  0.786193311214447,
  0.7849035859107971,
  0.7828329205513,
  0.7808616757392883,
  0.782423198223114,
  0.7793885469436646,
  0.7779027223587036,
  0.7763926982879639,
  0.7743109464645386,
  0.7722752094268799,
  0.7740130424499512,
  0.7721089720726013,
  0.7701542377471924,
  0.7682367563247681,
  0.7666833400726318,
  0.7680052518844604,
  0.763780951499939,
  0.7630131244659424,
  0.7619012594223022,
  0.7597768306732178,
  0.7591747641563416,
  0.7631739974021912,
  0.7563193440437317,
  0.7583529949188232,
  0.7558452486991882,
  0.7518863677978516,
  0.7510568499565125,
  0.7510783672332764,
  0.7475226521492004,
  0.7469722032546997,
  0.746235191822052,
  0.7436991333961487,
  0.7447234988212585,
  0.7458512783050537,
  0.7433249354362488,
  0.7401219010353088,
  0.7383904457092285],
 'val_loss': [8.405193328857422,
  7.54771614074707,
  6.712080955505371,
  5.806857585906982,
  4.80334329605102 5,
```

```
  3.772233486175537,
  2.82258677482605,
  2.0711276531219482,
  1.6015647649765015,
  1.3546206951141357,
  1.255607008934021,
  1.193969964981079,
  1.1527286767959595,
  1.1204265356063843,
  1.0892513990402222,
  1.067193627357483,
  1.0386165380477905,
  1.0099133253097534,
  0.989105224609375,
  0.9673846364021301,
  0.947060227394104,
  0.9333518743515015,
  0.9215692281723022,
  0.9125630259513855,
  0.8993271589279175,
  0.8898035883903503,
  0.8869578838348389,
  0.8824961185455322,
  0.8775020241737366,
  0.8693603873252869,
  0.8702656030654907,
  0.8638948202133179,
  0.8704119324684143,
  0.8636702299118042,
  0.8650457859039307,
  0.8660791516304016,
  0.860426664352417,
  0.853843092918396,
  0.8590659499168396,
  0.856156051158905,
  0.8556830883026123,
  0.860825777053833,
  0.85867840051651,
  0.8638899922370911,
  0.8693864345550537,
  0.8623682856559753,
  0.8601869940757751,
  0.8598901629447937,
  0.8638142347335815,
  0.8657808303833008,
  0.8676807284355164,
  0.8672389984130859,
  0.8677518963813782,
  0.8632642030715942,
```

```
  0.8780477046966553,
  0.8734531998634338,
  0.8710103034973145,
  0.8731124401092529,
  0.8696308135986328,
  0.886982619762420 7,
  0.8792617917060852,
  0.8766781091690063,
  0.8896731734275818,
  0.8778603076934814,
  0.8881784081459045,
  0.8889865279197693,
  0.8791942000389099,
  0.8751035928726196,
  0.8848183155059814,
  0.8916499018669128,
  0.8875445127487183,
  0.8924211263656616,
  0.8866345286369324,
  0.8856092095375061,
  0.8990939855575562,
  0.8943877816200256,
  0.898996889591217,
  0.9030351042747498,
  0.8949810862541199,
  0.9046469926834106,
  0.9032623767852783,
  0.8968498110771179,
  0.91294264793396,
  0.9039939641952515,
  0.8936936855316162,
  0.9120066165924072,
  0.920314610004425,
  0.9119778871536255,
  0.9135839343070984,
  0.910736620426178,
  0.9088135957717896,
  0.9251121282577515,
  0.9135187864303589,
  0.9194415211677551,
  0.9275515675544739,
  0.9254248738288879,
  0.9139906764030457,
  0.9263414740562439,
  0.9183180928230286,
  0.9205638766288757],
 'val_root_mean_squared_error': [2.8991711139678955,
  2.7473106384277344,
  2.590768337249756,
```

```
  2.4097421169281006,
  2.19165301322937,
  1.9422239065170288,
  1.6800556182861328,
  1.4391412734985352,
  1.2655293941497803,
  1.163881778717041,
  1.1205387115478516,
  1.0926892757415771,
  1.0736520290374756,
  1.0585020780563354,
  1.0436720848083496,
  1.0330506563186646,
  1.0191253423690796,
  1.0049444437026978,
  0.9945377111434937,
  0.9835571050643921,
  0.9731702208518982,
  0.9661014080047607,
  0.9599839448928833,
  0.955281674861908,
  0.9483286142349243,
  0.9432939887046814,
  0.9417844414710999,
  0.9394126534461975,
  0.9367507696151733,
  0.9323949813842773,
  0.9328802824020386,
  0.929459452629089 4,
  0.932958722114563,
  0.9293385744094849,
  0.930078387260437,
  0.9306337237358093,
  0.927591860294342,
  0.9240363240242004,
  0.926858127117157,
  0.9252870082855225,
  0.9250314235687256,
  0.9278069734573364,
  0.9266490340232849,
  0.9294568300247192,
  0.9324089288711548,
  0.9286378622055054,
  0.927462637424469,
  0.9273025989532471,
  0.9294160604476929,
  0.9304734468460083,
  0.9314938187599182,
  0.9312566518783569,
```

```
  0.9315320253372192,
  0.9291201233863831,
  0.9370419979095459,
  0.934587180614714,
  0.9332793354988098,
  0.9344048500061035,
  0.9325399994850159,
  0.9417975544929504,
  0.9376896023750305,
  0.936310887336731,
  0.9432249069213867,
  0.9369419813156128,
  0.942432165145874,
  0.9428608417510986,
  0.9376535415649414,
  0.9354697465896606,
  0.9406478404998779,
  0.9442721605300903,
  0.9420958161354065,
  0.9446804523468018,
  0.941612720489502,
  0.9410681128501892,
  0.9482056498527527,
  0.9457207918167114,
  0.9481544494628906,
  0.9502816200256348,
  0.9460343718528748,
  0.9511293172836304,
  0.9504011869430542,
  0.9470215439796448,
  0.9554803371429443,
  0.9507859945297241,
  0.9453537464141846,
  0.9549903869628906,
  0.9593303203582764,
  0.9549753069877625,
  0.955815851688385,
  0.9543251991271973,
  0.9533171653747559,
  0.9618275165557861,
  0.9557817578315735,
  0.9588751196861267,
  0.9630947709083557,
  0.9619900584220886,
  0.9560285806655884,
  0.9624663591384888,
  0.9582891464233398,
  0.9594601988792419]}
```

```
#Мы оцениваем модель на тестовых данных с помощью evaluate(). Оценка
MSE
model_smn.evaluate(X_test_smn, y_test_smn)
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.0468 -
root_mean_squared_error: 1.0228

[1.0267751216888428, 1.0132991075515747]
```

```
# Осуществляем прогноз и выводим метрики
y_pred_model = model_smn.predict(X_test_smn)

print("Значения метрик:")
print("Корень из среднеквадратичной ошибки:",
np.sqrt(metrics.mean_squared_error(y_test_smn, y_pred_model)))
print("MSE:", metrics.mean_squared_error(y_test_smn, y_pred_model))
print("MAE:", metrics.mean_absolute_error(y_test_smn, y_pred_model))
print("R2", metrics.r2_score(y_test_smn, y_pred_model))
print("MAPE:", metrics.mean_absolute_percentage_error(y_test_smn,
y_pred_model))
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
Значения метрик:
Корень из среднеквадратичной ошибки: 1.0132991274489693
MSE: 1.0267751216888428
MAE: 0.8330175876617432
R2 -0.1282951831817627
MAPE: 0.3753160536289215
```

Тестируем вручную прогнозную способность нейросети на первой строке датафрейма

```
j = np.array([[2030.0,      738.736842,      50.00,      210.0,      70.0,
      3000.0,    4.0,  60.0, 1,    0,    52.250000, 167.750000]])
print(model_smn.predict(j))
```

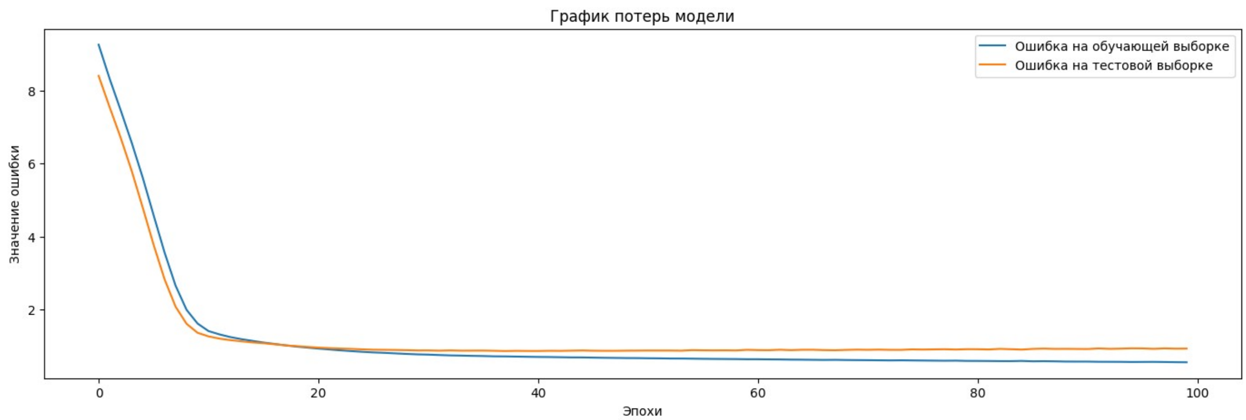
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
[[2.7351878]]
```

Правильное значение должно было быть: 1.857143

Визуализируем результаты работы нейросети

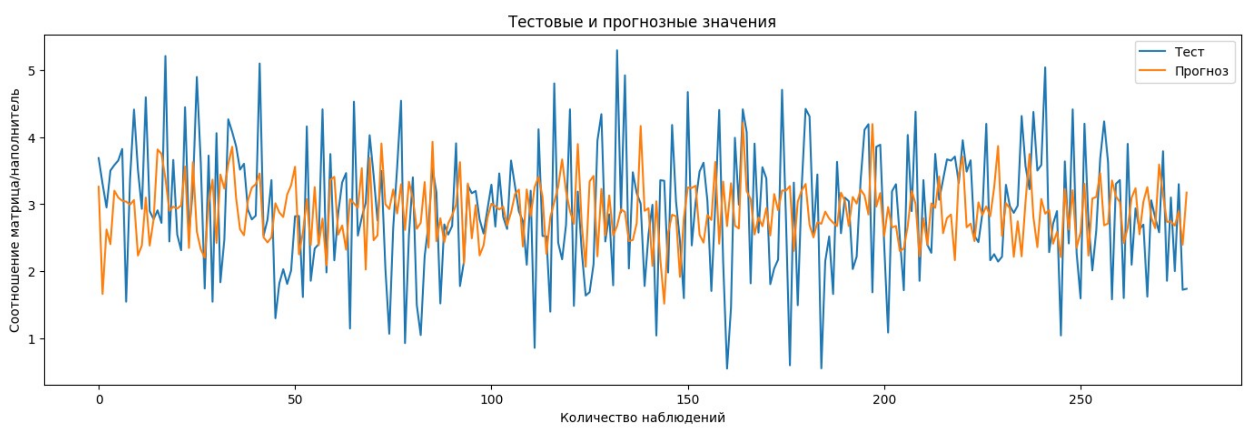
```
# Строим график потерь модели
plt.figure(figsize=(17, 5))
plt.plot(model_history.history["loss"], label="ошибка на обучающей
выборке")
plt.plot(model_history.history["val_loss"], label="ошибка на тестовой
выборке")
plt.title("График потерь модели")
```

```
plt.ylabel("Значение ошибки")
plt.xlabel("Эпохи")
plt.legend(["Ошибка на обучающей выборке", "Ошибка на тестовой выборке"], loc="best")
plt.show()
```




```
# График расхождения тестовых и прогнозных значений
plt.figure(figsize=(17, 5))
plt.title(f"Тестовые и прогнозные значения")
plt.plot(y_test_smn.values, label="Тест")
plt.plot(model_smn.predict(X_test_smn.values), label="Прогноз")
plt.legend(loc="best")
plt.ylabel('Соотношение матрица/наполнитель')
plt.xlabel("Количество наблюдений")
plt.show()

9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

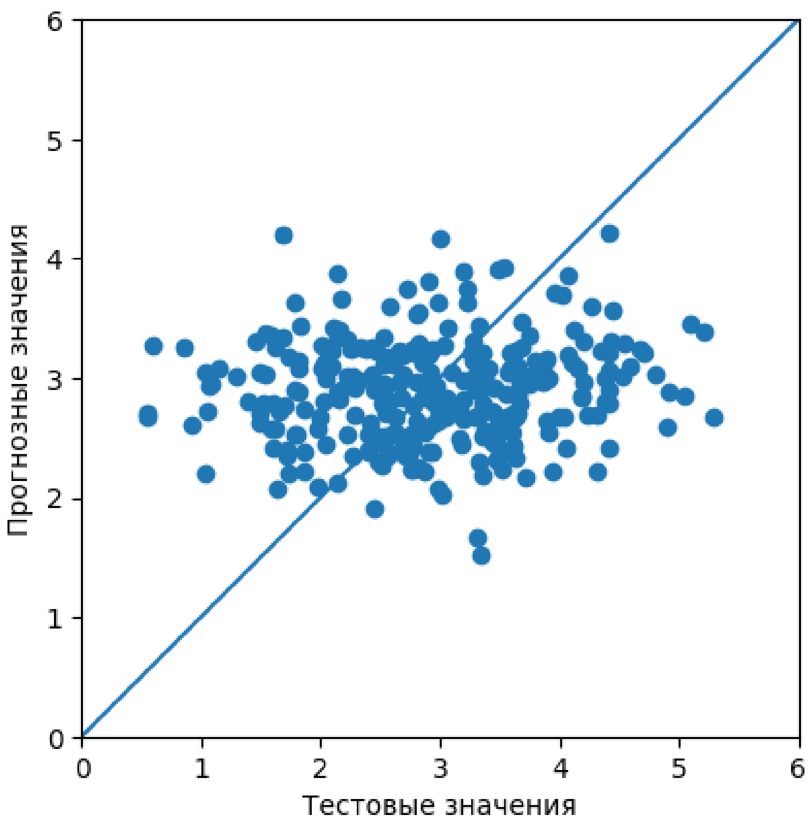



```
test_predictions = model_smn.predict(X_test_smn).flatten()

a = plt.axes(aspect="equal")
plt.scatter(y_test_smn, test_predictions)
```

```
plt.xlabel("Тестовые значения")
plt.ylabel("Прогнозные значения")
lims = [0, 6]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)

9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```



Сохраняем итоговую модель

```
with open('model_smn.pkl', 'wb') as file:
    pickle.dump(model_smn, file)
```

Вывод Метрики нейросети показали следующие результаты: Корень из среднеквадратичной ошибки: 1.0132991274489693 MSE: 1.0267751216888428 MAE: 0.8330175876617432 R2 -0.1282951831817627 MAPE: 0.3753160536289215

В целом прогнозная способность модели оказалась недостаточно точной. В целом MAE позволяет в какой-то мере понимать приблизительный результат соотношения матрицы-наполнителя. Необходимо отметить, что изменения количество слоев нейронов и увеличение количества эпох обучения в целом не дало намного лучшего результата. Поэтому нужно рассматривать либо иные модели нейронных сетей для решения данной задачи либо обратить внимание на предоставленные данные (с целью уточнения важности критериев, способов их сбора и возможно дополнительной их обработки, создания или вычленения новых признаков).