In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# Импортирование необходимых модулей и атрибутов
from sklearn import linear_model
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from matplotlib import pyplot
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_p
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegre
from sklearn.linear_model import LinearRegression, LogisticRegression, SG
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers

print(tf.__version__)
```

```
2.11.0
```

In [2]:
```
data_iqr_new = pd.read_excel(r'C:\Users\55944\Desktop\888/data_iqr_new.xl
data_iqr_new.drop(['Unnamed: 0'], axis=1,inplace=True)
```

In [3]:
```
data_iqr_new
```

Out[3]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	По
0	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	
1	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	

2	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385
3	2.767918	2000.000000	748.000000	111.860000	22.267857	284.615385
4	2.569620	1910.000000	807.000000	111.860000	22.267857	284.615385
...	...	...	...	...	...	...
915	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576
916	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401
917	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047
918	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840
919	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708

920 rows × 13 columns

In [4]:

```
# посмотрим с каким типом переменных нам предстоит работать
# для этого есть метод .info()
data_iqr_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 13 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Соотношение матрица-наполнитель       920 non-null    float64
 1   Плотность, кг/м3                      920 non-null    float64
 2   модуль упругости, ГПа                 920 non-null    float64
 3   Количество отвердителя, м.%           920 non-null    float64
 4   Содержание эпоксидных групп,%_2       920 non-null    float64
 5   Температура вспышки, С_2              920 non-null    float64
 6   Поверхностная плотность, г/м2         920 non-null    float64
 7   Модуль упругости при растяжении, ГПа  920 non-null    float64
 8   Прочность при растяжении, МПа         920 non-null    float64
 9   Потребление смолы, г/м2               920 non-null    float64
 10  Угол нашивки                          920 non-null    int64  
 11  Шаг нашивки                           920 non-null    float64
 12  Плотность нашивки                     920 non-null    float64
dtypes: float64(12), int64(1)
memory usage: 93.6 KB
```

In [5]:

```
# посмотрим на основные статистические показатели (summary statistics)
# с помощью метода .describe()
data_iqr_new.describe()
```

Out[5]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2
count	920.000000	920.000000	920.000000	920.000000	920.000000	920.000000
mean	2.926372	1974.155232	735.557925	111.153473	22.197906	286.208267
std	0.895102	71.066017	327.740846	26.778743	2.395341	39.453365
min	0.547391	1784.482245	2.436909	38.668500	15.695894	179.374391
25%	2.319283	1923.443748	498.438068	92.781590	20.555086	259.218101
50%	2.907832	1977.321002	734.464332	111.183627	22.166307	286.220763
75%	3.548025	2020.158764	956.411813	130.164272	23.955936	313.029126

max	5.314144	2161.565216	1628.000000	181.828448	28.955094	386.067992

In [6]:
```
# проверим, есть ли пропущенные значения

data_iqr_new.isnull().sum()
```

Out[6]:
```
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 [7]:
```
# посчитаем коэффициент корреляции для всего датафрейма и округлим значен
# получается корреляционная матрица
corr_matrix=data_iqr_new.corr()
corr_matrix
```

Out[7]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Темпе всп
Соотношение матрица-наполнитель	1.000000	0.009873	0.050317	0.002223	0.020305	-0.(
Плотность, кг/м3	0.009873	1.000000	-0.000982	-0.049445	0.005589	-0.(
модуль упругости, ГПа	0.050317	-0.000982	1.000000	0.045107	-0.002341	0.(
Количество отвердителя, м.%	0.002223	-0.049445	0.045107	1.000000	0.011945	0.(
Содержание эпоксидных групп,%_2	0.020305	0.005589	-0.002341	0.011945	1.000000	-0.(
Температура вспышки, С_2	-0.010702	-0.020804	0.038188	0.070592	-0.025338	1.(
Поверхностная плотность, г/м2	0.012439	0.062258	-0.006213	0.038464	-0.015443	0.(
Модуль упругости при растяжении, ГПа	-0.029204	-0.012335	0.018457	-0.055980	0.051470	0.(
Прочность при растяжении, МПа	0.019667	-0.081340	0.029136	-0.065738	-0.013156	-0.(
Потребление смолы, г/м2	0.076048	-0.009816	0.006692	-0.014363	0.010113	0.(
Угол нашивки	-0.030406	-0.053618	-0.029789	0.033480	0.035715	0.(

Шаг нашивки	0.043237	-0.050780	0.011725	-0.018336	0.009404	0.(
Плотность нашивки	0.045075	0.088828	0.078131	0.008752	-0.036162	-0.(

In [8]:
```
data_iqr_new.columns
```

Out[8]:
```
Index(['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',
       'Угол нашивки', 'Шаг нашивки', 'Плотность нашивки'],
      dtype='object')
```

In [9]:
```
# отберем признаки и поместим их в переменную X
x=data_iqr_new[['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,
       'Прочность при растяжении, МПа', 'Потребление смолы, г/м2',
       'Угол нашивки', 'Шаг нашивки', 'Плотность нашивки']]
```

In [10]:
```
# целевую переменную поместим в переменную y
y=data_iqr_new[['Соотношение матрица-наполнитель']]
```

In [11]:
```
print(type(x), type(y))
```

```
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.frame.DataFram
e'>
```

In [12]:
```
# разобьем данные на обучающую и тестовую выборку
# размер тестовой выборки составит 30%
# также зададим точку отсчета для воспроизводимости
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                     test_size = 0.3,
                                                     random_state = 42)
```

In [13]:
```
# посмотрим на новую размерность обучающей
print(x_train.shape, y_train.shape)

# и тестовой выборки
print(x_test.shape, y_test.shape)
```

```
(644, 12) (644, 1)
(276, 12) (276, 1)
```

In [14]:
```
normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(np.array(x_train))
def build_and_compile_model(norm):
  model = keras.Sequential([
      norm,
      layers.Dense(64, activation='relu'),
      layers.Dense(64, activation='relu'),
      layers.Dense(1)
  ])

  model.compile(loss='mean_squared_error',
                optimizer=tf.keras.optimizers.Adam(0.001),metrics=['mae']
  return model
```

```
model = build_and_compile_model(normalizer)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 normalization (Normalizatio  (None, 12)               25        
 n)                                                              
                                                                 
 dense (Dense)               (None, 64)                832       
                                                                 
 dense_1 (Dense)             (None, 64)                4160      
                                                                 
 dense_2 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 5,082
Trainable params: 5,057
Non-trainable params: 25
_________________________________________________________________
```

In [15]:

```
%%time
history = model.fit(
    x_train,
    y_train,
    validation_split=0.2,
    verbose=1, epochs=100)
```

```
Epoch 1/100
17/17 [==============================] - 24s 13ms/step - loss: 6.9719 - m
ae: 2.4424 - val_loss: 3.2553 - val_mae: 1.6097
Epoch 2/100
17/17 [==============================] - 0s 2ms/step - loss: 2.3083 - ma
e: 1.2482 - val_loss: 1.0167 - val_mae: 0.8153
Epoch 3/100
17/17 [==============================] - 0s 3ms/step - loss: 1.2909 - ma
e: 0.9181 - val_loss: 1.0163 - val_mae: 0.8129
Epoch 4/100
17/17 [==============================] - 0s 3ms/step - loss: 1.1627 - ma
e: 0.8751 - val_loss: 0.9344 - val_mae: 0.7802
Epoch 5/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0888 - ma
e: 0.8486 - val_loss: 0.9283 - val_mae: 0.7747
Epoch 6/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0517 - ma
e: 0.8342 - val_loss: 0.9329 - val_mae: 0.7776
Epoch 7/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0217 - ma
e: 0.8227 - val_loss: 0.9419 - val_mae: 0.7837
Epoch 8/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0031 - ma
e: 0.8169 - val_loss: 0.9391 - val_mae: 0.7832
Epoch 9/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9730 - ma
e: 0.8044 - val_loss: 0.9323 - val_mae: 0.7797
Epoch 10/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9555 - ma
e: 0.7970 - val_loss: 0.9460 - val_mae: 0.7929
Epoch 11/100
17/17 [==============================] - 0s 2ms/step - loss: 0.9285 - ma
e: 0.7841 - val_loss: 0.9251 - val_mae: 0.7815
Epoch 12/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8997 - ma
```

```
e: 0.7744 - val_loss: 0.9318 - val_mae: 0.7893
Epoch 13/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8844 - ma
e: 0.7679 - val_loss: 0.9325 - val_mae: 0.7893
Epoch 14/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8690 - ma
e: 0.7623 - val_loss: 0.9415 - val_mae: 0.7941
Epoch 15/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8452 - ma
e: 0.7521 - val_loss: 0.9300 - val_mae: 0.7898
Epoch 16/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8278 - ma
e: 0.7458 - val_loss: 0.9413 - val_mae: 0.8026
Epoch 17/100
17/17 [==============================] - 0s 2ms/step - loss: 0.8246 - ma
e: 0.7456 - val_loss: 0.9230 - val_mae: 0.7929
Epoch 18/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7976 - ma
e: 0.7314 - val_loss: 0.9318 - val_mae: 0.7962
Epoch 19/100
17/17 [==============================] - 0s 2ms/step - loss: 0.7801 - ma
e: 0.7253 - val_loss: 0.9302 - val_mae: 0.8009
Epoch 20/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7791 - ma
e: 0.7242 - val_loss: 0.9269 - val_mae: 0.7994
Epoch 21/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7473 - ma
e: 0.7081 - val_loss: 0.9652 - val_mae: 0.8137
Epoch 22/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7359 - ma
e: 0.6987 - val_loss: 0.9611 - val_mae: 0.8112
Epoch 23/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7137 - ma
e: 0.6901 - val_loss: 0.9599 - val_mae: 0.8119
Epoch 24/100
17/17 [==============================] - 0s 3ms/step - loss: 0.7039 - ma
e: 0.6864 - val_loss: 0.9456 - val_mae: 0.8049
Epoch 25/100
17/17 [==============================] - 0s 4ms/step - loss: 0.6925 - ma
e: 0.6811 - val_loss: 0.9419 - val_mae: 0.8048
Epoch 26/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6756 - ma
e: 0.6698 - val_loss: 0.9544 - val_mae: 0.8071
Epoch 27/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6677 - ma
e: 0.6649 - val_loss: 0.9341 - val_mae: 0.8003
Epoch 28/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6420 - ma
e: 0.6519 - val_loss: 0.9430 - val_mae: 0.8036
Epoch 29/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6318 - ma
e: 0.6476 - val_loss: 0.9541 - val_mae: 0.8079
Epoch 30/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6230 - ma
e: 0.6437 - val_loss: 0.9481 - val_mae: 0.8048
Epoch 31/100
17/17 [==============================] - 0s 3ms/step - loss: 0.6052 - ma
e: 0.6345 - val_loss: 0.9671 - val_mae: 0.8068
Epoch 32/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5877 - ma
e: 0.6204 - val_loss: 1.0044 - val_mae: 0.8261
Epoch 33/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5765 - ma
e: 0.6126 - val_loss: 0.9810 - val_mae: 0.8116
```

```
Epoch 34/100
17/17 [==============================] - 0s 4ms/step - loss: 0.5701 - ma
e: 0.6096 - val_loss: 1.0027 - val_mae: 0.8228
Epoch 35/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5706 - ma
e: 0.6124 - val_loss: 1.0143 - val_mae: 0.8216
Epoch 36/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5392 - ma
e: 0.5935 - val_loss: 1.0354 - val_mae: 0.8340
Epoch 37/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5233 - ma
e: 0.5840 - val_loss: 0.9891 - val_mae: 0.8143
Epoch 38/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5235 - ma
e: 0.5825 - val_loss: 0.9821 - val_mae: 0.8098
Epoch 39/100
17/17 [==============================] - 0s 3ms/step - loss: 0.5015 - ma
e: 0.5709 - val_loss: 1.0160 - val_mae: 0.8231
Epoch 40/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4877 - ma
e: 0.5643 - val_loss: 0.9993 - val_mae: 0.8198
Epoch 41/100
17/17 [==============================] - 0s 2ms/step - loss: 0.4793 - ma
e: 0.5563 - val_loss: 1.0154 - val_mae: 0.8285
Epoch 42/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4840 - ma
e: 0.5610 - val_loss: 1.0524 - val_mae: 0.8373
Epoch 43/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4629 - ma
e: 0.5443 - val_loss: 1.0198 - val_mae: 0.8267
Epoch 44/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4513 - ma
e: 0.5362 - val_loss: 1.0345 - val_mae: 0.8310
Epoch 45/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4507 - ma
e: 0.5395 - val_loss: 1.0623 - val_mae: 0.8344
Epoch 46/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4382 - ma
e: 0.5328 - val_loss: 1.0518 - val_mae: 0.8327
Epoch 47/100
17/17 [==============================] - 0s 2ms/step - loss: 0.4206 - ma
e: 0.5153 - val_loss: 1.0409 - val_mae: 0.8345
Epoch 48/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4125 - ma
e: 0.5104 - val_loss: 1.0557 - val_mae: 0.8348
Epoch 49/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4060 - ma
e: 0.5107 - val_loss: 1.0744 - val_mae: 0.8404
Epoch 50/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3916 - ma
e: 0.4980 - val_loss: 1.0709 - val_mae: 0.8487
Epoch 51/100
17/17 [==============================] - 0s 4ms/step - loss: 0.3897 - ma
e: 0.4946 - val_loss: 1.0756 - val_mae: 0.8476
Epoch 52/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3786 - ma
e: 0.4910 - val_loss: 1.0783 - val_mae: 0.8393
Epoch 53/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3690 - ma
e: 0.4840 - val_loss: 1.0791 - val_mae: 0.8407
Epoch 54/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3718 - ma
e: 0.4861 - val_loss: 1.0777 - val_mae: 0.8521
Epoch 55/100
```

```
17/17 [==============================] - 0s 3ms/step - loss: 0.3694 - ma
e: 0.4905 - val_loss: 1.0607 - val_mae: 0.8282
Epoch 56/100
17/17 [==============================] - 0s 2ms/step - loss: 0.3519 - ma
e: 0.4682 - val_loss: 1.0852 - val_mae: 0.8528
Epoch 57/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3376 - ma
e: 0.4592 - val_loss: 1.0906 - val_mae: 0.8524
Epoch 58/100
17/17 [==============================] - 0s 6ms/step - loss: 0.3368 - ma
e: 0.4607 - val_loss: 1.0895 - val_mae: 0.8437
Epoch 59/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3336 - ma
e: 0.4559 - val_loss: 1.1038 - val_mae: 0.8688
Epoch 60/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3210 - ma
e: 0.4458 - val_loss: 1.1216 - val_mae: 0.8596
Epoch 61/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3277 - ma
e: 0.4516 - val_loss: 1.1365 - val_mae: 0.8637
Epoch 62/100
17/17 [==============================] - 0s 2ms/step - loss: 0.3169 - ma
e: 0.4404 - val_loss: 1.1061 - val_mae: 0.8646
Epoch 63/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2996 - ma
e: 0.4308 - val_loss: 1.0959 - val_mae: 0.8616
Epoch 64/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3048 - ma
e: 0.4356 - val_loss: 1.1143 - val_mae: 0.8597
Epoch 65/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2944 - ma
e: 0.4286 - val_loss: 1.1191 - val_mae: 0.8711
Epoch 66/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2849 - ma
e: 0.4209 - val_loss: 1.1167 - val_mae: 0.8731
Epoch 67/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2884 - ma
e: 0.4186 - val_loss: 1.1447 - val_mae: 0.8821
Epoch 68/100
17/17 [==============================] - 0s 2ms/step - loss: 0.2788 - ma
e: 0.4134 - val_loss: 1.1262 - val_mae: 0.8810
Epoch 69/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2823 - ma
e: 0.4218 - val_loss: 1.1317 - val_mae: 0.8787
Epoch 70/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2600 - ma
e: 0.4008 - val_loss: 1.1217 - val_mae: 0.8717
Epoch 71/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2649 - ma
e: 0.4046 - val_loss: 1.1479 - val_mae: 0.8740
Epoch 72/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2487 - ma
e: 0.3909 - val_loss: 1.1355 - val_mae: 0.8814
Epoch 73/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2458 - ma
e: 0.3855 - val_loss: 1.1611 - val_mae: 0.8897
Epoch 74/100
17/17 [==============================] - 0s 2ms/step - loss: 0.2426 - ma
e: 0.3797 - val_loss: 1.1893 - val_mae: 0.9106
Epoch 75/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2412 - ma
e: 0.3807 - val_loss: 1.1543 - val_mae: 0.8916
Epoch 76/100
17/17 [==============================] - 0s 2ms/step - loss: 0.2354 - ma
```

```
e: 0.3795 - val_loss: 1.1719 - val_mae: 0.9024
Epoch 77/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2259 - ma
e: 0.3707 - val_loss: 1.1859 - val_mae: 0.9040
Epoch 78/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2333 - ma
e: 0.3757 - val_loss: 1.1842 - val_mae: 0.8903
Epoch 79/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2158 - ma
e: 0.3600 - val_loss: 1.1899 - val_mae: 0.9032
Epoch 80/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2076 - ma
e: 0.3497 - val_loss: 1.2103 - val_mae: 0.9098
Epoch 81/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2020 - ma
e: 0.3506 - val_loss: 1.1931 - val_mae: 0.9071
Epoch 82/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1975 - ma
e: 0.3417 - val_loss: 1.1960 - val_mae: 0.9115
Epoch 83/100
17/17 [==============================] - 0s 2ms/step - loss: 0.1989 - ma
e: 0.3471 - val_loss: 1.1977 - val_mae: 0.9144
Epoch 84/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1963 - ma
e: 0.3461 - val_loss: 1.2446 - val_mae: 0.9188
Epoch 85/100
17/17 [==============================] - 0s 2ms/step - loss: 0.1930 - ma
e: 0.3342 - val_loss: 1.2163 - val_mae: 0.9170
Epoch 86/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1814 - ma
e: 0.3291 - val_loss: 1.2235 - val_mae: 0.9203
Epoch 87/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1746 - ma
e: 0.3187 - val_loss: 1.2282 - val_mae: 0.9240
Epoch 88/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1712 - ma
e: 0.3185 - val_loss: 1.2115 - val_mae: 0.9169
Epoch 89/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1677 - ma
e: 0.3131 - val_loss: 1.2528 - val_mae: 0.9358
Epoch 90/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1619 - ma
e: 0.3026 - val_loss: 1.2298 - val_mae: 0.9272
Epoch 91/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1600 - ma
e: 0.3089 - val_loss: 1.2449 - val_mae: 0.9360
Epoch 92/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1538 - ma
e: 0.2953 - val_loss: 1.2647 - val_mae: 0.9409
Epoch 93/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1564 - ma
e: 0.2977 - val_loss: 1.2637 - val_mae: 0.9360
Epoch 94/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1492 - ma
e: 0.2952 - val_loss: 1.2491 - val_mae: 0.9353
Epoch 95/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1531 - ma
e: 0.2976 - val_loss: 1.2682 - val_mae: 0.9429
Epoch 96/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1460 - ma
e: 0.2853 - val_loss: 1.2855 - val_mae: 0.9514
Epoch 97/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1376 - ma
e: 0.2818 - val_loss: 1.2747 - val_mae: 0.9462
```

```
Epoch 98/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1337 - ma
e: 0.2752 - val_loss: 1.2734 - val_mae: 0.9437
Epoch 99/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1431 - ma
e: 0.2858 - val_loss: 1.2785 - val_mae: 0.9439
Epoch 100/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1395 - ma
e: 0.2840 - val_loss: 1.3110 - val_mae: 0.9551
Wall time: 29.1 s
```

In [16]:

```
mae=model.evaluate(x_test,y_test,verbose=1)
```

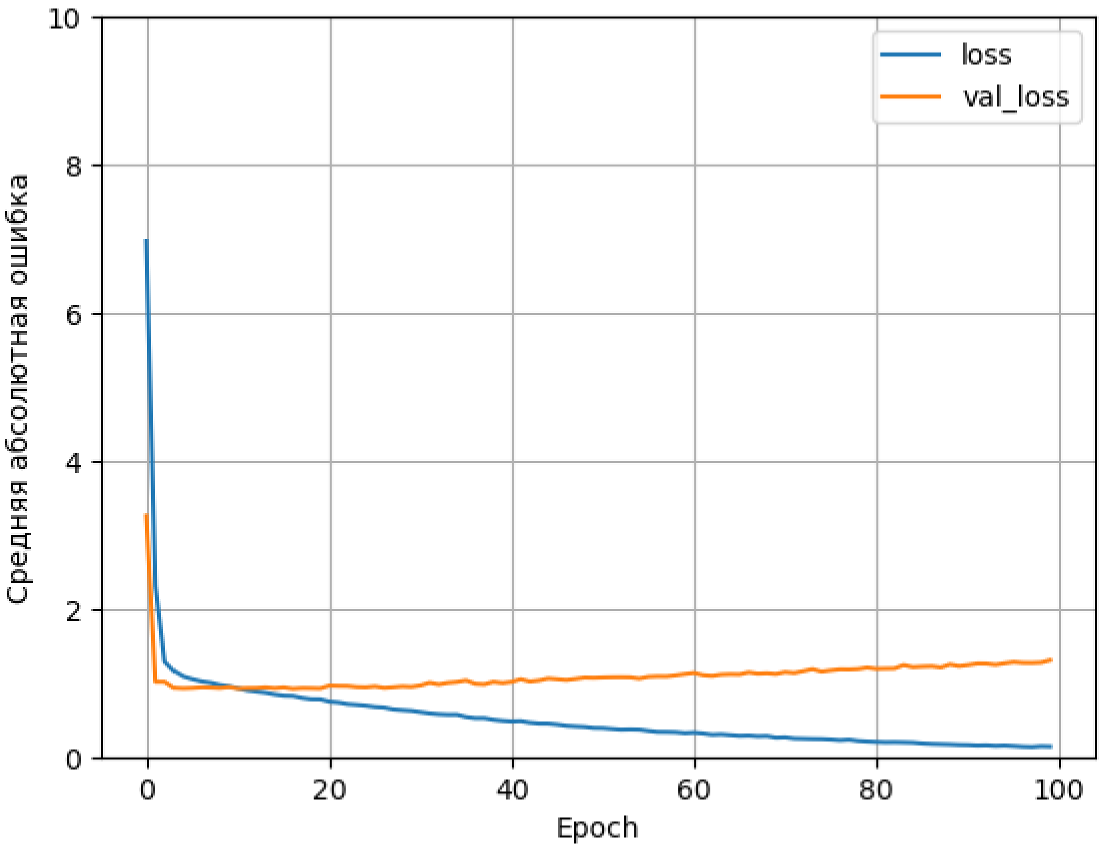
```
9/9 [==============================] - 0s 2ms/step - loss: 1.3124 - mae:
0.9175
```

In [17]:

```
def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 10])
  plt.xlabel('Epoch')
  plt.ylabel('Средняя абсолютная ошибка')
  plt.legend()
  plt.grid(True)
```

In [18]:

```
plot_loss(history)
```



In [19]:

```
y_pred_model = model.predict(x_test)

print('Model Results:')
print('Model_MAE: ', round(mean_absolute_error(y_test, y_pred_model)))
print('Model_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test,
print("Test score: {:.2f}".format(mean_squared_error(y_test, y_pred_model
```

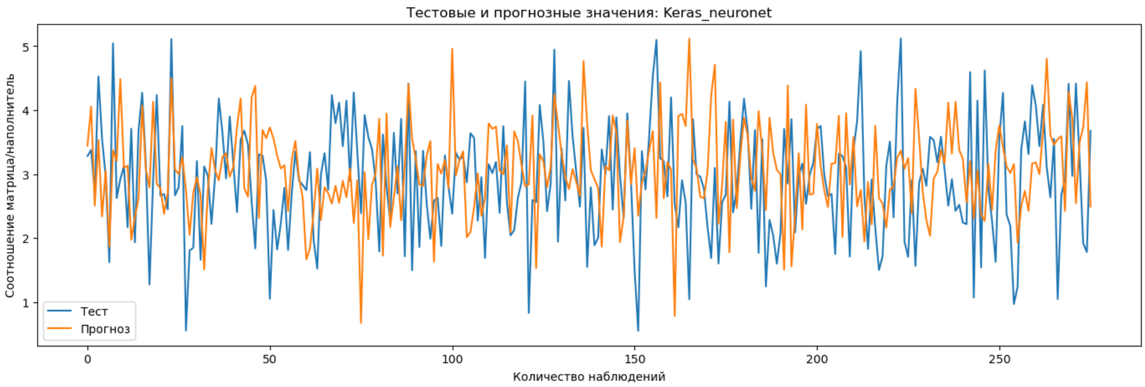
```
9/9 [==============================] - 0s 0s/step
```

```
Model Results:
Model_MAE:  1
Model_MAPE: 0.41
Test score: 1.31
```

In [20]:

```python
# Зададим функцию для визуализации факт/прогноз для результатов моделей
# Посмотрим на график результата работы модели
def actual_and_predicted_plot(orig, predict, var, model_name):
    plt.figure(figsize=(17,5))
    plt.title(f'Тестовые и прогнозные значения: {model_name}')
    plt.plot(orig, label = 'Тест')
    plt.plot(predict, label = 'Прогноз')
    plt.legend(loc = 'best')
    plt.ylabel(var)
    plt.xlabel('Количество наблюдений')
    plt.show()
actual_and_predicted_plot(y_test.values, model.predict(x_test.values), 'C
```

```
9/9 [==============================] - 0s 2ms/step
```

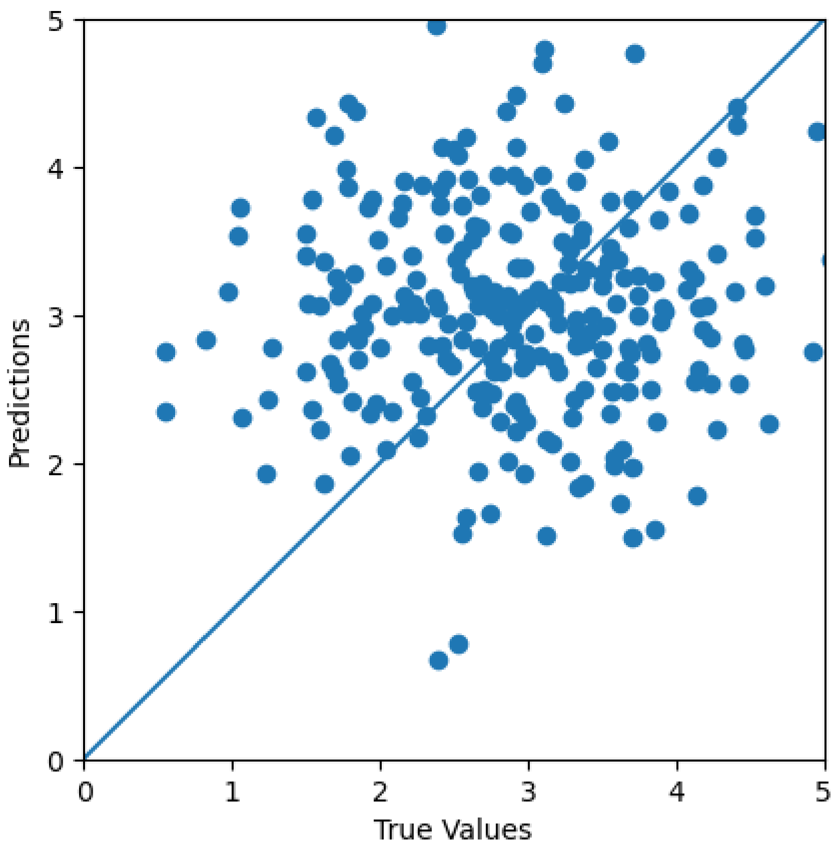


In [21]:

```python
test_predictions = model.predict(x_test).flatten()

a = plt.axes(aspect = 'equal')
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 5]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
9/9 [==============================] - 0s 2ms/step
```



In [22]:
```
pip install scikeras
```

```
Requirement already satisfied: scikeras in c:\users\55944\anaconda\lib\si
te-packages (0.10.0)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\55944\anac
onda\lib\site-packages (from scikeras) (1.0.2)
Requirement already satisfied: packaging>=0.21 in c:\users\55944\anaconda
\lib\site-packages (from scikeras) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\55944
\anaconda\lib\site-packages (from packaging>=0.21->scikeras) (3.0.9)
Requirement already satisfied: numpy>=1.14.6 in c:\users\55944\anaconda\l
ib\site-packages (from scikit-learn>=1.0.0->scikeras) (1.21.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\55944\ana
conda\lib\site-packages (from scikit-learn>=1.0.0->scikeras) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\55944\anaconda\li
b\site-packages (from scikit-learn>=1.0.0->scikeras) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\55944\anaconda\li
b\site-packages (from scikit-learn>=1.0.0->scikeras) (1.9.1)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:
```
from keras.wrappers.scikit_learn import KerasRegressor
import numpy as np
import tensorflow as tf
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [24]:
```
from tensorflow.keras.layers import Dropout
```

In [25]:
```
def create_model_1(lyrs=[32], act='softmax', optimizer='adam', dr=0.1):

    seed = 7
    np.random.seed(seed)
    tf.random.set_seed(seed)
```

```
    model_1 = Sequential()
    model_1.add(Dense(lyrs[0], input_dim=x_train.shape[1], activation=act
    for i in range(1,len(lyrs)):
        model_1.add(Dense(lyrs[i], activation=act))

    model_1.add(Dropout(dr))
    model_1.add(Dense(1))  # выходной слой

    model_1.compile(loss='mean_squared_error', optimizer='adam', metrics=

    return model_1
```

In [26]:
```
model_1 = KerasRegressor(build_fn=create_model_1, verbose=0)

batch_size = [10, 20, 30, 40, 50]
epochs = [10, 50, 100]

param_grid = dict(batch_size=batch_size, epochs=epochs)

grid = GridSearchCV(estimator=model_1, param_grid=param_grid, n_jobs=1,cv
grid_result = grid.fit(x_train, y_train)

#summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_pa
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
WARNING:tensorflow:5 out of the last 14 calls to <function Model.make_tes
t_function.<locals>.test_function at 0x0000015E74DE8B80> triggered tf.fun
ction retracing. Tracing is expensive and the excessive number of tracing
s could be due to (1) creating @tf.function repeatedly in a loop, (2) pas
sing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. Fo
r (2), @tf.function has reduce_retracing=True option that can avoid unnec
essary retracing. For (3), please refer to https://www.tensorflow.org/gui
de/function#controlling_retracing and https://www.tensorflow.org/api_doc
s/python/tf/function for  more details.
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_tes
t_function.<locals>.test_function at 0x0000015E72814F70> triggered tf.fun
ction retracing. Tracing is expensive and the excessive number of tracing
s could be due to (1) creating @tf.function repeatedly in a loop, (2) pas
sing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. Fo
r (2), @tf.function has reduce_retracing=True option that can avoid unnec
essary retracing. For (3), please refer to https://www.tensorflow.org/gui
de/function#controlling_retracing and https://www.tensorflow.org/api_doc
s/python/tf/function for  more details.
Best: -0.808694 using {'batch_size': 10, 'epochs': 100}
-3.535093 (0.521314) with: {'batch_size': 10, 'epochs': 10}
-0.821332 (0.153429) with: {'batch_size': 10, 'epochs': 50}
-0.808694 (0.160369) with: {'batch_size': 10, 'epochs': 100}
-5.811971 (1.550652) with: {'batch_size': 20, 'epochs': 10}
-1.264936 (0.368176) with: {'batch_size': 20, 'epochs': 50}
-0.823463 (0.156108) with: {'batch_size': 20, 'epochs': 100}
-6.063465 (0.749669) with: {'batch_size': 30, 'epochs': 10}
-2.107765 (0.460876) with: {'batch_size': 30, 'epochs': 50}
-0.884638 (0.170303) with: {'batch_size': 30, 'epochs': 100}
-6.888370 (1.225409) with: {'batch_size': 40, 'epochs': 10}
-3.311201 (0.985067) with: {'batch_size': 40, 'epochs': 50}
-1.228713 (0.412477) with: {'batch_size': 40, 'epochs': 100}
```

```
-7.086890 (1.509289) with: {'batch_size': 50, 'epochs': 10}
-3.150868 (0.657717) with: {'batch_size': 50, 'epochs': 50}
-1.566316 (0.340155) with: {'batch_size': 50, 'epochs': 100}
```

In [27]:

```
model_1 = KerasRegressor(build_fn=create_model_1, epochs=100, batch_size=

optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Nadam']
param_grid = dict(optimizer=optimizer)

grid = GridSearchCV(estimator=model_1, param_grid=param_grid, cv=10, verb
grid_result = grid.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 6 candidates, totalling 60 fits
[CV] END .......................................optimizer=SGD; total time=
6.2s
[CV] END .......................................optimizer=SGD; total time=
6.4s
[CV] END .......................................optimizer=SGD; total time=
6.2s
[CV] END .......................................optimizer=SGD; total time=
6.5s
[CV] END .......................................optimizer=SGD; total time=
5.8s
[CV] END .......................................optimizer=SGD; total time=
5.9s
[CV] END .......................................optimizer=SGD; total time=
6.1s
[CV] END .......................................optimizer=SGD; total time=
6.0s
[CV] END .......................................optimizer=SGD; total time=
5.9s
[CV] END .......................................optimizer=SGD; total time=
5.8s
[CV] END ...................................optimizer=RMSprop; total time=
6.3s
[CV] END ...................................optimizer=RMSprop; total time=
6.3s
[CV] END ...................................optimizer=RMSprop; total time=
6.2s
[CV] END ...................................optimizer=RMSprop; total time=
6.2s
[CV] END ...................................optimizer=RMSprop; total time=
5.9s
[CV] END ...................................optimizer=RMSprop; total time=
5.8s
[CV] END ...................................optimizer=RMSprop; total time=
6.5s
[CV] END ...................................optimizer=RMSprop; total time=
5.8s
[CV] END ...................................optimizer=RMSprop; total time=
6.3s
[CV] END ...................................optimizer=RMSprop; total time=
6.0s
[CV] END ...................................optimizer=Adagrad; total time=
6.2s
[CV] END ...................................optimizer=Adagrad; total time=
6.6s
[CV] END ...................................optimizer=Adagrad; total time=
6.2s
[CV] END ...................................optimizer=Adagrad; total time=
6.3s
[CV] END ...................................optimizer=Adagrad; total time=
5.9s
[CV] END ...................................optimizer=Adagrad; total time=
```

```
5.9s
[CV] END ..................................optimizer=Adagrad; total time=
6.1s
[CV] END ..................................optimizer=Adagrad; total time=
5.8s
[CV] END ..................................optimizer=Adagrad; total time=
5.8s
[CV] END ..................................optimizer=Adagrad; total time=
6.0s
[CV] END .................................optimizer=Adadelta; total time=
6.3s
[CV] END .................................optimizer=Adadelta; total time=
6.2s
[CV] END .................................optimizer=Adadelta; total time=
6.2s
[CV] END .................................optimizer=Adadelta; total time=
6.0s
[CV] END .................................optimizer=Adadelta; total time=
5.9s
[CV] END .................................optimizer=Adadelta; total time=
5.8s
[CV] END .................................optimizer=Adadelta; total time=
6.4s
[CV] END .................................optimizer=Adadelta; total time=
5.8s
[CV] END .................................optimizer=Adadelta; total time=
5.5s
[CV] END .................................optimizer=Adadelta; total time=
6.0s
[CV] END .....................................optimizer=Adam; total time=
6.2s
[CV] END .....................................optimizer=Adam; total time=
6.3s
[CV] END .....................................optimizer=Adam; total time=
6.6s
[CV] END .....................................optimizer=Adam; total time=
6.1s
[CV] END .....................................optimizer=Adam; total time=
6.0s
[CV] END .....................................optimizer=Adam; total time=
6.0s
[CV] END .....................................optimizer=Adam; total time=
6.0s
[CV] END .....................................optimizer=Adam; total time=
6.4s
[CV] END .....................................optimizer=Adam; total time=
5.9s
[CV] END .....................................optimizer=Adam; total time=
5.8s
[CV] END ....................................optimizer=Nadam; total time=
6.6s
[CV] END ....................................optimizer=Nadam; total time=
6.5s
[CV] END ....................................optimizer=Nadam; total time=
11.0s
[CV] END ....................................optimizer=Nadam; total time=
6.4s
[CV] END ....................................optimizer=Nadam; total time=
5.9s
[CV] END ....................................optimizer=Nadam; total time=
5.8s
[CV] END ....................................optimizer=Nadam; total time=
5.9s
[CV] END ....................................optimizer=Nadam; total time=
```

```
5.9s
[CV] END ...................................optimizer=Nadam; total time=
5.9s
[CV] END ...................................optimizer=Nadam; total time=
5.9s
```

In [28]:

```
# результаты
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_pa
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: -0.807970 using {'optimizer': 'Adagrad'}
-0.808197 (0.160460) with: {'optimizer': 'SGD'}
-0.809522 (0.160431) with: {'optimizer': 'RMSprop'}
-0.807970 (0.159451) with: {'optimizer': 'Adagrad'}
-0.808436 (0.159496) with: {'optimizer': 'Adadelta'}
-0.808260 (0.159257) with: {'optimizer': 'Adam'}
-0.808976 (0.159796) with: {'optimizer': 'Nadam'}
```

In [29]:

```
model_1 = KerasRegressor(build_fn=create_model_1, epochs=100, batch_size=

layers = [[8],[16, 2],[32, 8, 2],[12, 6, 1], [64, 64, 3], [128, 64, 16, 3
param_grid = dict(lyrs=layers)

grid = GridSearchCV(estimator=model_1, param_grid=param_grid, cv=10, verb
grid_result = grid.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 6 candidates, totalling 60 fits
[CV] END ...........................................lyrs=[8]; total time=
6.0s
[CV] END ...........................................lyrs=[8]; total time=
6.1s
[CV] END ...........................................lyrs=[8]; total time=
6.1s
[CV] END ...........................................lyrs=[8]; total time=
6.1s
[CV] END ...........................................lyrs=[8]; total time=
5.8s
[CV] END ...........................................lyrs=[8]; total time=
5.8s
[CV] END ...........................................lyrs=[8]; total time=
6.2s
[CV] END ...........................................lyrs=[8]; total time=
5.7s
[CV] END ...........................................lyrs=[8]; total time=
5.8s
[CV] END ...........................................lyrs=[8]; total time=
5.8s
[CV] END .......................................lyrs=[16, 2]; total time=
7.6s
[CV] END .......................................lyrs=[16, 2]; total time=
6.7s
[CV] END .......................................lyrs=[16, 2]; total time=
6.6s
[CV] END .......................................lyrs=[16, 2]; total time=
6.6s
[CV] END .......................................lyrs=[16, 2]; total time=
6.3s
[CV] END .......................................lyrs=[16, 2]; total time=
6.3s
[CV] END .......................................lyrs=[16, 2]; total time=
```

```
6.2s
[CV] END ......................................lyrs=[16, 2]; total time=
6.2s
[CV] END ......................................lyrs=[16, 2]; total time=
6.4s
[CV] END ......................................lyrs=[16, 2]; total time=
6.6s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
7.2s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
7.0s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
7.2s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
7.1s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.8s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.9s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.7s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.9s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.8s
[CV] END ...................................lyrs=[32, 8, 2]; total time=
6.8s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
7.6s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
7.0s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
7.0s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.9s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.8s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.7s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.8s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.6s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.7s
[CV] END ...................................lyrs=[12, 6, 1]; total time=
6.7s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.3s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.8s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.3s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.6s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.2s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.3s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.1s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.2s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
```

```
7.2s
[CV] END ..................................lyrs=[64, 64, 3]; total time=
7.3s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.1s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.1s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.5s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.7s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.3s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.2s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.0s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
8.0s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
7.9s
[CV] END .............................lyrs=[128, 64, 16, 3]; total time=
7.9s
```

In [30]:

```
# результаты
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_pa
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: -0.807429 using {'lyrs': [8]}
-0.807429 (0.160722) with: {'lyrs': [8]}
-0.808426 (0.158461) with: {'lyrs': [16, 2]}
-0.808542 (0.161404) with: {'lyrs': [32, 8, 2]}
-0.807575 (0.159281) with: {'lyrs': [12, 6, 1]}
-0.808248 (0.160392) with: {'lyrs': [64, 64, 3]}
-0.809074 (0.163147) with: {'lyrs': [128, 64, 16, 3]}
```

In [31]:

```
model_1 =  KerasRegressor(build_fn=create_model_1, epochs=100, batch_size

drops = [0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5]
param_grid = dict(dr=drops)

grid = GridSearchCV(estimator=model_1, param_grid=param_grid, cv=10, verb
grid_result = grid.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 7 candidates, totalling 70 fits
[CV] END ............................................dr=0.0; total time=
6.0s
[CV] END ............................................dr=0.0; total time=
5.8s
[CV] END ............................................dr=0.0; total time=
6.1s
[CV] END ............................................dr=0.0; total time=
6.0s
[CV] END ............................................dr=0.0; total time=
6.0s
[CV] END ............................................dr=0.0; total time=
5.7s
[CV] END ............................................dr=0.0; total time=
5.9s
[CV] END ............................................dr=0.0; total time=
```

```
5.8s
[CV] END ............................................dr=0.0; total time=
5.8s
[CV] END ............................................dr=0.0; total time=
5.8s
[CV] END ...........................................dr=0.01; total time=
6.2s
[CV] END ...........................................dr=0.01; total time=
6.2s
[CV] END ...........................................dr=0.01; total time=
6.2s
[CV] END ...........................................dr=0.01; total time=
6.2s
[CV] END ...........................................dr=0.01; total time=
5.9s
[CV] END ...........................................dr=0.01; total time=
5.8s
[CV] END ...........................................dr=0.01; total time=
5.9s
[CV] END ...........................................dr=0.01; total time=
5.9s
[CV] END ...........................................dr=0.01; total time=
5.8s
[CV] END ...........................................dr=0.01; total time=
6.2s
[CV] END ...........................................dr=0.05; total time=
6.1s
[CV] END ...........................................dr=0.05; total time=
6.2s
[CV] END ...........................................dr=0.05; total time=
6.1s
[CV] END ...........................................dr=0.05; total time=
6.2s
[CV] END ...........................................dr=0.05; total time=
5.9s
[CV] END ...........................................dr=0.05; total time=
5.9s
[CV] END ...........................................dr=0.05; total time=
5.9s
[CV] END ...........................................dr=0.05; total time=
5.9s
[CV] END ...........................................dr=0.05; total time=
5.8s
[CV] END ...........................................dr=0.05; total time=
5.9s
[CV] END ............................................dr=0.1; total time=
6.1s
[CV] END ............................................dr=0.1; total time=
6.1s
[CV] END ............................................dr=0.1; total time=
6.1s
[CV] END ............................................dr=0.1; total time=
6.2s
[CV] END ............................................dr=0.1; total time=
6.1s
[CV] END ............................................dr=0.1; total time=
5.9s
[CV] END ............................................dr=0.1; total time=
5.8s
[CV] END ............................................dr=0.1; total time=
5.9s
[CV] END ............................................dr=0.1; total time=
5.9s
[CV] END ............................................dr=0.1; total time=
```

```
5.9s
[CV] END ............................................dr=0.2; total time=
6.2s
[CV] END ............................................dr=0.2; total time=
6.2s
[CV] END ............................................dr=0.2; total time=
6.3s
[CV] END ............................................dr=0.2; total time=
6.1s
[CV] END ............................................dr=0.2; total time=
6.0s
[CV] END ............................................dr=0.2; total time=
5.9s
[CV] END ............................................dr=0.2; total time=
5.8s
[CV] END ............................................dr=0.2; total time=
5.9s
[CV] END ............................................dr=0.2; total time=
6.0s
[CV] END ............................................dr=0.2; total time=
6.1s
[CV] END ............................................dr=0.3; total time=
6.2s
[CV] END ............................................dr=0.3; total time=
6.1s
[CV] END ............................................dr=0.3; total time=
6.1s
[CV] END ............................................dr=0.3; total time=
6.2s
[CV] END ............................................dr=0.3; total time=
5.8s
[CV] END ............................................dr=0.3; total time=
5.9s
[CV] END ............................................dr=0.3; total time=
5.9s
[CV] END ............................................dr=0.3; total time=
5.8s
[CV] END ............................................dr=0.3; total time=
5.9s
[CV] END ............................................dr=0.3; total time=
5.9s
[CV] END ............................................dr=0.5; total time=
6.1s
[CV] END ............................................dr=0.5; total time=
6.3s
[CV] END ............................................dr=0.5; total time=
6.2s
[CV] END ............................................dr=0.5; total time=
6.2s
[CV] END ............................................dr=0.5; total time=
6.0s
[CV] END ............................................dr=0.5; total time=
6.1s
[CV] END ............................................dr=0.5; total time=
5.9s
[CV] END ............................................dr=0.5; total time=
5.8s
[CV] END ............................................dr=0.5; total time=
6.0s
[CV] END ............................................dr=0.5; total time=
6.1s
```

In [32]:

```
# результаты
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_pa
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: -0.804978 using {'dr': 0.01}
-0.809019 (0.164376) with: {'dr': 0.0}
-0.804978 (0.163927) with: {'dr': 0.01}
-0.805363 (0.161511) with: {'dr': 0.05}
-0.824838 (0.174830) with: {'dr': 0.1}
-0.810791 (0.158583) with: {'dr': 0.2}
-0.809039 (0.159315) with: {'dr': 0.3}
-0.810640 (0.159274) with: {'dr': 0.5}
```

In [45]:
```
# построение окончательной модели
model_1 = create_model_1(lyrs=[12,8,4], dr=0.01)

print(model_1.summary())
```

```
Model: "sequential_347"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_797 (Dense)           (None, 12)                156       
                                                                 
 dense_798 (Dense)           (None, 8)                 104       
                                                                 
 dense_799 (Dense)           (None, 4)                 36        
                                                                 
 dropout_346 (Dropout)       (None, 4)                 0         
                                                                 
 dense_800 (Dense)           (None, 1)                 5         
                                                                 
=================================================================
Total params: 301
Trainable params: 301
Non-trainable params: 0
_________________________________________________________________
None
```

In [46]:
```
# обучаем нейросеть, 80/20 CV
model_1_hist = model_1.fit(x_train,
    y_train,
    epochs = 100,
    verbose = 1,
    validation_split = 0.2)
```

```
Epoch 1/100
17/17 [==============================] - 1s 10ms/step - loss: 8.6813 - ma
e: 2.8024 - val_loss: 8.9237 - val_mae: 2.8663
Epoch 2/100
17/17 [==============================] - 0s 3ms/step - loss: 8.3863 - ma
e: 2.7501 - val_loss: 8.6337 - val_mae: 2.8153
Epoch 3/100
17/17 [==============================] - 0s 4ms/step - loss: 8.1075 - ma
e: 2.6994 - val_loss: 8.3523 - val_mae: 2.7648
Epoch 4/100
17/17 [==============================] - 0s 3ms/step - loss: 7.8538 - ma
e: 2.6509 - val_loss: 8.0760 - val_mae: 2.7144
Epoch 5/100
17/17 [==============================] - 0s 3ms/step - loss: 7.5735 - ma
e: 2.5980 - val_loss: 7.8098 - val_mae: 2.6649
Epoch 6/100
17/17 [==============================] - 0s 3ms/step - loss: 7.3255 - ma
```

```
e: 2.5496 - val_loss: 7.5459 - val_mae: 2.6149
Epoch 7/100
17/17 [==============================] - 0s 4ms/step - loss: 7.0666 - ma
e: 2.4981 - val_loss: 7.2905 - val_mae: 2.5656
Epoch 8/100
17/17 [==============================] - 0s 3ms/step - loss: 6.8369 - ma
e: 2.4518 - val_loss: 7.0409 - val_mae: 2.5165
Epoch 9/100
17/17 [==============================] - 0s 4ms/step - loss: 6.5890 - ma
e: 2.4008 - val_loss: 6.7920 - val_mae: 2.4666
Epoch 10/100
17/17 [==============================] - 0s 3ms/step - loss: 6.3583 - ma
e: 2.3520 - val_loss: 6.5526 - val_mae: 2.4175
Epoch 11/100
17/17 [==============================] - 0s 3ms/step - loss: 6.1252 - ma
e: 2.3025 - val_loss: 6.3189 - val_mae: 2.3687
Epoch 12/100
17/17 [==============================] - 0s 4ms/step - loss: 5.8995 - ma
e: 2.2532 - val_loss: 6.0873 - val_mae: 2.3193
Epoch 13/100
17/17 [==============================] - 0s 3ms/step - loss: 5.6718 - ma
e: 2.2024 - val_loss: 5.8616 - val_mae: 2.2701
Epoch 14/100
17/17 [==============================] - 0s 3ms/step - loss: 5.4461 - ma
e: 2.1509 - val_loss: 5.6368 - val_mae: 2.2201
Epoch 15/100
17/17 [==============================] - 0s 3ms/step - loss: 5.2412 - ma
e: 2.1030 - val_loss: 5.4218 - val_mae: 2.1711
Epoch 16/100
17/17 [==============================] - 0s 3ms/step - loss: 5.0414 - ma
e: 2.0551 - val_loss: 5.2089 - val_mae: 2.1219
Epoch 17/100
17/17 [==============================] - 0s 4ms/step - loss: 4.8408 - ma
e: 2.0043 - val_loss: 4.9934 - val_mae: 2.0713
Epoch 18/100
17/17 [==============================] - 0s 3ms/step - loss: 4.6374 - ma
e: 1.9547 - val_loss: 4.7886 - val_mae: 2.0220
Epoch 19/100
17/17 [==============================] - 0s 4ms/step - loss: 4.4714 - ma
e: 1.9097 - val_loss: 4.5893 - val_mae: 1.9734
Epoch 20/100
17/17 [==============================] - 0s 4ms/step - loss: 4.2584 - ma
e: 1.8597 - val_loss: 4.3932 - val_mae: 1.9245
Epoch 21/100
17/17 [==============================] - 0s 2ms/step - loss: 4.0735 - ma
e: 1.8100 - val_loss: 4.2029 - val_mae: 1.8759
Epoch 22/100
17/17 [==============================] - 0s 3ms/step - loss: 3.8880 - ma
e: 1.7588 - val_loss: 4.0201 - val_mae: 1.8278
Epoch 23/100
17/17 [==============================] - 0s 3ms/step - loss: 3.7383 - ma
e: 1.7159 - val_loss: 3.8446 - val_mae: 1.7805
Epoch 24/100
17/17 [==============================] - 0s 3ms/step - loss: 3.5826 - ma
e: 1.6730 - val_loss: 3.6808 - val_mae: 1.7351
Epoch 25/100
17/17 [==============================] - 0s 3ms/step - loss: 3.4071 - ma
e: 1.6238 - val_loss: 3.5143 - val_mae: 1.6880
Epoch 26/100
17/17 [==============================] - 0s 3ms/step - loss: 3.2807 - ma
e: 1.5849 - val_loss: 3.3589 - val_mae: 1.6433
Epoch 27/100
17/17 [==============================] - 0s 3ms/step - loss: 3.1138 - ma
e: 1.5382 - val_loss: 3.2086 - val_mae: 1.5993
```

```
Epoch 28/100
17/17 [==============================] - 0s 3ms/step - loss: 2.9877 - ma
e: 1.4974 - val_loss: 3.0707 - val_mae: 1.5580
Epoch 29/100
17/17 [==============================] - 0s 4ms/step - loss: 2.8554 - ma
e: 1.4539 - val_loss: 2.9359 - val_mae: 1.5170
Epoch 30/100
17/17 [==============================] - 0s 3ms/step - loss: 2.7256 - ma
e: 1.4168 - val_loss: 2.8061 - val_mae: 1.4770
Epoch 31/100
17/17 [==============================] - 0s 3ms/step - loss: 2.6260 - ma
e: 1.3849 - val_loss: 2.6869 - val_mae: 1.4400
Epoch 32/100
17/17 [==============================] - 0s 3ms/step - loss: 2.4897 - ma
e: 1.3411 - val_loss: 2.5714 - val_mae: 1.4035
Epoch 33/100
17/17 [==============================] - 0s 3ms/step - loss: 2.3916 - ma
e: 1.3109 - val_loss: 2.4639 - val_mae: 1.3685
Epoch 34/100
17/17 [==============================] - 0s 3ms/step - loss: 2.3099 - ma
e: 1.2767 - val_loss: 2.3622 - val_mae: 1.3345
Epoch 35/100
17/17 [==============================] - 0s 3ms/step - loss: 2.2066 - ma
e: 1.2456 - val_loss: 2.2670 - val_mae: 1.3025
Epoch 36/100
17/17 [==============================] - 0s 3ms/step - loss: 2.1407 - ma
e: 1.2177 - val_loss: 2.1767 - val_mae: 1.2715
Epoch 37/100
17/17 [==============================] - 0s 4ms/step - loss: 2.0671 - ma
e: 1.1975 - val_loss: 2.0895 - val_mae: 1.2410
Epoch 38/100
17/17 [==============================] - 0s 3ms/step - loss: 1.9792 - ma
e: 1.1644 - val_loss: 2.0078 - val_mae: 1.2118
Epoch 39/100
17/17 [==============================] - 0s 3ms/step - loss: 1.8599 - ma
e: 1.1271 - val_loss: 1.9289 - val_mae: 1.1832
Epoch 40/100
17/17 [==============================] - 0s 3ms/step - loss: 1.8825 - ma
e: 1.1234 - val_loss: 1.8565 - val_mae: 1.1562
Epoch 41/100
17/17 [==============================] - 0s 3ms/step - loss: 1.7923 - ma
e: 1.0920 - val_loss: 1.7897 - val_mae: 1.1304
Epoch 42/100
17/17 [==============================] - 0s 3ms/step - loss: 1.7154 - ma
e: 1.0691 - val_loss: 1.7250 - val_mae: 1.1049
Epoch 43/100
17/17 [==============================] - 0s 2ms/step - loss: 1.6364 - ma
e: 1.0418 - val_loss: 1.6650 - val_mae: 1.0812
Epoch 44/100
17/17 [==============================] - 0s 3ms/step - loss: 1.5994 - ma
e: 1.0278 - val_loss: 1.6072 - val_mae: 1.0587
Epoch 45/100
17/17 [==============================] - 0s 3ms/step - loss: 1.5259 - ma
e: 1.0030 - val_loss: 1.5543 - val_mae: 1.0385
Epoch 46/100
17/17 [==============================] - 0s 3ms/step - loss: 1.5216 - ma
e: 0.9950 - val_loss: 1.5024 - val_mae: 1.0185
Epoch 47/100
17/17 [==============================] - 0s 3ms/step - loss: 1.4914 - ma
e: 0.9798 - val_loss: 1.4532 - val_mae: 0.9991
Epoch 48/100
17/17 [==============================] - 0s 4ms/step - loss: 1.4282 - ma
e: 0.9600 - val_loss: 1.4072 - val_mae: 0.9808
Epoch 49/100
```

```
17/17 [==============================] - 0s 3ms/step - loss: 1.3595 - ma
e: 0.9400 - val_loss: 1.3615 - val_mae: 0.9622
Epoch 50/100
17/17 [==============================] - 0s 3ms/step - loss: 1.3643 - ma
e: 0.9324 - val_loss: 1.3212 - val_mae: 0.9463
Epoch 51/100
17/17 [==============================] - 0s 3ms/step - loss: 1.3543 - ma
e: 0.9265 - val_loss: 1.2872 - val_mae: 0.9328
Epoch 52/100
17/17 [==============================] - 0s 3ms/step - loss: 1.3045 - ma
e: 0.9152 - val_loss: 1.2514 - val_mae: 0.9185
Epoch 53/100
17/17 [==============================] - 0s 2ms/step - loss: 1.2150 - ma
e: 0.8802 - val_loss: 1.2166 - val_mae: 0.9043
Epoch 54/100
17/17 [==============================] - 0s 3ms/step - loss: 1.2402 - ma
e: 0.8819 - val_loss: 1.1854 - val_mae: 0.8917
Epoch 55/100
17/17 [==============================] - 0s 3ms/step - loss: 1.2422 - ma
e: 0.8776 - val_loss: 1.1552 - val_mae: 0.8795
Epoch 56/100
17/17 [==============================] - 0s 3ms/step - loss: 1.1463 - ma
e: 0.8504 - val_loss: 1.1281 - val_mae: 0.8681
Epoch 57/100
17/17 [==============================] - 0s 3ms/step - loss: 1.1230 - ma
e: 0.8428 - val_loss: 1.1022 - val_mae: 0.8574
Epoch 58/100
17/17 [==============================] - 0s 2ms/step - loss: 1.1447 - ma
e: 0.8459 - val_loss: 1.0781 - val_mae: 0.8472
Epoch 59/100
17/17 [==============================] - 0s 3ms/step - loss: 1.1260 - ma
e: 0.8446 - val_loss: 1.0553 - val_mae: 0.8374
Epoch 60/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0858 - ma
e: 0.8260 - val_loss: 1.0336 - val_mae: 0.8278
Epoch 61/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0743 - ma
e: 0.8206 - val_loss: 1.0137 - val_mae: 0.8190
Epoch 62/100
17/17 [==============================] - 0s 2ms/step - loss: 1.0893 - ma
e: 0.8265 - val_loss: 0.9951 - val_mae: 0.8108
Epoch 63/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0106 - ma
e: 0.8004 - val_loss: 0.9773 - val_mae: 0.8030
Epoch 64/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9993 - ma
e: 0.7948 - val_loss: 0.9617 - val_mae: 0.7960
Epoch 65/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0180 - ma
e: 0.7967 - val_loss: 0.9454 - val_mae: 0.7890
Epoch 66/100
17/17 [==============================] - 0s 2ms/step - loss: 1.0245 - ma
e: 0.7994 - val_loss: 0.9304 - val_mae: 0.7825
Epoch 67/100
17/17 [==============================] - 0s 3ms/step - loss: 1.0154 - ma
e: 0.7930 - val_loss: 0.9169 - val_mae: 0.7769
Epoch 68/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9927 - ma
e: 0.7888 - val_loss: 0.9049 - val_mae: 0.7723
Epoch 69/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9782 - ma
e: 0.7816 - val_loss: 0.8939 - val_mae: 0.7681
Epoch 70/100
17/17 [==============================] - 0s 4ms/step - loss: 0.9503 - ma
```

```
e: 0.7754 - val_loss: 0.8823 - val_mae: 0.7636
Epoch 71/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9554 - ma
e: 0.7729 - val_loss: 0.8716 - val_mae: 0.7594
Epoch 72/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9617 - ma
e: 0.7692 - val_loss: 0.8611 - val_mae: 0.7555
Epoch 73/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9278 - ma
e: 0.7647 - val_loss: 0.8527 - val_mae: 0.7523
Epoch 74/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9534 - ma
e: 0.7740 - val_loss: 0.8447 - val_mae: 0.7493
Epoch 75/100
17/17 [==============================] - 0s 2ms/step - loss: 0.9598 - ma
e: 0.7685 - val_loss: 0.8367 - val_mae: 0.7461
Epoch 76/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9222 - ma
e: 0.7584 - val_loss: 0.8283 - val_mae: 0.7427
Epoch 77/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9333 - ma
e: 0.7608 - val_loss: 0.8204 - val_mae: 0.7394
Epoch 78/100
17/17 [==============================] - 0s 4ms/step - loss: 0.9398 - ma
e: 0.7606 - val_loss: 0.8137 - val_mae: 0.7364
Epoch 79/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9021 - ma
e: 0.7533 - val_loss: 0.8067 - val_mae: 0.7336
Epoch 80/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9230 - ma
e: 0.7591 - val_loss: 0.8005 - val_mae: 0.7312
Epoch 81/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8872 - ma
e: 0.7479 - val_loss: 0.7941 - val_mae: 0.7287
Epoch 82/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8785 - ma
e: 0.7452 - val_loss: 0.7893 - val_mae: 0.7267
Epoch 83/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8971 - ma
e: 0.7529 - val_loss: 0.7851 - val_mae: 0.7249
Epoch 84/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8768 - ma
e: 0.7432 - val_loss: 0.7804 - val_mae: 0.7229
Epoch 85/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8971 - ma
e: 0.7469 - val_loss: 0.7759 - val_mae: 0.7210
Epoch 86/100
17/17 [==============================] - 0s 2ms/step - loss: 0.8717 - ma
e: 0.7390 - val_loss: 0.7720 - val_mae: 0.7193
Epoch 87/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8795 - ma
e: 0.7450 - val_loss: 0.7704 - val_mae: 0.7186
Epoch 88/100
17/17 [==============================] - 0s 2ms/step - loss: 0.8802 - ma
e: 0.7461 - val_loss: 0.7680 - val_mae: 0.7175
Epoch 89/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8523 - ma
e: 0.7390 - val_loss: 0.7642 - val_mae: 0.7157
Epoch 90/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8831 - ma
e: 0.7485 - val_loss: 0.7613 - val_mae: 0.7144
Epoch 91/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8958 - ma
e: 0.7470 - val_loss: 0.7581 - val_mae: 0.7129
```

```
Epoch 92/100
17/17 [==============================] - 0s 2ms/step - loss: 0.8648 - ma
e: 0.7427 - val_loss: 0.7561 - val_mae: 0.7120
Epoch 93/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8945 - ma
e: 0.7448 - val_loss: 0.7538 - val_mae: 0.7109
Epoch 94/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8803 - ma
e: 0.7425 - val_loss: 0.7529 - val_mae: 0.7104
Epoch 95/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8933 - ma
e: 0.7478 - val_loss: 0.7515 - val_mae: 0.7097
Epoch 96/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8857 - ma
e: 0.7480 - val_loss: 0.7491 - val_mae: 0.7085
Epoch 97/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8505 - ma
e: 0.7348 - val_loss: 0.7470 - val_mae: 0.7074
Epoch 98/100
17/17 [==============================] - 0s 3ms/step - loss: 0.8570 - ma
e: 0.7400 - val_loss: 0.7456 - val_mae: 0.7068
Epoch 99/100
17/17 [==============================] - 0s 2ms/step - loss: 0.8614 - ma
e: 0.7415 - val_loss: 0.7432 - val_mae: 0.7055
Epoch 100/100
17/17 [==============================] - 0s 3ms/step - loss: 0.9280 - ma
e: 0.7598 - val_loss: 0.7415 - val_mae: 0.7048
```

In [47]:

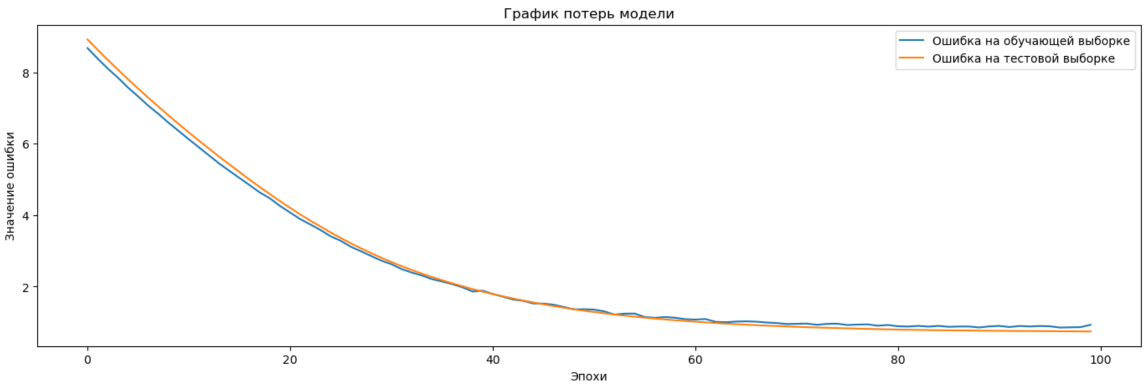
```
# оценим модель
scores = model_1.evaluate(x_test, y_test)
print("\n%s: %.2f%%" % (model_1.metrics_names[1], scores[1]*100))
```

```
9/9 [==============================] - 0s 3ms/step - loss: 0.8063 - mae:
0.7136

mae: 71.36%
```

In [48]:

```
# Посмотрим на график потерь на тренировочной и тестовой выборках
def model_1_loss_plot(model_1_hist):
    plt.figure(figsize = (17,5))
    plt.plot(model_1_hist.history['loss'],
             label = 'ошибка на обучающей выборке')
    plt.plot(model_1_hist.history['val_loss'],
            label = 'ошибка на тестовой выборке')
    plt.title('График потерь модели')
    plt.ylabel('Значение ошибки')
    plt.xlabel('Эпохи')
    plt.legend(['Ошибка на обучающей выборке', 'Ошибка на тестовой выборк
    plt.show()
model_1_loss_plot(model_1_hist)
```



In [49]:

```python
# Зададим функцию для визуализации факт/прогноз для результатов моделей
# Посмотрим на график результата работы модели
def actual_and_predicted_plot(orig, predict, var, model_name):
    plt.figure(figsize=(17,5))
    plt.title(f'Тестовые и прогнозные значения: {model_name}')
    plt.plot(orig, label = 'Тест')
    plt.plot(predict, label = 'Прогноз')
    plt.legend(loc = 'best')
    plt.ylabel(var)
    plt.xlabel('Количество наблюдений')
    plt.show()
actual_and_predicted_plot(y_test.values, model_1.predict(x_test.values),
```

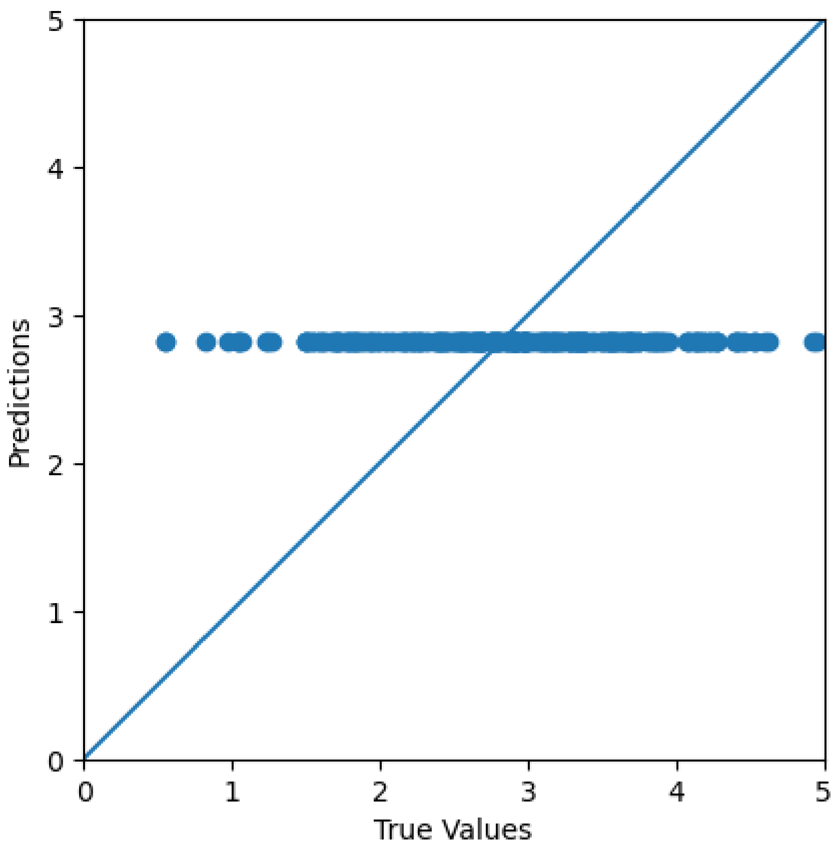
```
9/9 [==============================] - 0s 0s/step
```

In [50]:

```python
test_predictions = model_1.predict(x_test).flatten()

a = plt.axes(aspect = 'equal')
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 5]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
9/9 [==============================] - 0s 0s/step
```



In [1]:
```
#Сохраним первый вариант нейросети и проверим ее загрузку
```

In [51]:
```
model.save('App/model/NEIRO')
```

```
WARNING:absl:Found untraced functions such as _update_step_xla while savi
ng (showing 1 of 1). These functions will not be directly callable after
loading.
INFO:tensorflow:Assets written to: App/model/NEIRO\assets
INFO:tensorflow:Assets written to: App/model/NEIRO\assets
```

In [52]:
```
model_loaded = keras.models.load_model('App/model/NEIRO')
```

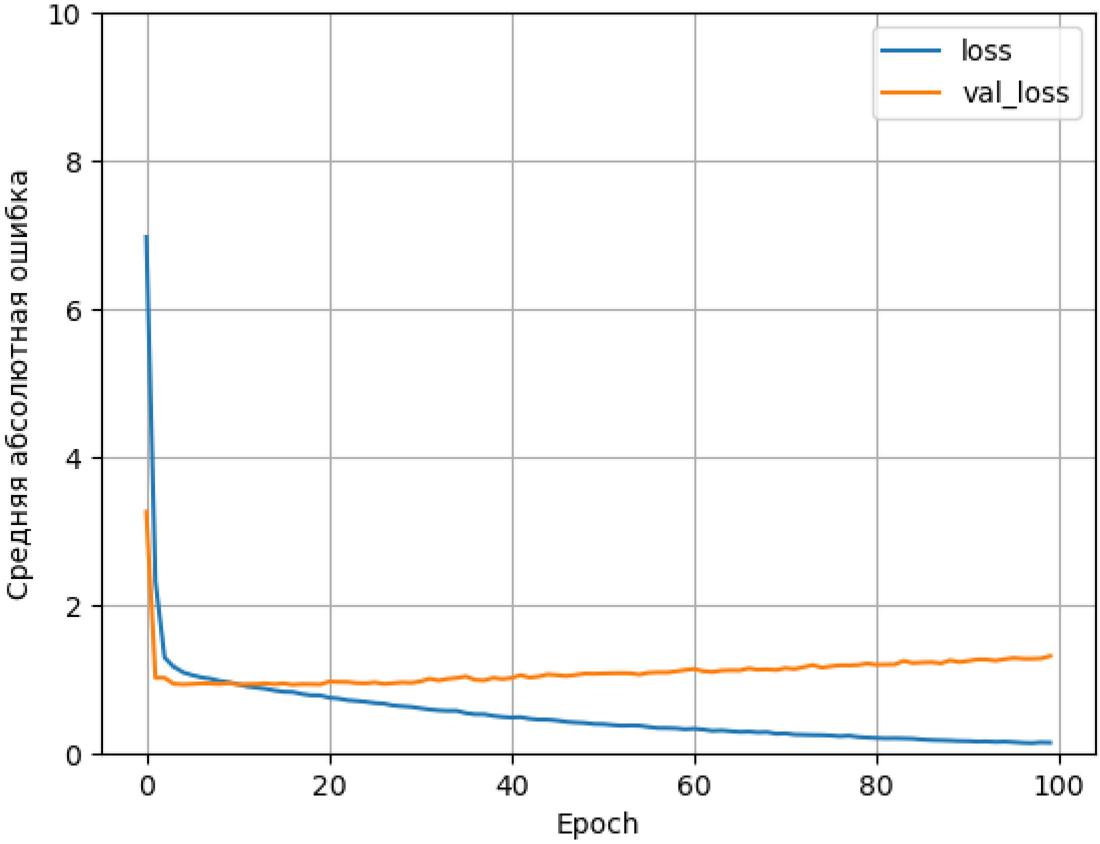
In [53]:
```
model_loaded.evaluate(x_test, y_test)
```

```
9/9 [==============================] - 0s 813us/step - loss: 1.3124 - ma
e: 0.9175
```

Out[53]:
```
[1.3124141693115234, 0.9174538850784302]
```

In [54]:
```
def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 10])
  plt.xlabel('Epoch')
  plt.ylabel('Средняя абсолютная ошибка')
  plt.legend()
  plt.grid(True)
```

In [55]:
```
plot_loss(history)
```



10
8
6
4
2
0
Средняя абсолютная ошибка
0
20
40
60
80
100
Epoch
loss
val_loss


```
In [ ]:
```