

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО КОНТРОЛЬНОЙ РАБОТЕ № 1

Дисциплина: Методы машинного обучения

Студент: Маслова Анастасия

Группа: НКНбд-01-21

Москва 2024

Вариант №24

1. Набор данных: wine_quality
2. Независимая переменная: features/total sulfur dioxide
3. Зависимая переменная: features/chlorides
4. Доп. признак: имеющий максимальную дисперсию
5. Визуализация доп. признака – эмпирическая функция распределения
6. Показатель качества регрессии – MAE (mean absolute error)
7. Степень полинома: 4
8. Параметры глубокой нейронной сети: кол-во скрытых слоев – 4, кол-во нейронов в скрытом слое – 64, функция активации – гиперболический тангенс.

Решение:

1. Загрузите заданный в индивидуальном задании набор данных из TensorFlow Datasets, включая указанные в задании независимый признак и зависимый признак (отклик). Оставьте в наборе признаки, принимающие числовые значения.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
import tensorflow_datasets as tfds
import scipy as sp
from sklearn.model_selection import train_test_split
```

```
In [2]: ds = tfds.load("wine_quality", split="train")
df = tfds.as_dataframe(ds)
df.head()
```

WARNING:absl:You use TensorFlow DType <dtype: 'float32'> in tfds.features This will soon be deprecated in favor of NumPy DTypes. In the mean time it was converted to float32.
WARNING:absl:You use TensorFlow DType <dtype: 'float64'> in tfds.features This will soon be deprecated in favor of NumPy DTypes. In the mean time it was converted to float64.

Out[2]:

	features/alcohol	features/chlorides	features/citric acid	features/density	features/fixed acidity	fea
0	9.0	0.054	0.34	1.00080	7.6	
1	12.2	0.063	0.49	0.99110	6.3	
2	11.2	0.029	0.11	0.99076	5.3	
3	9.0	0.110	0.27	0.99672	6.6	
4	12.0	0.035	0.30	0.99016	5.9	



```

In [3]: df['alcohol'] = df['features/alcohol']
df.drop(columns=['features/alcohol'],inplace=True)
df['chlorides'] = df['features/chlorides']
df.drop(columns=['features/chlorides'],inplace=True)
df['citric acid'] = df['features/citric acid']
df.drop(columns=['features/citric acid'],inplace=True)
df['density'] = df['features/density']
df.drop(columns=['features/density'],inplace=True)
df['fixed acidity'] = df['features/fixed acidity']
df.drop(columns=['features/fixed acidity'],inplace=True)
df['free sulfur dioxide'] = df['features/free sulfur dioxide']
df.drop(columns=['features/free sulfur dioxide'],inplace=True)
df['pH'] = df['features/pH']
df.drop(columns=['features/pH'],inplace=True)
df['residual sugar'] = df['features/residual sugar']
df.drop(columns=['features/residual sugar'],inplace=True)
df['sulphates'] = df['features/sulphates']
df.drop(columns=['features/sulphates'],inplace=True)
df['total sulfur dioxide'] = df['features/total sulfur dioxide']
df.drop(columns=['features/total sulfur dioxide'],inplace=True)
df['volatile acidity'] = df['features/volatile acidity']
df.drop(columns=['features/volatile acidity'],inplace=True)
df.drop(columns=['quality'],inplace=True)
df.head()

```

Out[3]:

	alcohol	chlorides	citric acid	density	fixed acidity	free sulfur dioxide	pH	residual sugar	sulphates	total sulfur dioxide
0	9.0	0.054	0.34	1.00080	7.6	44.0	3.22	18.35	0.55	197.0
1	12.2	0.063	0.49	0.99110	6.3	35.0	3.38	1.20	0.42	92.0
2	11.2	0.029	0.11	0.99076	5.3	6.0	3.51	1.10	0.48	51.0
3	9.0	0.110	0.27	0.99672	6.6	20.0	3.08	10.70	0.41	103.0
4	12.0	0.035	0.30	0.99016	5.9	57.0	3.09	3.80	0.34	135.0

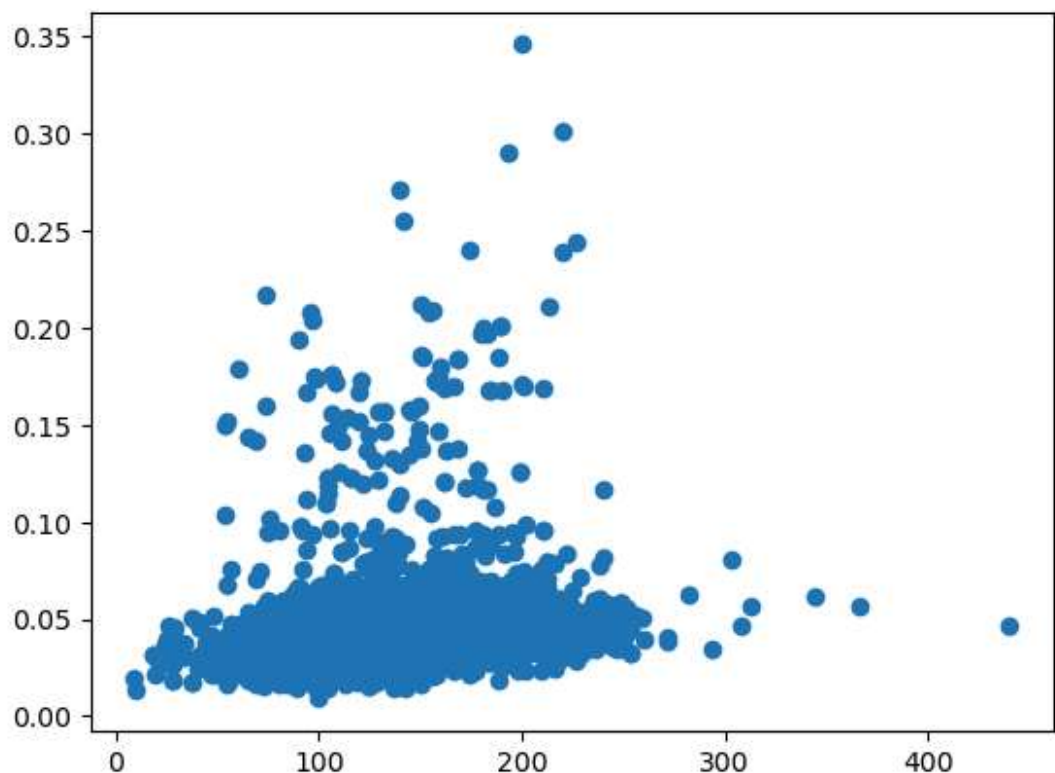
- Удалите из набора точки с выбросами при помощи стандартизованной оценки (Z-score) таким образом, чтобы точки с выбросами составляли от 5% до 10% всех точек набора данных. Визуализируйте точки исходного набора данных на плоскости в виде диаграммы рассеяния (ось X – независимый признак, ось Y – зависимый признак), показывая оставленные в наборе точки и удаленные точки разными цветами, подписывая оси и рисунок и создавая легенду.

```
In [4]: x = df['total sulfur dioxide']  
y = df["chlorides"]  
df.info()
```

```
<class 'tensorflow_datasets.core.as_dataframe.as_dataframe.<locals>.St  
yledDataFrame'>  
RangeIndex: 4898 entries, 0 to 4897  
Data columns (total 11 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   alcohol               4898 non-null   float32  
1   chlorides              4898 non-null   float32  
2   citric acid            4898 non-null   float32  
3   density                4898 non-null   float32  
4   fixed acidity           4898 non-null   float32  
5   free sulfur dioxide     4898 non-null   float32  
6   pH                     4898 non-null   float32  
7   residual sugar         4898 non-null   float32  
8   sulphates              4898 non-null   float64  
9   total sulfur dioxide    4898 non-null   float32  
10  volatile acidity       4898 non-null   float32  
dtypes: float32(10), float64(1)  
memory usage: 229.7 KB
```

```
In [5]: plt.scatter(x, y)
```

```
Out[5]: <matplotlib.collections.PathCollection at 0x1d893f13810>
```



```
In [6]: sp.stats.zscore(df, axis=0, ddof=0, nan_policy='omit')
```

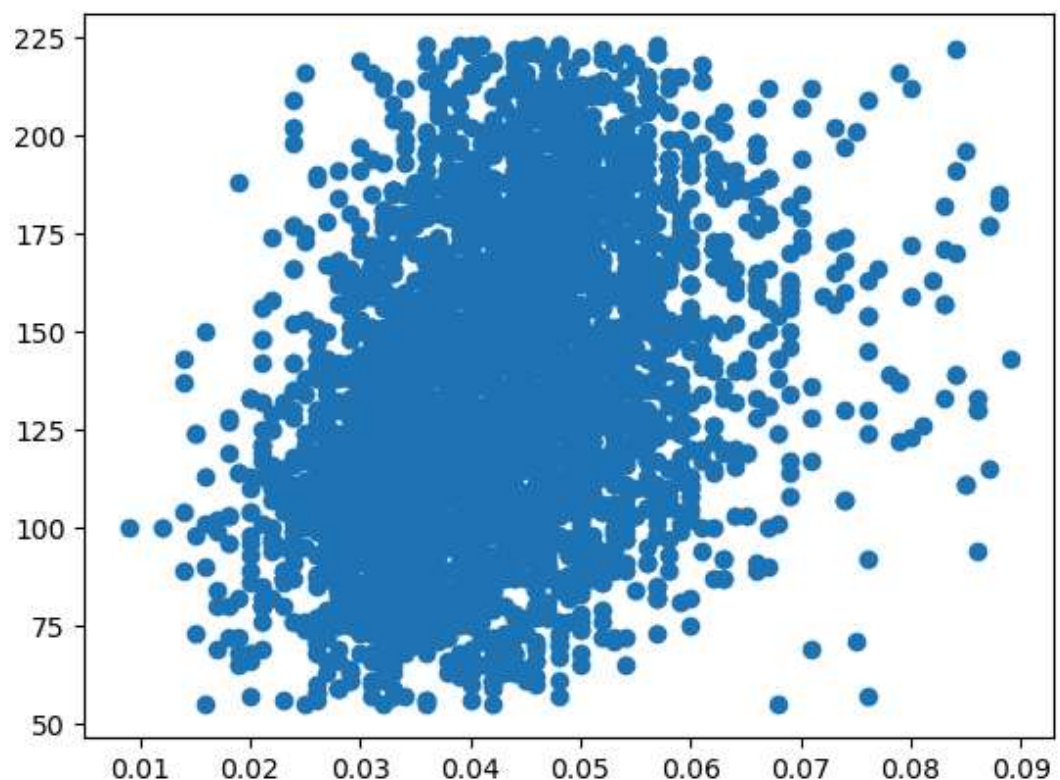
Out[6]:

	alcohol	chlorides	citric acid	density	fixed acidity	free sulfur dioxide	pH	residual sugar
0	-1.230616	0.376625	0.048001	2.264640	0.883181	0.511127	0.210176	2.3579
1	1.369963	0.788604	1.287594	-0.978854	-0.657501	-0.018117	1.269883	-1.0236
2	0.557282	-0.767763	-1.852708	-1.092538	-1.842641	-1.723457	2.130894	-1.0433
3	-1.230616	2.940054	-0.530476	0.900367	-0.301959	-0.900190	-0.717068	0.8495
4	1.207427	-0.493110	-0.282557	-1.293180	-1.131557	1.275590	-0.650837	-0.5109
...
4893	0.638551	0.239298	0.874396	-1.152729	-0.538987	0.217102	-0.518374	-1.0236
4894	0.232210	0.193523	-0.530476	0.244968	-0.894529	0.922761	0.210176	1.0073
4895	-1.474421	0.239298	2.113989	1.615937	1.475752	1.452004	-1.511848	1.4213
4896	0.069674	-0.676212	-0.447836	2.983539	-0.064931	-0.488556	-0.849532	3.8762
4897	1.857572	-0.767763	-0.447836	-1.166102	-0.301959	-1.370628	0.011481	-0.0574

4898 rows × 11 columns

```
In [7]: zscore = 2
df2 = df[['total sulfur dioxide', 'chlorides']].copy()
df2_z = df2.apply(sp.stats.zscore)
df2_out = df2[((df2_z >= -zscore).sum(axis=1)==2) & ((df2_z <= zscore).sum(axis=1)==2)]
plt.scatter(df2_out["chlorides"], df2_out["total sulfur dioxide"])
#x2_out = df2[((df2 >= -z_score).sum(axis=1)==2) & ((df2 <= z_score).sum(axis=1)==2)]
```

Out[7]: <matplotlib.collections.PathCollection at 0x1d893f2ec90>



```
In [8]: perc = round((1-df2_out.shape[0]/df.shape[0])*100,2)
perc
```

Out[8]: 6.61

3. Выполните стандартизацию независимого признака и масштабирование на интервал $[-1, 1]$ зависимого признака. Решите задачи линейной регрессии и полиномиальной регрессии для степени полинома, указанной в индивидуальном задании (4), при помощи нейронных сетей с одним нейроном и оцените качество полученных моделей по показателю, указанному в индивидуальном задании (MAE mean absolute error). Отследите обучение нейронных сетей, изменяя, при необходимости, гиперпараметры (функцию потерь, оптимизатор, шаг обучения и т.п.) или применяя регуляризацию.

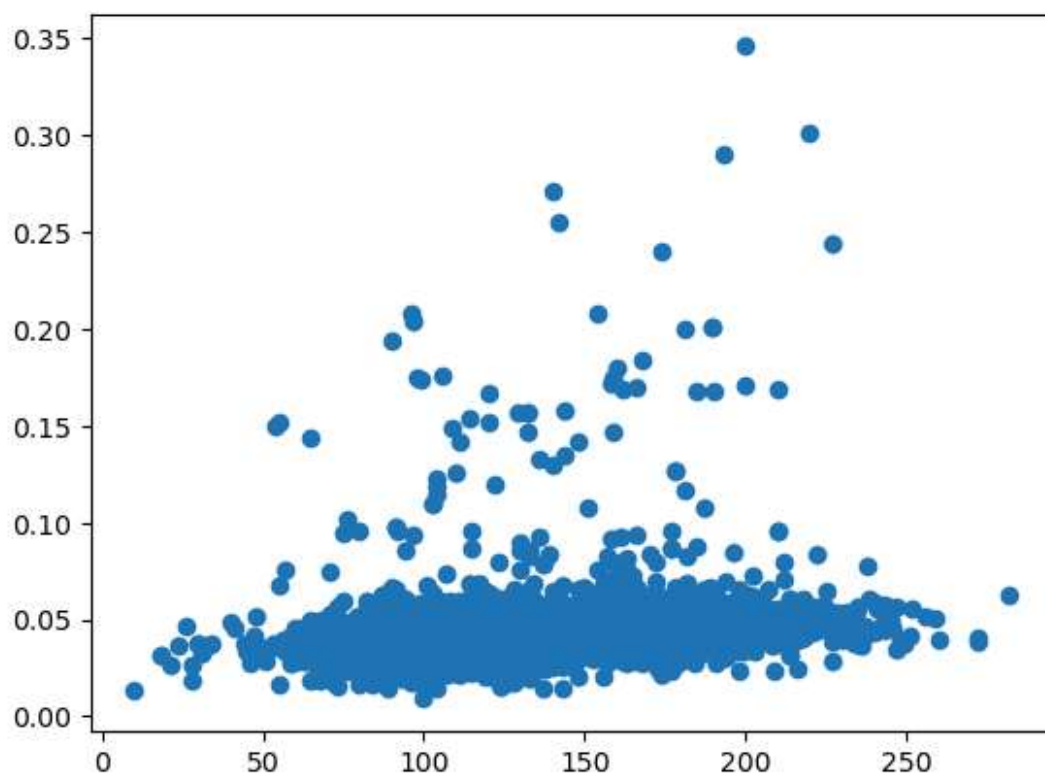
```
In [9]: df_p = df
df_p.info()
```

```
<class 'tensorflow_datasets.core.as_dataframe.as_dataframe.<locals>.StyledDataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   alcohol                             4898 non-null   float32
1   chlorides                           4898 non-null   float32
2   citric acid                         4898 non-null   float32
3   density                             4898 non-null   float32
4   fixed acidity                       4898 non-null   float32
5   free sulfur dioxide                 4898 non-null   float32
6   pH                                  4898 non-null   float32
7   residual sugar                     4898 non-null   float32
8   sulphates                          4898 non-null   float64
9   total sulfur dioxide                 4898 non-null   float32
10  volatile acidity                    4898 non-null   float32
dtypes: float32(10), float64(1)
memory usage: 229.7 KB
```

```
In [10]: df_p_train = df_p[0:2449]
df_p_test = df_p[2449:4899]
x_train = df_p_train['total sulfur dioxide']
y_train = df_p_train['chlorides']
x_test = df_p_test['total sulfur dioxide']
y_test = df_p_test['chlorides']
```

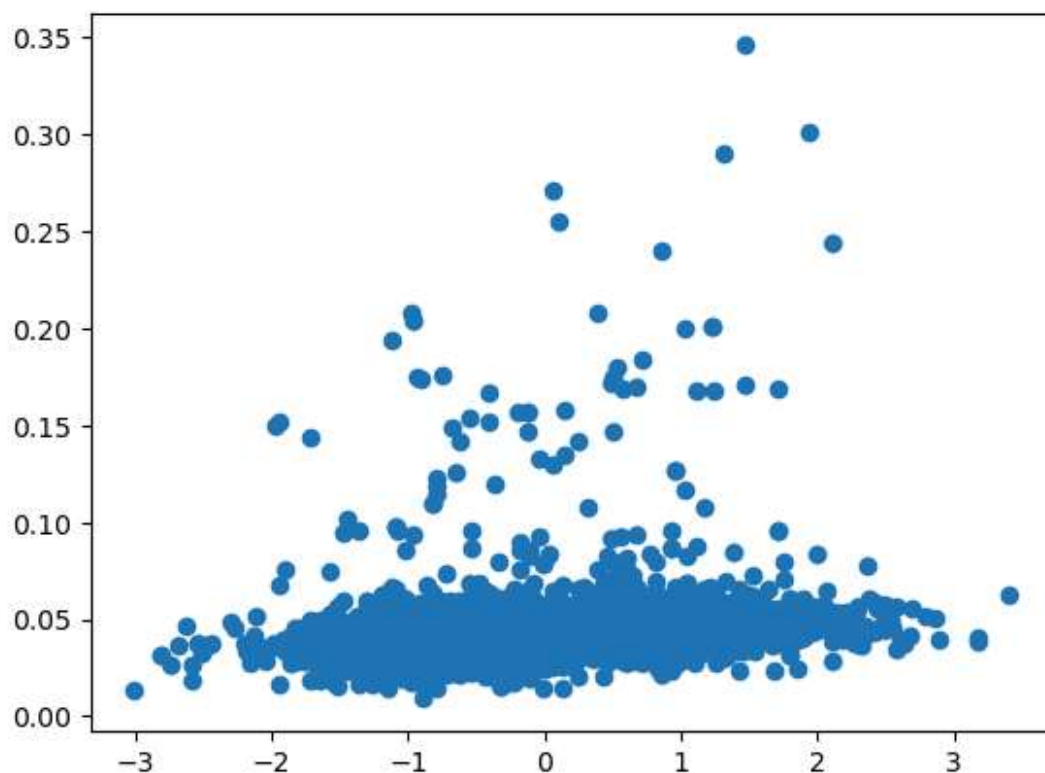
```
In [11]: ▶ plt.scatter(x_train,y_train)
```

```
Out[11]: <matplotlib.collections.PathCollection at 0x1d893f3b8d0>
```



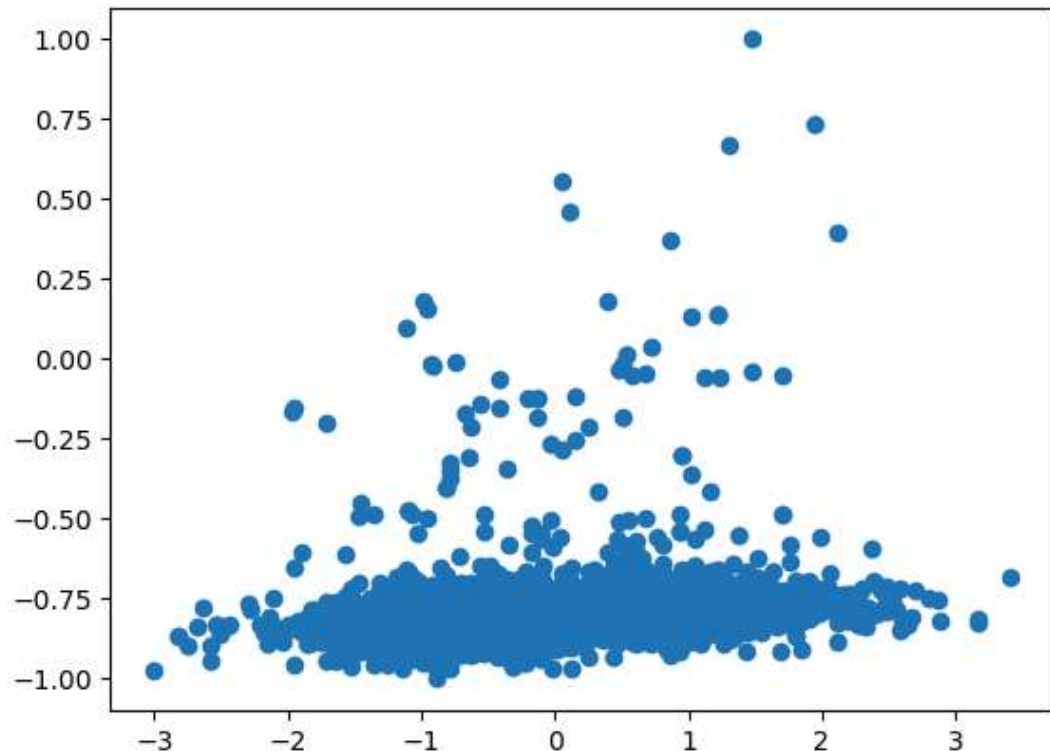
```
In [12]: ▶ #стандартизация  
x_train1 = (x_train - np.mean(x_train))/np.std(x_train)  
x_test1 = (x_test - np.mean(x_test))/np.std(x_test)  
plt.scatter(x_train1, y_train)
```

```
Out[12]: <matplotlib.collections.PathCollection at 0x1d895902010>
```




```
In [13]: ▶ #масштабирование
y_train1 = -1 + (((y_train - np.min(y_train))*2)/(np.max(y_train) - np.
y_test1 = -1 + (((y_test - np.min(y_test))*2)/(np.max(y_test) - np.min(
plt.scatter(x_train1, y_train1)
np.min(y_train1), np.max(y_train1))
```

Out[13]: (-1.0, 1.0)



Полиномиальная регрессия

```
In [14]: ▶ model = tf.keras.Sequential([ tf.keras.layers.Dense(1, input_shape=(1,

c:\Users\anast\anaconda3\Lib\site-packages\keras\src\layers\core\dens
e.py:88: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
t to a layer. When using Sequential models, prefer using an `Input(shape)`
object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
In [15]: ▶ model.compile(
    loss=tf.keras.losses.mean_absolute_error,
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.25),
    metrics=['mean_absolute_error']
)
```

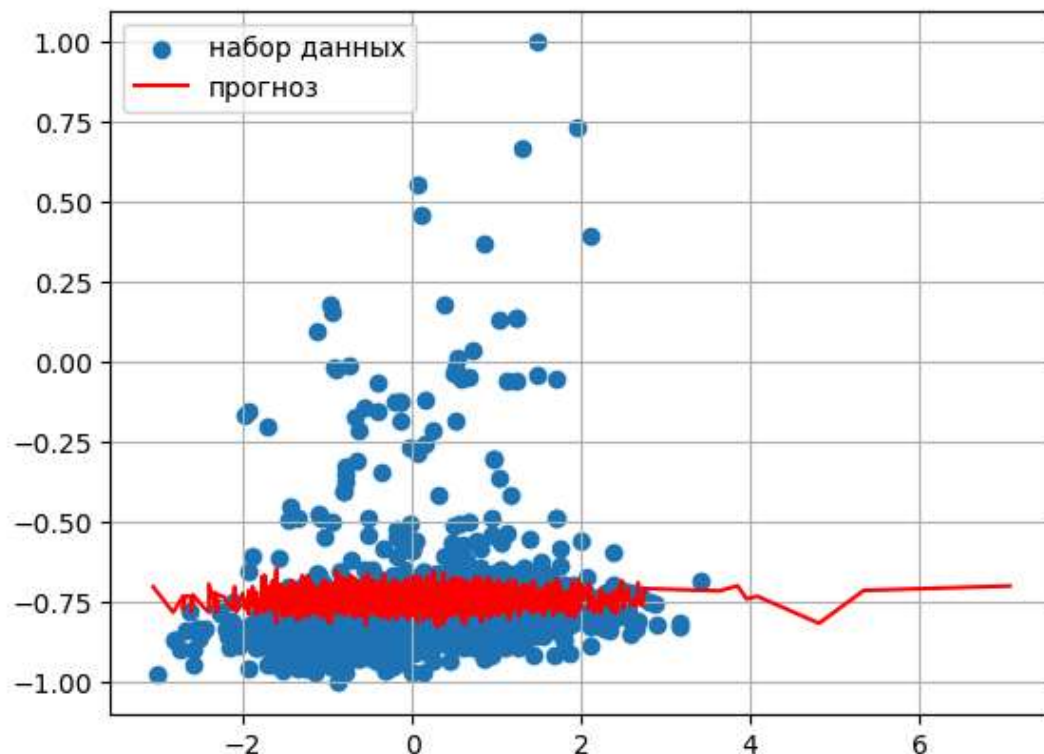

In [16]: history = model.fit(x_train1, y_train1, epochs=100)

```
Epoch 1/100
77/77 ————— 1s 2ms/step - loss: 0.2210 - mean_absolute_error: 0.2210
Epoch 2/100
77/77 ————— 0s 1ms/step - loss: 0.0746 - mean_absolute_error: 0.0746
Epoch 3/100
77/77 ————— 0s 1ms/step - loss: 0.0726 - mean_absolute_error: 0.0726
Epoch 4/100
77/77 ————— 0s 1ms/step - loss: 0.0837 - mean_absolute_error: 0.0837
Epoch 5/100
77/77 ————— 0s 1ms/step - loss: 0.0729 - mean_absolute_error: 0.0729
Epoch 6/100
77/77 ————— 0s 2ms/step - loss: 0.0734 - mean_absolute_error: 0.0734
Epoch 7/100
77/77 ————— 0s 2ms/step - loss: 0.0717 - mean_absolute_error: 0.0717
```

In [17]: y_predict = model.predict(x_train1)

```
plt.scatter(x_train1, y_train1, label='набор данных')
plt.plot(np.sort(x_test1), y_predict[np.argsort(x_test1)], color='r', linestyle='solid')
plt.legend(loc='upper left')
plt.grid()
```

77/77 ————— 0s 2ms/step



Линейная регрессия

In [18]: `model2 = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=(1`

```
c:\Users\anast\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:88: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [19]: `model2.compile(
 loss=tf.keras.losses.mean_absolute_error,
 optimizer=tf.keras.optimizers.Adam(learning_rate=0.25),
 metrics=['mean_absolute_error']
)`

In [20]: `history2 = model2.fit(x_train1, y_train1, epochs=100)`

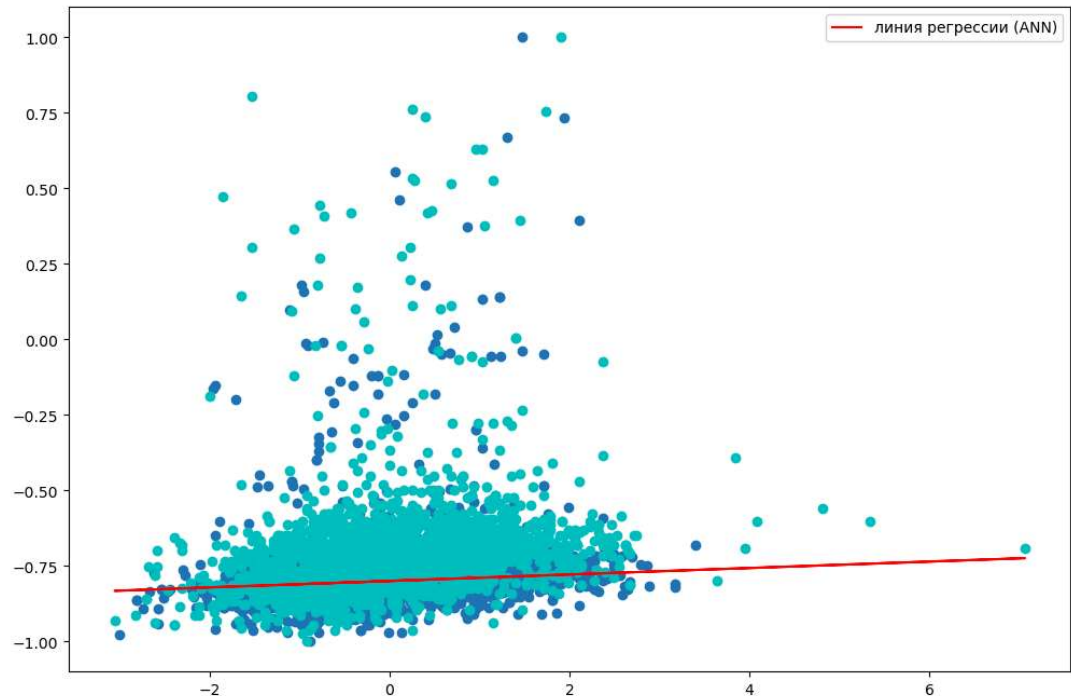
```
Epoch 1/100
77/77 ————— 1s 2ms/step - loss: 0.2007 - mean_absolute_error: 0.2007
Epoch 2/100
77/77 ————— 0s 1ms/step - loss: 0.0754 - mean_absolute_error: 0.0754
Epoch 3/100
77/77 ————— 0s 1ms/step - loss: 0.0769 - mean_absolute_error: 0.0769
Epoch 4/100
77/77 ————— 0s 1ms/step - loss: 0.0734 - mean_absolute_error: 0.0734
Epoch 5/100
77/77 ————— 0s 1ms/step - loss: 0.0687 - mean_absolute_error: 0.0687
Epoch 6/100
77/77 ————— 0s 2ms/step - loss: 0.0848 - mean_absolute_error: 0.0848
Epoch 7/100
77/77 ————— 0s 1ms/step - loss: 0.0780 - mean_absolute_error: 0.0780
```

In [21]: `y_predict2 = model2.predict(x_test1)`

```
77/77 ————— 0s 2ms/step
```

```
In [22]: ▶ plt.figure(figsize=(12,8))
plt.scatter(x_train1, y_train1)
plt.scatter(x_test1, y_test1, c='c')
plt.plot(x_test1, y_predict2, c='r', label='линия регрессии (ANN)')
plt.legend()
```

Out[22]: <matplotlib.legend.Legend at 0x1d89845f350>



4. Постройте кривые обучения для построенных нейронных сетей с зависимостью от количества эпох. На визуализации создайте легенду.

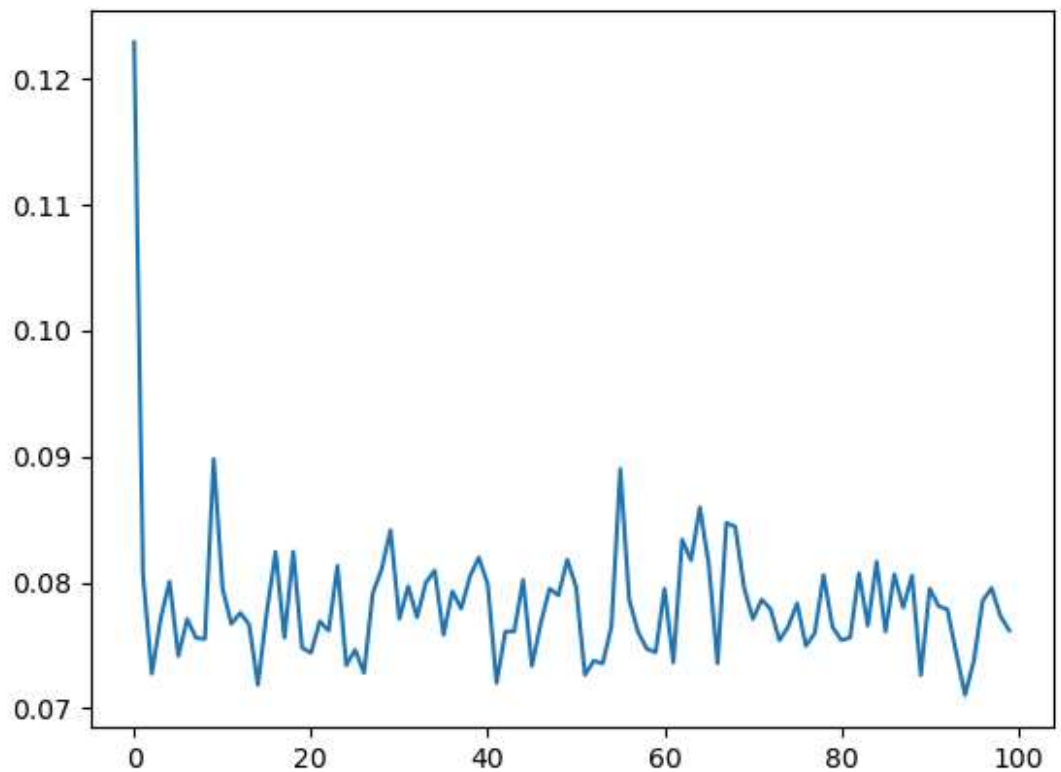
```
In [23]: ▶ def plot_loss(history):
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.ylim([0, max(history.history['loss'])*0.5])
plt.xlabel('Эпохи обучения')
plt.ylabel('Ошибка')
plt.legend()
plt.grid(True)
```

```
In [24]: plot_loss(history)
```

```
-----  
-----  
KeyError                                Traceback (most recent call  
last)  
Cell In[24], line 1  
----> 1 plot_loss(history)
```

```
Cell In[23], line 3, in plot_loss(history)  
      1 def plot_loss(history):  
      2     plt.plot(history.history['loss'], label='loss')  
----> 3     plt.plot(history.history['val_loss'], label='val_loss')  
      4     plt.ylim([0, max(history.history['loss'])*0.5])  
      5     plt.xlabel('Эпохи обучения')
```

KeyError: 'val_loss'

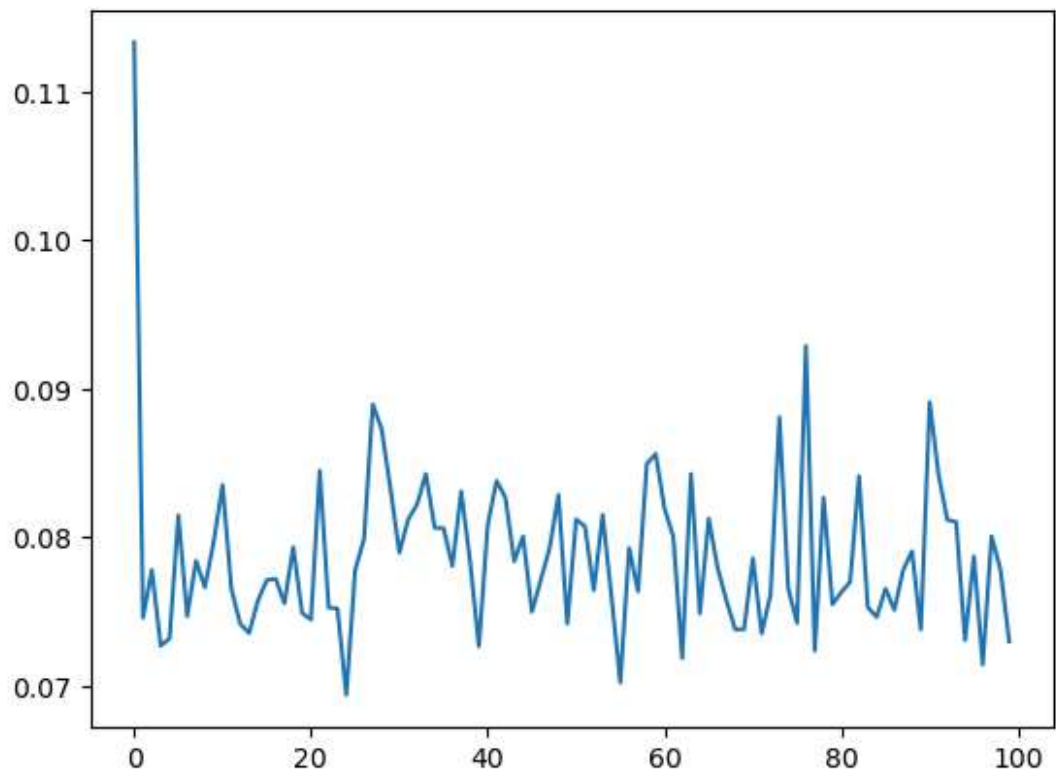


In [25]: `plot_loss(history2)`

```
-----
-----
KeyError                                Traceback (most recent call
last)
Cell In[25], line 1
----> 1 plot_loss(history2)
```

```
Cell In[23], line 3, in plot_loss(history)
      1 def plot_loss(history):
      2     plt.plot(history.history['loss'], label='loss')
----> 3     plt.plot(history.history['val_loss'], label='val_loss')
      4     plt.ylim([0, max(history.history['loss'])*0.5])
      5     plt.xlabel('Эпохи обучения')
```

KeyError: 'val_loss'



5. Визуализируйте точки набора данных на плоскости в виде диаграммы рассеяния (ось X – независимый признак, ось Y – зависимый признак), а также линии линейной и полиномиальной регрессий (другими цветами), подписывая оси и рисунок и создавая легенду.

Визуализация была сделана выше.

6. Определите в исходном наборе данных признак (отличный от независимого и зависимого признаков), принимающий непрерывные значения и имеющий свойства, указанные в индивидуальном задании.

```
In [26]: df.var()
```

```
Out[26]: alcohol          1.514432
chlorides          0.000477
citric acid        0.014646
density            0.000009
fixed acidity       0.712111
free sulfur dioxide 289.242584
pH                 0.022801
residual sugar     25.725870
sulphates          0.013025
total sulfur dioxide 1806.084595
volatile acidity    0.010159
dtype: float64
```

Независимая переменная - total sulfur dioxide, зависимая переменная - chlorides.
Кроме них, наибольшую дисперсию имеет признак free sulfur dioxide.

7. Стандартизируйте этот признак и визуализируйте его в соответствии с индивидуальным заданием (эмпирическая функция распределения).

```
In [27]: z = df["free sulfur dioxide"]
z = (z - np.mean(z))/np.std(z)
```

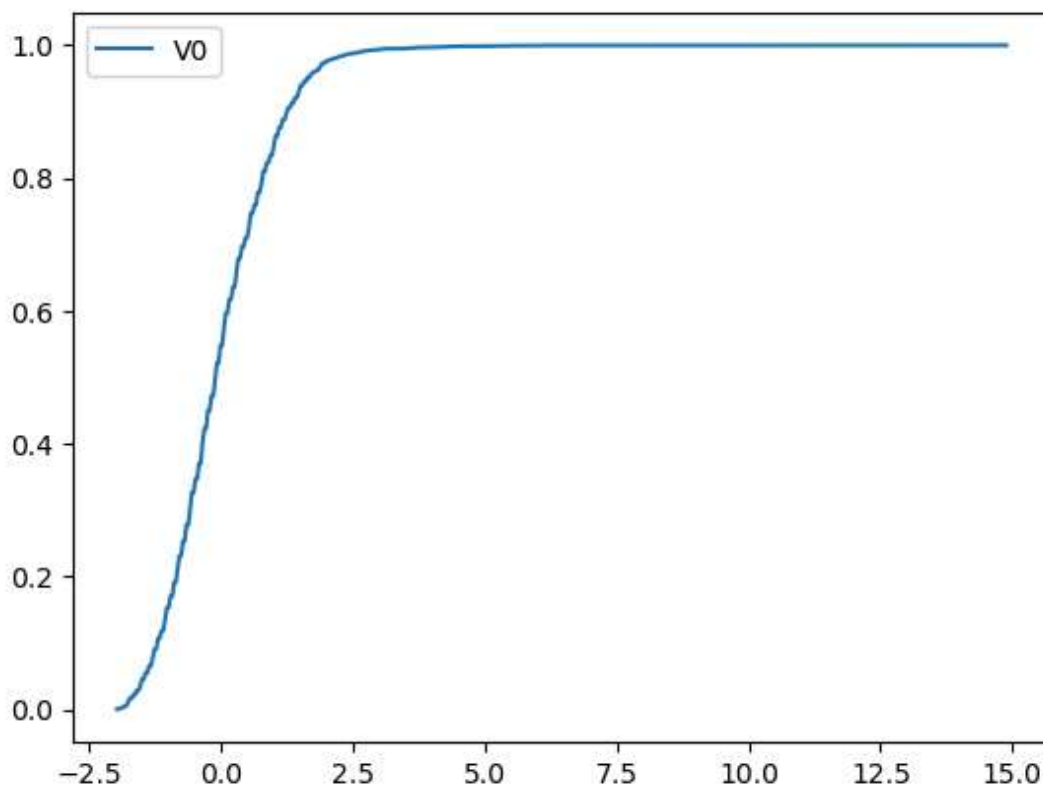
```
In [28]: def ECDF(data, x):
    counter = 0
    for v in data:
        if v <= x:
            counter += 1
    return counter / len(data)
```

```
In [29]: samples = z
npoints = 500
dz = (samples.max()-samples.min())/npoints

xlist = [samples.min()+dz*i for i in range(npoints)]
ylist = [ECDF(samples, x) for x in xlist]
```

```
In [30]: df_ECDF = pd.DataFrame(ylist, columns=['V0'], index=xlist)
df_ECDF.plot.line()
```

Out[30]: <Axes: >



8. Сформируйте набор входных данных из двух стандартизованных признаков набора данных (независимый признак и определенный признак), постройте нейронную сеть (нелинейный регрессор) с количеством скрытых слоев, количеством нейронов и функцией активации, указанными в индивидуальном задании, и одним нейроном в выходном слое и обучите ее на наборе данных из двух признаков и отклика. Отследите обучение нейронной сети, изменяя, при необходимости, гиперпараметры (функцию потерь, оптимизатор, шаг обучения и т.п.) или применяя регуляризацию.

(Параметры глубокой нейронной сети: кол-во скрытых слоев – 4, кол-во нейронов в скрытом слое – 64, функция активации – гиперболический тангенс)

```
In [31]: x8, z8 = np.array((x - np.mean(x))/np.std(x)), np.array((z - np.mean(z))
x8.shape, z8.shape)
```

Out[31]: ((4898,), (4898,))


```
In [32]: feature_normalizer = tf.keras.layers.Normalization(axis=None,input_shape=(8,))
feature_normalizer.adapt(x8)
```

c:\Users\anast\anaconda3\Lib\site-packages\keras\src\layers\preprocessing\normalization.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
super().__init__(**kwargs)

```
In [33]: large_model = tf.keras.Sequential([
    feature_normalizer,
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=1)
])

large_model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Par
normalization (Normalization)	(None, 1)	
dense_2 (Dense)	(None, 64)	
dense_3 (Dense)	(None, 64)	4
dense_4 (Dense)	(None, 64)	4
dense_5 (Dense)	(None, 64)	4
dense_6 (Dense)	(None, 1)	

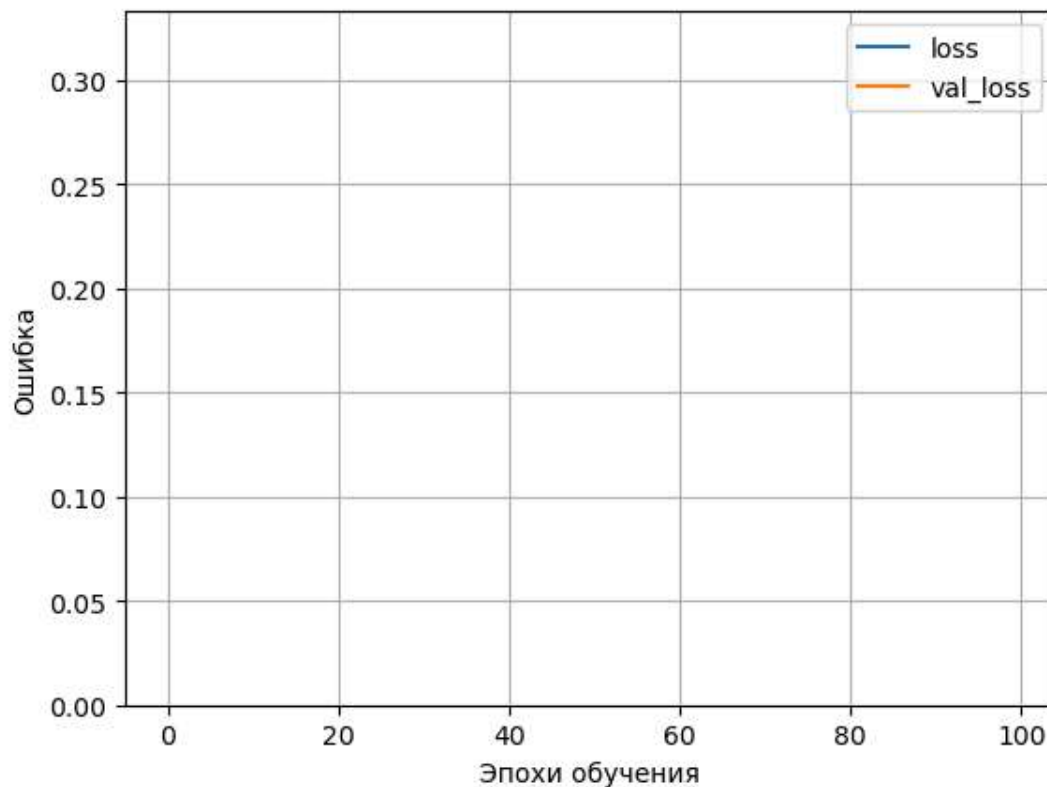
Total params: 12,676 (49.52 KB)
Trainable params: 12,673 (49.50 KB)
Non-trainable params: 3 (16.00 B)

```
In [34]: #Скомпилируем модель, используя в качестве функции потерь среднеквадратичную
large_model.compile(loss='mse')
```

```
In [35]: history8_1 = large_model.fit(
    x8, z8,
    epochs=100,
    # уровень выводимой информации
    verbose=1,
    # проверка (валидация) на 30% обучающих данных
    validation_split = 0.3)
```


```
Epoch 1/100
108/108 ————— 3s 7ms/step - loss: 0.7330 - val_loss: 0.6953
Epoch 2/100
108/108 ————— 0s 3ms/step - loss: 0.6126 - val_loss: 0.6383
Epoch 3/100
108/108 ————— 0s 3ms/step - loss: 0.6186 - val_loss: 0.6525
Epoch 4/100
108/108 ————— 0s 3ms/step - loss: 0.7136 - val_loss: 0.6085
Epoch 5/100
108/108 ————— 0s 3ms/step - loss: 0.6781 - val_loss: 0.6090
Epoch 6/100
108/108 ————— 0s 3ms/step - loss: 0.6215 - val_loss: 0.6021
Epoch 7/100
108/108 ————— 0s 3ms/step - loss: 0.6376 - val_loss: 0.6111
```

```
In [36]: plot_loss(history8_1)
```



In []: 


Обучение идет, как мне кажется, не очень успешно, потому что потери снижаются недостаточно быстро, поэтому я попробую поменять функцию потерь.

```
In [37]:  large_model1 = tf.keras.Sequential([
    feature_normalizer,
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=64, activation='tanh'),
    tf.keras.layers.Dense(units=1)
])

large_model1.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Par
normalization (Normalization)	(None, 1)	
dense_7 (Dense)	(None, 64)	
dense_8 (Dense)	(None, 64)	4
dense_9 (Dense)	(None, 64)	4
dense_10 (Dense)	(None, 64)	4
dense_11 (Dense)	(None, 1)	



Total params: 12,676 (49.52 KB)

Trainable params: 12,673 (49.50 KB)

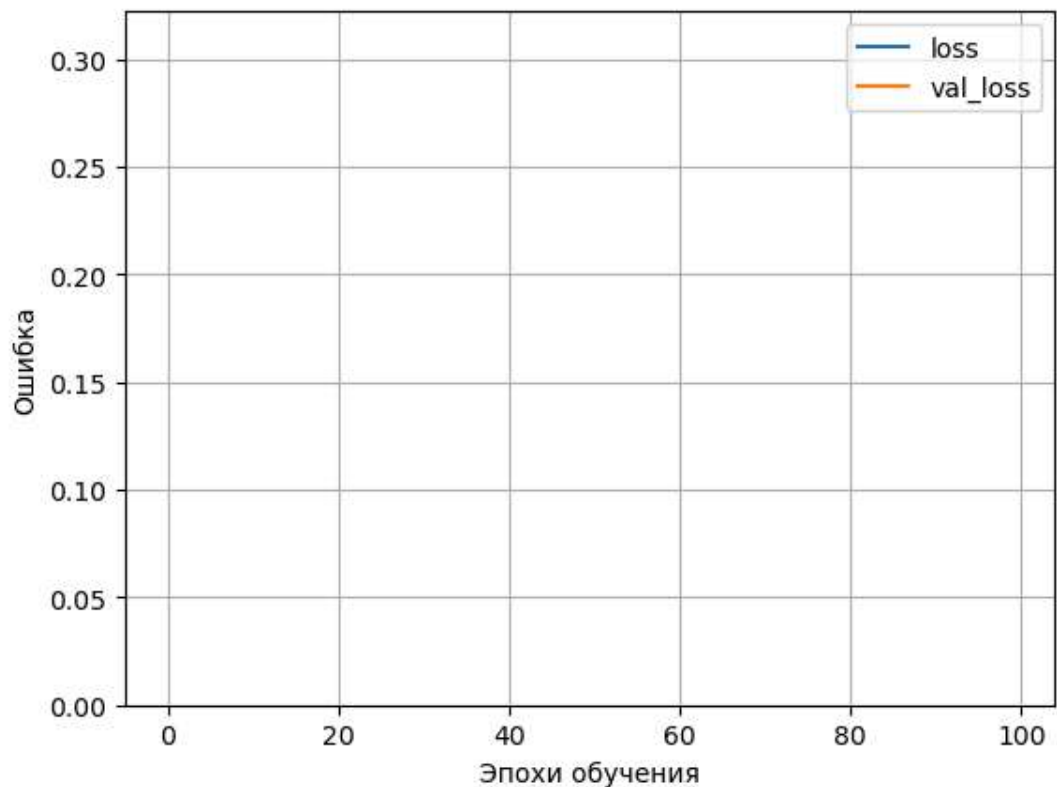
Non-trainable params: 3 (16.00 B)

```
In [38]:  large_model1.compile(loss='mae')
```

```
In [39]: history8_2 = large_model.fit(
    x8, z8,
    epochs=100,
    # уровень выводимой информации
    verbose=1,
    # проверка (валидация) на 30% обучающих данных
    validation_split = 0.3)
```

```
Epoch 1/100
108/108 ————— 1s 5ms/step - loss: 0.6392 - val_loss: 0.6031
Epoch 2/100
108/108 ————— 1s 5ms/step - loss: 0.5843 - val_loss: 0.6023
Epoch 3/100
108/108 ————— 0s 4ms/step - loss: 0.6892 - val_loss: 0.6109
Epoch 4/100
108/108 ————— 0s 3ms/step - loss: 0.6425 - val_loss: 0.6124
Epoch 5/100
108/108 ————— 0s 3ms/step - loss: 0.6251 - val_loss: 0.6085
Epoch 6/100
108/108 ————— 0s 3ms/step - loss: 0.6451 - val_loss: 0.6171
Epoch 7/100
108/108 ————— 0s 3ms/step - loss: 0.6271 - val_loss: 0.6111
```

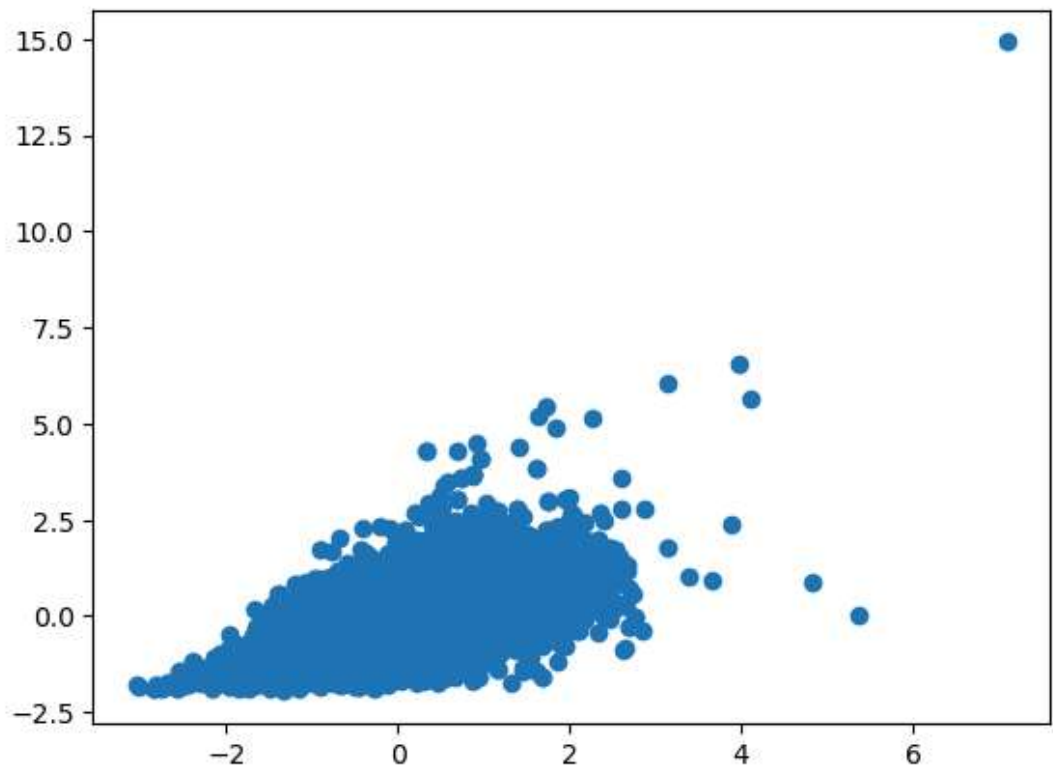
```
In [40]: plot_loss(history8_2)
```



- Визуализируйте набор данных в виде диаграммы рассеяния и прогноз нейронной сети в виде поверхности в трехмерном пространстве, подписывая оси и рисунок.

```
In [41]: ▶ plt.scatter(x8, z8)
```

```
Out[41]: <matplotlib.collections.PathCollection at 0x1d89be56990>
```



```
In [43]: ▶ z8_p = large_model1.predict(x8)
z8_p.shape
```

```
154/154 ————— 0s 2ms/step
```

```
Out[43]: (4898, 1)
```

10. Разбейте набор данных из двух признаков и отклика на обучающую и тестовую выборки и постройте кривые обучения для заданного показателя качества в зависимости от количества точек в обучающей выборке, подписывая оси и рисунок и создавая легенду.

```
In [ ]: ▶
```