```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
sns.set_context("paper", font_scale=2)

plt.style.use('seaborn-ticks')
plt.rcParams.update({'font.size': 14})

X = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
              [1, 1, 2, 1, 3, 0, 5, 10, 1, 2],  # стаж
              [500, 700, 750, 600, 1450,  # средняя стоимость занятия
               800, 1500, 2000, 450, 1000],
              [1, 1, 2, 1, 2, 1, 3, 3, 1, 2]], dtype=np.float64)  # квалификация
репетитора

y = np.array([0, 0, 1, 0, 1, 0, 1, 0, 1, 1])  # подходит или нет репетитор


def calc_std_feat(x):
    res = (x - x.mean()) / x.std()
    return res


def calc_logloss(y, y_pred):
    err = np.mean(- y * np.log(y_pred) - (1.0 - y) * np.log(1.0 - y_pred))
    return err
```

```
def sigmoid(z):
    res = 1 / (1 + np.exp(-z))
    return res


def calc_mse(y, y_pred):
    err = np.mean((y - y_pred) ** 2)
    return err


z = np.linspace(-10, 10, 101)


def eval_LR_model(X, y, iterations, alpha=1e-4):
    np.random.seed(42)
    w = np.random.randn(X.shape[0])
    n = X.shape[1]
    for i in range(1, iterations + 1):

        z = np.dot(w, X)
        y_pred = sigmoid(z)
        logloss_err = calc_logloss(y, y_pred)

        y_pred = np.dot(w, X)
        mse_err = calc_mse(y, y_pred)

        w -= alpha * (1 / n * np.dot((y_pred - y), X.T))
        if i % (iterations / 10) == 0:
```

```
            print(i, w, logloss_err, mse_err)
    return w, logloss_err


# 1 *Измените функцию calc_logloss так, чтобы нули по возможности не попадали в np.log
 (как вариант - np.clip).
X_st = X.copy().astype(np.float64)
X_st[1] = (X[1] - X[1].mean()) / X[1].std()
X_st[2] = (X[2] - X[2].mean()) / X[2].std()
X_st[3] = (X[3] - X[3].mean()) / X[3].std()
X_st[0, 0] = 0 # Подбрасываем нули в массив
X_st[1, 1] = 0 # Подбрасываем нули в массив

X_st[X_st == 0] = 1e-6 # Преобразовываем нули в какой то достаточный минимум

print(X_st)

# 2 Подберите аргументы функции eval_LR_model для логистической регрессии таким образом
, чтобы log loss был минимальным.
alpha = 1e-4
delta_alpha = 1e-4
limit_alpha = 1e-4
first = True
while alpha <= limit_alpha:
    (w, log_loss) = eval_LR_model(X_st, y, 1000, alpha)
    if first:
        data = np.array([[alpha, log_loss], ])
        first = False
    else:
```

```
        data = np.append(data, [[alpha, log_loss], ], axis=0)
    alpha += delta_alpha

plt.scatter(data[:, 0], data[:, 1])
plt.show()

# Ответ альфа после 0.02 logloss не меняется сильно.

print(1)
# 3 Создайте функцию calc_pred_proba, возвращающую предсказанную вероятность класса 1 (
на вход подаются веса, которые уже посчитаны функцией eval_LR_model и X, на выходе -
массив y_pred_proba).

def calc_pred_proba(w, X):
    return sigmoid(np.dot(w, X))

w, log_loss = eval_LR_model(X_st, y, 1000, alpha)

print(calc_pred_proba(w, X_st))


# 4 Создайте функцию calc_pred, возвращающую предсказанный класс (на вход подаются веса
, которые уже посчитаны функцией eval_LR_model и X, на выходе - массив y_pred).
def calc_pred(w, X):
    m = X.shape[1]

    y_predicted = np.zeros((1, m))
    w = w.reshape(X.shape[0], 1)
```

```
    A = sigmoid(np.dot(w.T, X))

    #     За порог отнесения к тому или иному классу примем вероятность 0.5
    for i in range(A.shape[1]):
        if (A[:, i] > 0.5):
            y_predicted[:, i] = 1
        elif (A[:, i] <= 0.5):
            y_predicted[:, i] = 0

    return y_predicted


y_pred = calc_pred(w, X)
print(f'y_pred={y_pred}')

# 5 Посчитайте accuracy, матрицу ошибок, precision и recall, а также F1-score.

a = [y == y_pred]
accuracy = round(np.count_nonzero(np.int_(a)) / y.__len__(), 2)
matrix_error = np.zeros((2, 2), dtype=int)

for i, x in enumerate(np.nditer(y_pred)):
    if x == 1 and y[i] == 0:  #FP
        matrix_error[0,1] += 1
    if x == 1 and y[i] == 1: #TP
        matrix_error[0, 0] += 1
    if x == 0 and y[i] == 1: # FN
        matrix_error[0, 1] += 1
    if x == 0 and y[i] == 0: # TN
```

```
        matrix_error[0, 1] += 1

precision = matrix_error[0,0]/(matrix_error[0,0] + matrix_error[1,0])
recall = matrix_error[0,0]/(matrix_error[0,0] + matrix_error[0,1])

print(f'accuracy={accuracy}')
print(f'matrix_error={matrix_error}')
print(f'precision={precision}')
print(f'recall={recall}')
print(f'F1-score={2 * precision * recall / (precision + recall)}')

# y_pred=[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
# accuracy=0.5
# matrix_error=[[5 5]
#               [0 0]]
# precision=1.0
# recall=0.5
# F1-score=0.6666666666666666

# 6 Могла ли модель переобучиться? Почему?
#Ответ: Переобучится не получится слишком мало данных в данном сете.Модель не может
выучить закономерность. Что нам и сообщает F1-score

# 7 *Создайте функции eval_LR_model_l1 и eval_LR_model_l2 с применением L1 и L2
регуляризации соответственно.
```