В [13]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
sns.set_context("paper", font_scale=2)

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 р

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

1. Измените функцию calc_logloss так, чтобы нули по возможности не попадали в np.log (как вариант - np.clip).

В [20]:

```
def calc_logloss(y, y_pred):
    y_pred = np.clip(y_pred, 0.001, 0.999)
    err = np.mean(- y * np.log(y_pred) - (1.0 - y) * np.log(1.0 - y_pred))
    return err
```

2. Подберите аргументы функции eval_LR_model для логистической регрессии таким образом, чтобы log loss был минимальным.

B [26]:

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

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)
        err = calc_logloss(y, y_pred)

        w -= alpha * (1/n * np.dot((y_pred - y), X.T))
        if i % (iterations / 50) == 0:
            print(i, w, err)
    return w

for alpha in [8e-6,6e-6,4e-6,2e-6,1e-6]:
    print(f'alpha: {alpha}')
    eval_LR_model(X, y, 20000, alpha)

weights = eval_LR_model(X, y, 20000, 8e-6)
```

```
alpha: 8e-06
400 [ 0.49586173 -0.13999361 -0.002425    1.52212495] 0.604728110758
1461
800 [ 0.49559934 -0.13979495 -0.0024313   1.52227486] 0.604683094888
0975
1200 [ 0.49533704 -0.13959672 -0.00243193  1.52242477] 0.60464229151
09513
1600 [ 0.49507483 -0.13939893 -0.00243255  1.52257467] 0.60460155871
96913
2000 [ 0.49481272 -0.13920157 -0.00243317  1.52272456] 0.60456089627
40102
2400 [ 0.49455071 -0.13900465 -0.00243379  1.52287444] 0.60452030393
92403
2800 [ 0.49428878 -0.13880817 -0.00243441  1.52302432] 0.60447978148
14078
3200 [ 0.49402696 -0.13861212 -0.00243503  1.52317418] 0.60443932866
72299
3600 [ 0.49376522 -0.1384165  -0.00243565  1.52332404] 0.60439894526
41145
```

Ответ: 8e-6

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).

B [33]:

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

calc_pred_proba(weights,X)
```

Out[33]:

```
array([0.65742376, 0.53987606, 0.80776372, 0.60008976, 0.39719037,
       0.51115079, 0.67431192, 0.23938839, 0.68456528, 0.69435432])
```

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).

B [37]:

```
def calc_pred(w, X):
    border = 0.5
    y_pred_proba = calc_pred_proba(w,X)
    return (y_pred_proba > border).astype(np.int)

y_pred = calc_pred(weights, X)
y_pred
```

Out[37]:

```
array([1, 1, 1, 1, 0, 1, 1, 0, 1, 1])
```

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

B [53]:

```
def accurancy(y_true, y_pred):
    return np.sum(y_true==y_pred) / len(y_true)

def conf_matrix(y_true, y_pred):
    matrix = { 'tp': 0, 'tn': 0, 'fp': 0, 'fn': 0 }

    for i,_ in enumerate(y_true):
        if y_true[i] == 1:
            if y_pred[i] == y_true[i]:
                el = 'tp'
            else:
                el = 'fn'
        else:
            if y_pred[i] == y_true[i]:
                el = 'tn'
            else:
                el = 'fp'

        matrix[el] += 1

    return matrix


def precision(y_true, y_pred):
    matrix = conf_matrix(y_true, y_pred)
    return (matrix['tp'] + matrix['fp']) / len(y_true)

def recall(y_true, y_pred):
    matrix = conf_matrix(y_true, y_pred)
    return (matrix['tp'] + matrix['fn']) / len(y_true)

def f1_score(y_true, y_pred):
    prec = precision(y_true, y_pred)
    rec = recall(y_true, y_pred)
    return 2/ (1 / prec + 1 / rec)

print(accurancy(y, y_pred))
print(conf_matrix(y, y_pred))
print(precision(y, y_pred))
print(recall(y, y_pred))
print(f1_score(y, y_pred))
```

```
0.5
{'tp': 4, 'tn': 1, 'fp': 4, 'fn': 1}
0.8
0.5
0.6153846153846154
```

6. Могла ли модель переобучиться? Почему?

В данном случае, так как данных не очень много, можель вполне может переобучиться. На таком датасете довольно сложно говорить о точности.

Type *Markdown* and LaTeX: α^2

B [ ]:

```

```