1. Постройте график зависимости весов всех признаков от lambda в L2-регуляризации (на данных из урока).

B [33]:

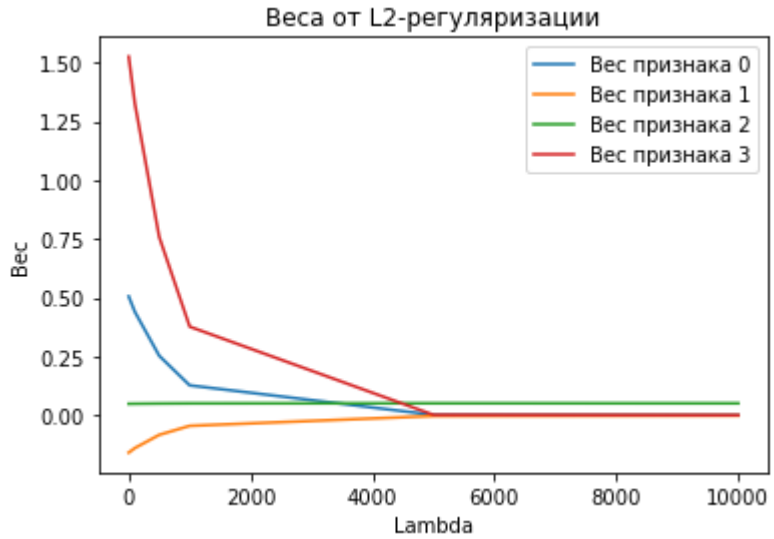
```
import numpy as np
import matplotlib.pyplot as plt

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

def eval_model_reg2(X, y, iterations, alpha=1e-4, lambda_=1e-8):
    np.random.seed(42)
    W = np.random.randn(X.shape[0])
    n = X.shape[1]
    for i in range(1, iterations + 1):
        y_pred = np.dot(W, X)
        err = calc_mse(y, y_pred)
        W -= alpha * (1/n * 2 * np.dot((y_pred - y), X.T) + 2 * lambda_ * W)
    return W
```

B [12]:

```
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, 1450,800, 1500, 2000, 450, 1000], # средняя стои
              [1, 1, 2, 1, 2, 1, 3, 3, 1, 2]])                        # квалификация

y = [45, 55, 50, 59, 65, 35, 75, 80, 50, 60]
```

B [42]:

```
# сформируем диапазон lambda
lambdas = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 5e2, 1e3, 5e3, 1e4]

weights = []

for l in lambdas:
    weight = eval_model_reg2(X,y,1000, alpha=7e-7, lambda_=l)
    weights.append(weight)


all_weights = list(zip(*weights))
#print(all_weights)

fig, ax = plt.subplots()

for i, weight in enumerate(all_weights):
    ax.plot(lambdas, weight, label=f'Вес признака {i}')

ax.set_title('Веса от L2-регуляризации')
ax.set_xlabel('Lambda')
ax.set_ylabel('Вес')
ax.legend()

plt.show()
```



2. Можно ли к одному и тому же признаку применить сразу и нормализацию и стандартизацию?

Имеет смысл только в том случае, если задублировать признак и к одному применить нормализацию, а ко второму стандартизацию и посмотреть какой вариант будет лучше отражаться на модели.

3. Напишите функцию наподобие eval_model_reg2, но для применения L1-регуляризации

B [35]:

```
def eval_model_reg1(X, y, iterations, alpha=1e-4, lambda_=1e-8):
    np.random.seed(42)
    W = np.random.randn(X.shape[0])
    n = X.shape[1]
    for i in range(1, iterations + 1):
        y_pred = np.dot(W, X)
        err = calc_mse(y, y_pred)
        W -= alpha * (1/n * 2 * np.dot((y_pred - y), X.T) + lambda_ * np.sign(W))
    return W
```

B [ ]:

```

```