РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. П. Лумумба

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Дисциплина: *Компьютерный практикум по моделированию*

Студент: Королёв Иван Андреевич

Группа: НКАбд-04-22

МОСКВА

2024 г.

Цель работы: Научиться работать с библиотеками numpy и pandas.

Выполнение работы

Тема «Вычисления с помощью Numpy»

Задание 1

Подберите скорость обучения (eta) и количество итераций

Листинг программы на языке Python:

1. Подберите скорость обучения (eta) и количество итераций

n [112]:
```
n = X.shape[0]

# Задаем списки для подбора скорости обучения (eta) и количества итераций
etas = [1e-2, 1e-3, 1e-4]
n_iters = [50, 100, 150, 200, 300, 400, 500]

# Инициализируем переменные для хранения лучших результатов
best_eta = None
best_n_iter = None
best_err = float('inf')  # Инициализируем с бесконечностью, чтобы любое начальное значение ошибки было лучше

W = np.array([1, 0.5])
print(f'Number of objects = {n} \
      \nInitial weights = {W} \n')

# Проходим по всем комбинациям скоростей обучения и количества итераций
for eta in etas:
    for n_iter in n_iters:
        W = np.array([1, 0.5])  # Восстанавливаем начальные веса перед каждым запуском
        for i in range(n_iter):
            y_pred = np.dot(X, W)
            err = calc_mse(y, y_pred)
            for k in range(W.shape[0]):
                W[k] -= eta * (1/n * 2 * X[:, k] @ (y_pred - y))

        # Сохраняем лучшую комбинацию параметров, если ошибка уменьшилась
        if err < best_err:
            best_err = err
            best_eta = eta
            best_n_iter = n_iter

print(f'Best learning rate = {best_eta}, Best number of iterations = {best_n_iter}, Best MSE = {round(best_err, 2)}')
```

```
Number of objects = 10
Initial weights = [1.  0.5]

Best learning rate = 0.01, Best number of iterations = 500, Best MSE = 44.06
```

Задание 2.

В этом коде мы избавляемся от итераций по весам, но тут есть ошибка, исправьте ее

Листинг программы на языке Python:

2. В этом коде мы избавляемся от итераций по весам, но тут есть ошибка, исправьте ее

In [115]:
```
n = X.shape[0]

eta = 1e-2
n_iter = 100

W = np.array([1, 0.5])
print(f'Number of objects = {n} \
      \nLearning rate = {eta} \
      \nInitial weights = {W} \n')

for i in range(n_iter):
    y_pred = np.dot(X, W)
    err = calc_mse(y, y_pred)
#     for k in range(W.shape[0]):
#         W[k] -= eta * (1/n * 2 * X[:, k] @ (y_pred - y))
    # ИЗМЕНЕНИЯ
    W -= eta * (1/n * 2 * np.dot(X.T, y_pred - y)) # ТРАНСПОНИРУЕМ МАТРИЦУ
    # ИЗМЕНЕНИЯ
    #
    if i % 10 == 0:
        print(f'Iteration #{i}: W_new = {W}, MSE = {round(err,2)}')
```

```
Number of objects = 10
Learning rate = 0.01
Initial weights = [1.  0.5]

Iteration #0: W_new = [2.08 4.27], MSE = 3047.75
Iteration #10: W_new = [ 7.0011236 10.6169007], MSE = 738.65
Iteration #20: W_new = [10.3486292  10.10603105], MSE = 622.03
Iteration #30: W_new = [13.38789582  9.55618391], MSE = 525.24
Iteration #40: W_new = [16.16088505  9.05336203], MSE = 444.66
Iteration #50: W_new = [18.69110735  8.59454545], MSE = 377.58
Iteration #60: W_new = [20.99981865  8.17589626], MSE = 321.72
Iteration #70: W_new = [23.10641138  7.79389815], MSE = 275.22
Iteration #80: W_new = [25.02858024  7.44534246], MSE = 236.5
Iteration #90: W_new = [26.78247081  7.12730145], MSE = 204.27
```

Задание 3.

Вместо того, чтобы задавать количество итераций, задайте другое условие остановки алгоритма - когда веса перестают изменяться меньше определенного порога

Листинг программы на языке Python:

3. Вместо того, чтобы задавать количество итераций, задайте другое условие остановки алгоритма - когда веса перестают изменяться меньше определенного порога ϵ.

In [103]:
```
epsilon = 1e-5  # Порог изменения весов
max_iter = 1000  # Максимальное количество итераций (защита от бесконечного цикла)

for i in range(max_iter):
    prev_W = W.copy()  # Сохраняем предыдущие значения весов

    # Вычисляем предсказания и ошибку
    y_pred = np.dot(X, W)
    err = calc_mse(y, y_pred)

    # Вычисляем градиент и обновляем веса
    gradient = (2/n) * np.dot(X.T, y_pred - y)
    W -= eta * gradient

    # Проверяем условие остановки
    if np.linalg.norm(W - prev_W) < epsilon:
        print(f"Алгоритм сходится на итерации {i}")
        break

    # Если порог не достигнут, итерируемся дальше
    if i % 10 == 0:
        print(f'Iteration #{i}: W_new = {W}, MSE = {round(err,2)}')
```

```
Iteration #0: W_new = [28.38281518  6.83710367], MSE = 177.43
Iteration #10: W_new = [29.84305573  6.57231156], MSE = 155.08
Iteration #20: W_new = [31.17545797  6.33070096], MSE = 136.48
Iteration #30: W_new = [32.39121367  6.11024241], MSE = 120.99
Iteration #40: W_new = [33.50053475  5.90908413], MSE = 108.09
Iteration #50: W_new = [34.51273915  5.72553647], MSE = 97.36
Iteration #60: W_new = [35.43632906  5.55805768], MSE = 88.42
Iteration #70: W_new = [36.27906231  5.405241  ], MSE = 80.98
Iteration #80: W_new = [37.0480176   5.26580281], MSE = 74.78
Iteration #90: W_new = [37.74965389  5.13857189], MSE = 69.62
Iteration #100: W_new = [38.38986469  5.02247953], MSE = 65.33
Iteration #110: W_new = [38.97402756  4.9165506 ], MSE = 61.75
Iteration #120: W_new = [39.50704928  4.81989533], MSE = 58.77
Iteration #130: W_new = [39.99340705  4.73170185], MSE = 56.29
Iteration #140: W_new = [40.43718613  4.65122936], MSE = 54.23
Iteration #150: W_new = [40.84211409  4.57780191], MSE = 52.51
Iteration #160: W_new = [41.21159221  4.51080275], MSE = 51.08
Iteration #170: W_new = [41.54872398  4.4496691 ], MSE = 49.89
Iteration #180: W_new = [41.8563412   4.39388747], MSE = 48.9
Iteration #190: W_new = [42.13702774  4.34298929], MSE = 48.07
Iteration #200: W_new = [42.39314129  4.29654705], MSE = 47.39
Iteration #210: W_new = [42.6268331   4.25417064], MSE = 46.81
Iteration #220: W_new = [42.84006612  4.21550412], MSE = 46.34
Iteration #230: W_new = [43.03463143  4.1802227 ], MSE = 45.94
Iteration #240: W_new = [43.21216332  4.14803003], MSE = 45.61
Iteration #250: W_new = [43.37415299  4.1186557 ], MSE = 45.34
```