Задание 7

1. В моей работе получилось наоборот, при стохастическом спуске при любых параметрах функционал монотонно убывает, а при градиентном спуске иногда начинает скакать (наблюдается при логарифмической функции потерь).

   Для градиентного спуска такое поведение объясняется тем, что мы идем в наиболее выгодную в данный момент сторону (против градиента), поэтому довольно очевидно, что минимизируемая функция должна стать меньше. Но нет гарантии, что любая величина шага приведет к меньшему значению, поэтому мы можем случайно сделать большой шаг и сделать хуже. Например, при логарифмической функции потерь, при больший альфа графики начинают колебаться вокруг одного значения. Так как у нас постоянный коэффициент перед градиентом, то это по всей видимости означает, что мы попали куда-то рядом с локальным минимумом, и не можем в него прийти из-за слишком большого шага, поэтому мы начинаем просто прыгать вокруг него вперед-назад.

   Для стохастического спуска в принципе те же рассуждения. В целом понятно, что мы идем примерно в направлении уменьшения функции (отличие в том, что мы берем какое-то подмножество признаков), но при этом из-за неточного выбора коэффициента перед градиентом или из-за того что берем только подмножество признаков, можем случайно прийти в точку с большим значением.

2. Для градиентного спуска легко видеть, что при больших альфа график стабилизируется быстрее (то есть алгоритм быстрее сходится).

   Для стохастического спуска отличий почти не наблюдается (даже если они есть, то по сравнению с общим количеством итераций ими можно пренебречь).

3. По графикам видно, что в целом картина везде одинаковая, но можно заметить, что при большем k график получается чуть менее округлым (более пологим) и сходится немного раньше, чем при маленьких k

4. Для стохастического спуска, основываясь на результатах вывода print_precision_recall() можно заметить, что лучшие результаты (0.59, 0.86) достигаются при

   SGD with loss=sigmoid_loss, alp = 0.000100, k=10
   SGD with loss=sigmoid_loss, alp = 0.010000, k=10
   SGD with loss=sigmoid_loss, alp = 0.010000, k=50
   SGD with loss=log_loss, alp = 0.010000, k=1
   SGD with loss=log_loss, alp = 0.010000, k=10

   Как видно, лучше брать альфа равное 0,01, зависимости от других параметров

не видно

Для градиентого спуска, основываясь на результатах вывода print_precision_recall() можно заметить, что лучшие результаты (0.58, 0.86) достигаются при

GradientDescent with loss=sigmoid_loss, alp = 0.010000
GradientDescent with loss=sigmoid_loss, alp = 1.000000

Как видно, лучше брать небольшое альфа и сигмоидную функцию потерь, зависимости от других параметров не видно

5. Видно, что при некоторых запусках получаются веса, похожие на какие-то из предыдущих (естественно, не такие же, но в целом видно сходством). Думаю, это объясняется тем, что есть несколько хороших гиперплоскостей для нашей задачи, поэтому каждый результат в итоге тяготеет к одной из них.

6. Так как отличий в результатах особо нет (и там, и там лучшие значения (0.58, 0.86)), то мне кажется предпочтительней метод стохастического спуска, так как быстрее сходится.

Задание 8

Рассмотрим вектор весов [0, 0, 0, 0, 0, 0, 0, 1]. Это означает, что мы считаем, что мы считаем, что диабет напрямую связан с возрастом человека и никак не зависит от других признаков.