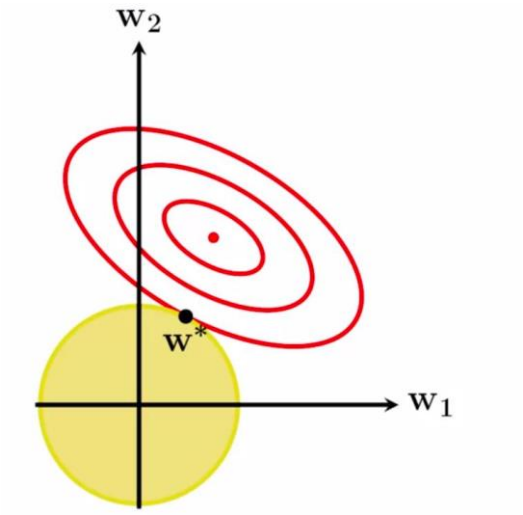
1. Если говорить о L1, L2 регуляризациях, то они штрафуют модель, основываясь на "содержимом" вектора весов. Более длинные векторы (в обеих метриках) будут увеличивать дополнительное слагаемое в целевой функции (ошибка + lambdaL1_or_L2_norm), которую мы хотим минимизировать. Без регуляризатора мы бы просто искали точку минимума функции ошибок. После же добавления регуляризатора мы требуем, чтобы решение находилось внутри некоторой круглой области с центром в 0 (ромб в случае L1) И при этом же было как можно ближе к оптимальному решению без регуляризатора. [На картинке норма L2 и линии уровня минимизируемой функции, т.е w близок к минимуму и при этом имеет сравнительно небольшую норму]



2. В ходе обучения веса модели изменяются. Скорость обучения -- параметр, влияющий на то, как сильно они изменяются каждый раз. Логика состоит в том, чтобы использовать скорость обучения для обеспечения умеренно быстрой и последовательной сходимости к оптимальному значению. Когда сеть впервые начинает обучение, ошибка, вероятно, будет большой. Более высокая скорость обучения помогает нейронной сети делать большие шаги в направлении минимума. Если она велика, эти скачки также могут быть довольно большими, и сеть будет обучаться скорее хаотично, потому что веса не сходятся постепенно к минимальной ошибке. (Можно представить эллиптический параболоид и тогда мы будем прыгать с одного "склона" чаши на другой). Если выбрать скорость обучения очень маленькой, модель будет слишком долго обучаться. Соответственно константная скорость обучения -- не самое лучшее решение, и поэтому в используемых минимизационных алгоритмах на каждом шаге сама скорость тоже изменяется.
3. Если под устойчивостью понимать получаемую величину ошибки, то безусловно mae более устойчив, ибо суммируется абсолютная величина отклонений, в то время как в mse она еще и возводится в квадрат. (Однако функции использующие модуль везде в математике стараются избегать из-за недифференцируемости в нуле, так что думаю, можно сказать, что mse чуть более универсален)
4. Отсутствует. Делается функцией np.random.shuffle(массив_данных)
5. (Потому что нам дали такой код)
6. Отправляем в модель val_data, val_targets, где предсказанные моделью значения сравниваются с истинными, давая нам собственно val_mae, val_mse.