Линейная регрессия - построение аппроксимирующей функции данных точек с помощью прямой.

В качестве тестовых данных взят пятый файл.

Аппроксимирующая прямая может быть оценена с помощью Root Mean Squared Error – оценки:



То есть длины отрезков от каждой точки до ее аппроксимированного значения возводятся в квадрат и суммируются, далее берется корень от суммы, деленной на количество этих точек.

NRMSE – нормированный RMSE – формула еще делится на разницу между максимальным и минимальным актуальным (т. е. по существующим точкам) значением y.

Learning rate – коэффициент скорости, величина шага алгоритма. Большой шаг позволяет быстрее добиться цели, но вносит риск переобучения (слишком хорошо подходим к обучающим данным, тестовые данные предсказываем плохо)

Коэффициент регуляризации также влияет на уменьшение переобучения, это штраф, накладываемый на алгоритм

Параметр кросс-валидации определяет, какую часть данных мы будем отделять для теста

**Метод наименьших квадратов** – регуляризация подбиралась перебором, для данного метода и датасета получилось значение решуляризации 94.8. При такой регуляризации минимальна оценка Test NRMSE (0.03315489446317982). График зависимости тестовой ошибки от значения регуляризации:

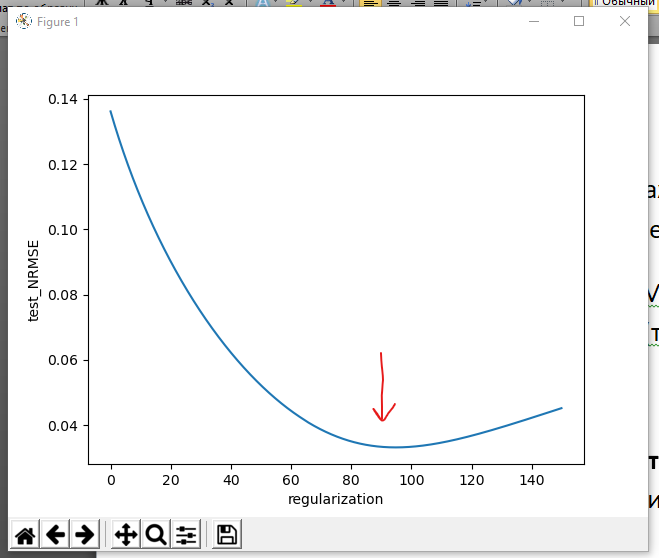
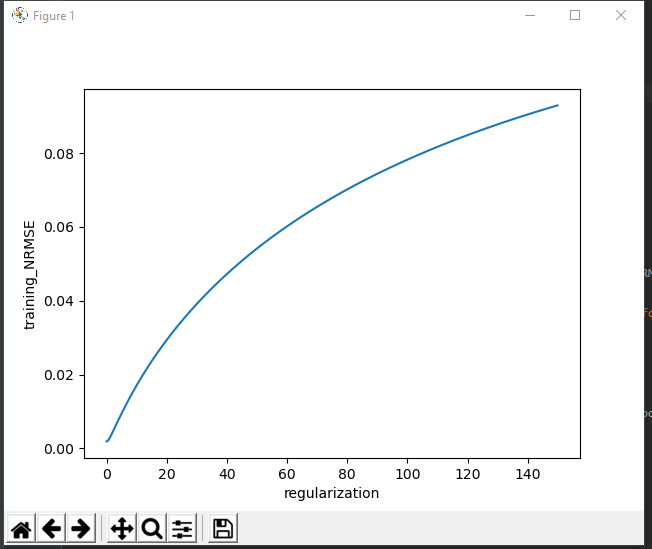
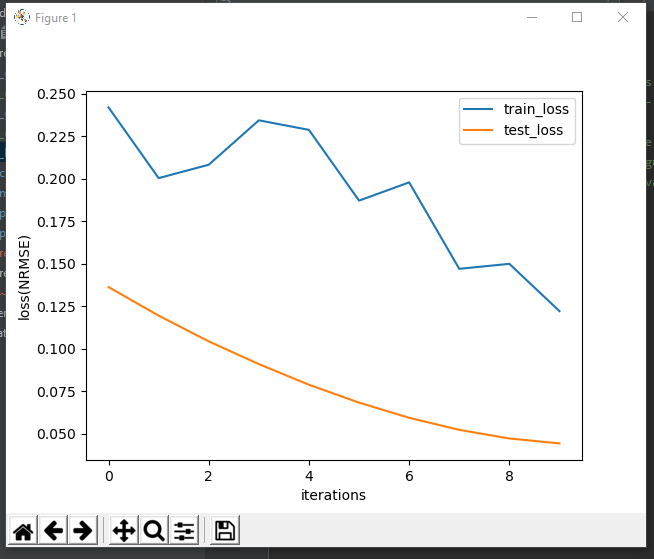


График для тренировочной ошибки:



**Градиентный спуск** – перед нами стоит задача аппроксимации неизвестной зависимости прямой линией, то есть необходимо найти такие w0, w1, что y = w0 \* x + w1 максимально хорошо аппроксимирует имеющиеся данные, то есть функция ошибки NRMSE минимальна. Она зависит от предсказанных у, а значит, от w0 и w1.

Параметрами в задаче являются learning\_rate, regularization\_strength, cross\_val\_parameter и num\_iterations. Оптимизировать нужно первые три посредством перебора всех и выбора test\_loss NRMSE оценки по предсказанию последнего запуска (где достигается максимально допустимое количество итераций) с предложенной комбинацией.



Комбинация получилась следующей:

{'last\_test\_loss': 0.04440757474423136, 'parameters': {'learning\_rate': 0.5, 'regulariz

ation\_strength': 0.001, 'cross\_val\_parameter': 50, 'num\_iterations': 10}},

параметры для расчета были предложены

{ 'learning\_rate': [0.001, 0.01, 0.5, 1, 5],

'regularization\_strength': [0.001, 0.01, 0.5, 1, 5, 10, 30],

'num\_iterations': 10,

'cross\_val\_parameter': [5, 10, 20, 50]

}.

В принципе, если необходимо сделать очень точные измерения, можно отредактировать возможные параметры исходя из результата, и запустить оптимизатор повторно. Но так как он считает долго, я этого делать не стала и ограничилась имеющимися.

Заспуск исследования зависимости функции ошибки от количества итераций для параметров {

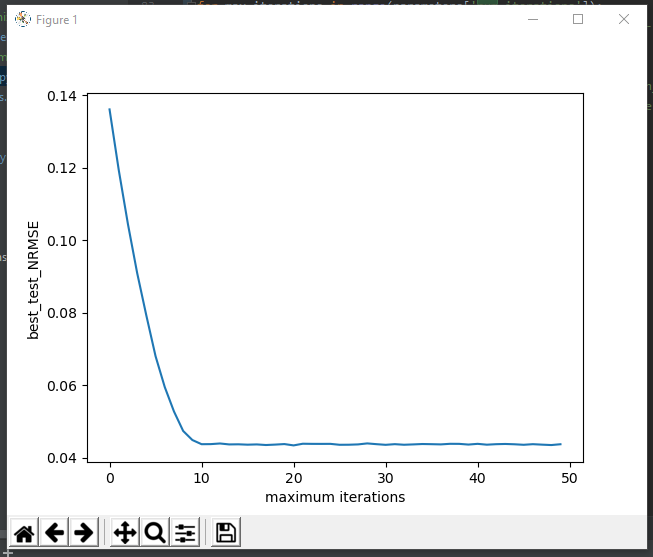
'num\_iterations': 50,

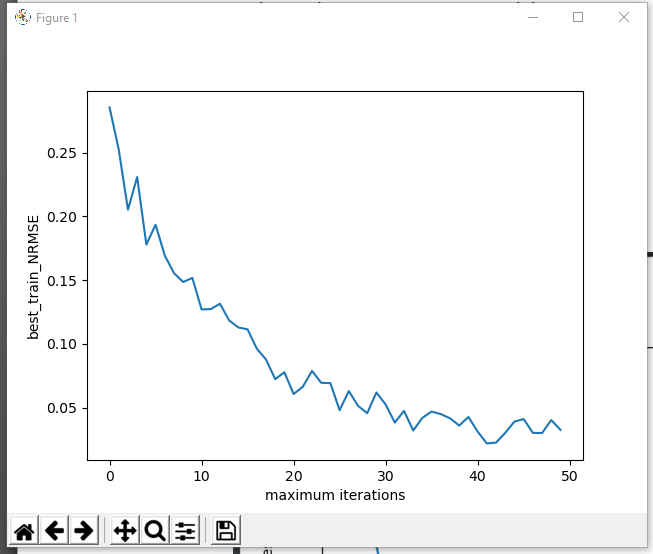
'regularization\_strength': 0.001,

'learning\_rate': 0.5,

'cross\_val\_parameter': 50

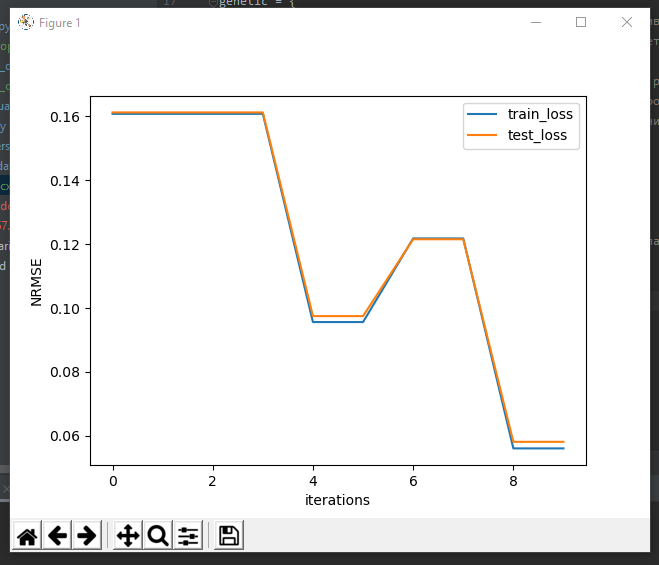
}



****

**Генетический алгоритм** – алгоритм, работающий по принципу эволюции. Есть некоторое множество индивидуумов – наборов переменных, и результатом работы будет самый приспособленный индивидуум. Начальный набор индивидуумов (т.е. популяция) будет обладать случайными данными. Fitness – степень адаптации индивидуума, его будем максимизировать. На каждой итерации каким-либо образом популяция скрещивается между собой, и в новую популяцию вносятся мутации (чтобы не было застоя старых генов, а то вдруг изначально они были плохие). Далее либо поддерживается численность на старом уровне, и алгоритм селекции отбирает лучших каким-либо образом, либо численность может постепенно расти (зависит от задачи).

Подбор лучших параметров для данного алгоритма – тоже перебор комбинаций и выбор лучшей по ошибке с тестовых данных. Это график с поиска, резкие изменения вызваны сменой параметров.

****

По итогу подобраны параметры {'mutation\_amplitude': 1, 'regularization\_strength': 1e-05, 'size\_population': 50, 'num\_generations': 10, 'parent\_proportion': 0.9, 'random\_gene\_amplitude': 1000} из возможных для выбора

genetic\_calculation = {

'size\_population': [20, 50],

'parent\_proportion': [0.1, 0.5, 0.9],

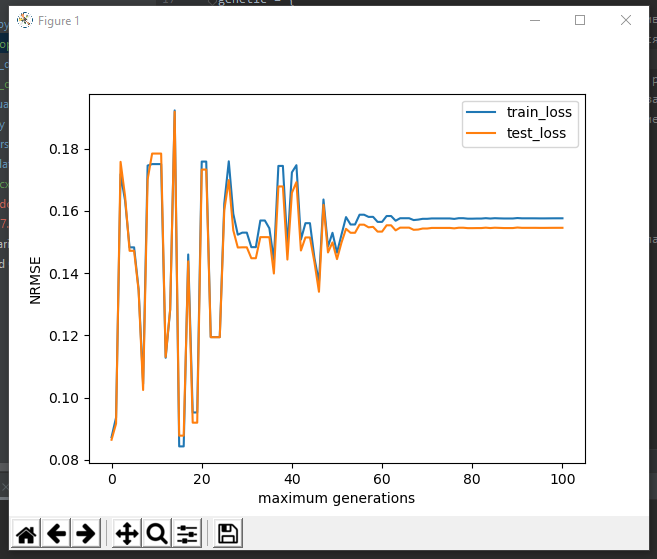
'num\_generations': 10,

'random\_gene\_amplitude': [10, 100, 1000, 10000],

'mutation\_amplitude': [0.1, 0.5, 1, 5],

'regularization\_strength': [0.00001, 0.001, 0.1]

}

****

Запуск алгоритма с подобранными параметрами на 100 поколений, на графике видно, как со временем стабилизируются параметры весов (хромосомы) и соответственно величина ошибки.

Величина NRMSE при эволюции не являлась fitness-функцией (там была лассо-поправка, зависящая от силы регуляризации), поэтому величина NRMSE в итоге не минимизирована.