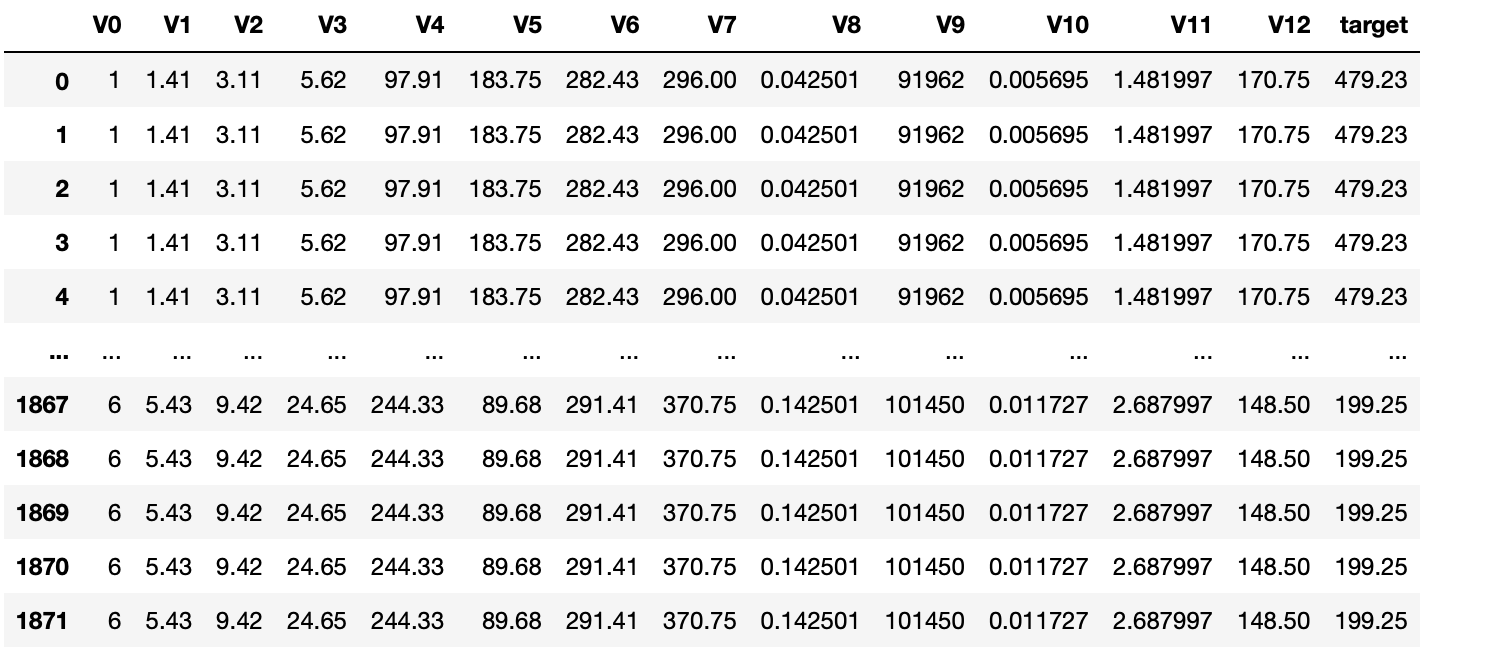
1. 数据处理

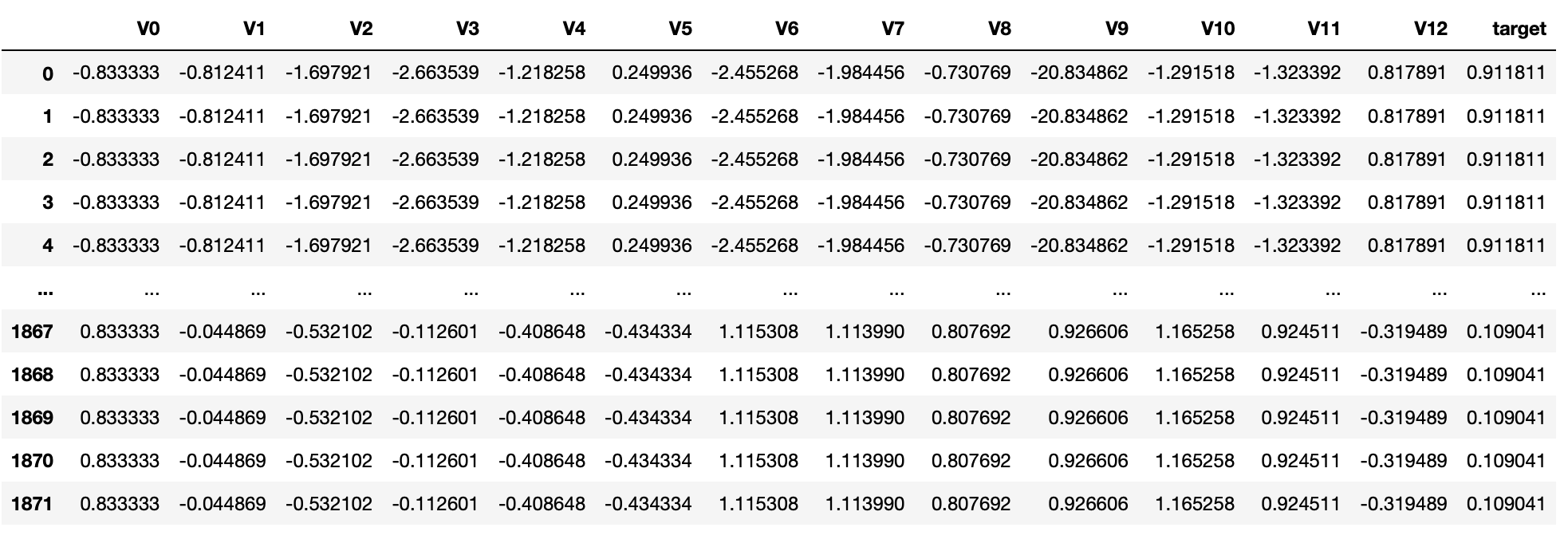
1、

首先是导入数据，进行查看



2、

归一化：



3、

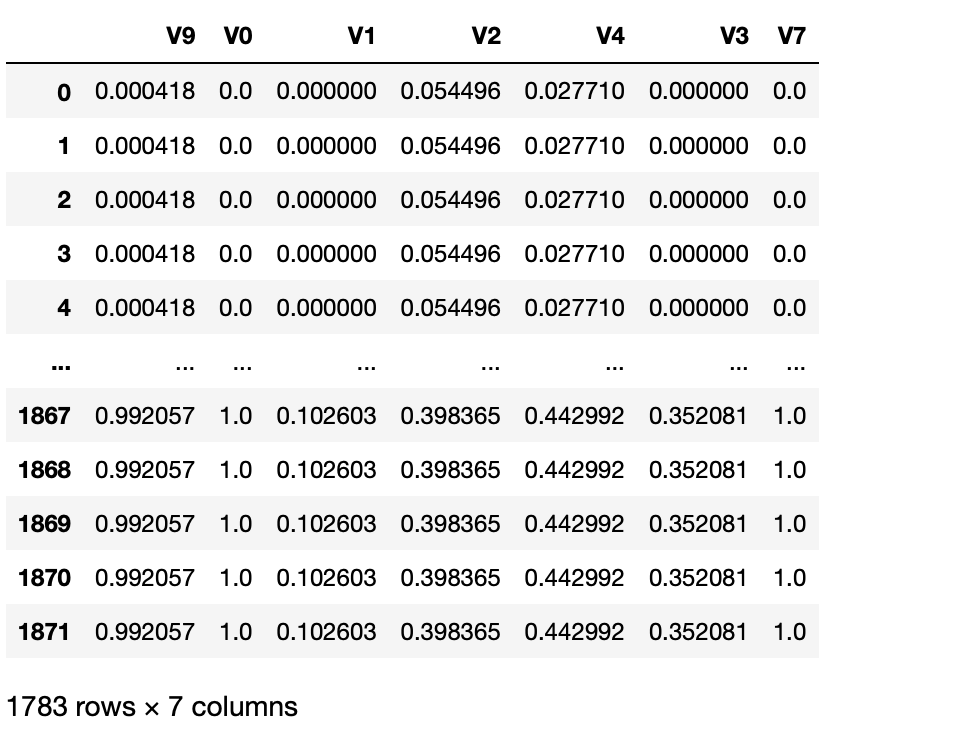
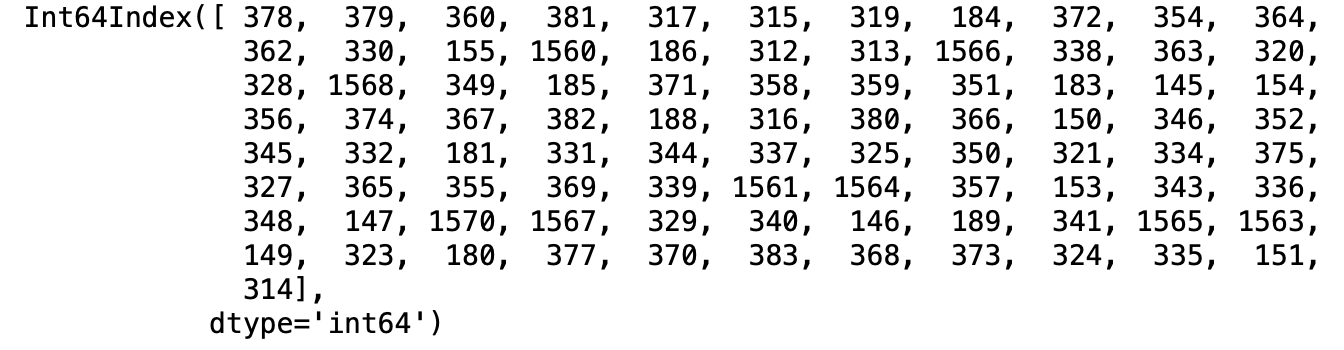
接下来查看箱式图，可用于异常值检测

4、

接下来对数据的相关性分析，可以发现特征变量和目标变量及特征变量之间的关系

5、过滤异常值

此处过滤异常样本的基本思想为：利用岭回归先对train 数据进行建模，建模后的获取 x\_train的预测值y\_train\_predict，若真实值与预测值相减的绝对值 |y\_train-y\_train\_predict| >α\*y\_train.std()，则判定该样本为异常值；此处α参数经过多次测试，取了0.8 较为合适；获取行索引，异常值过滤

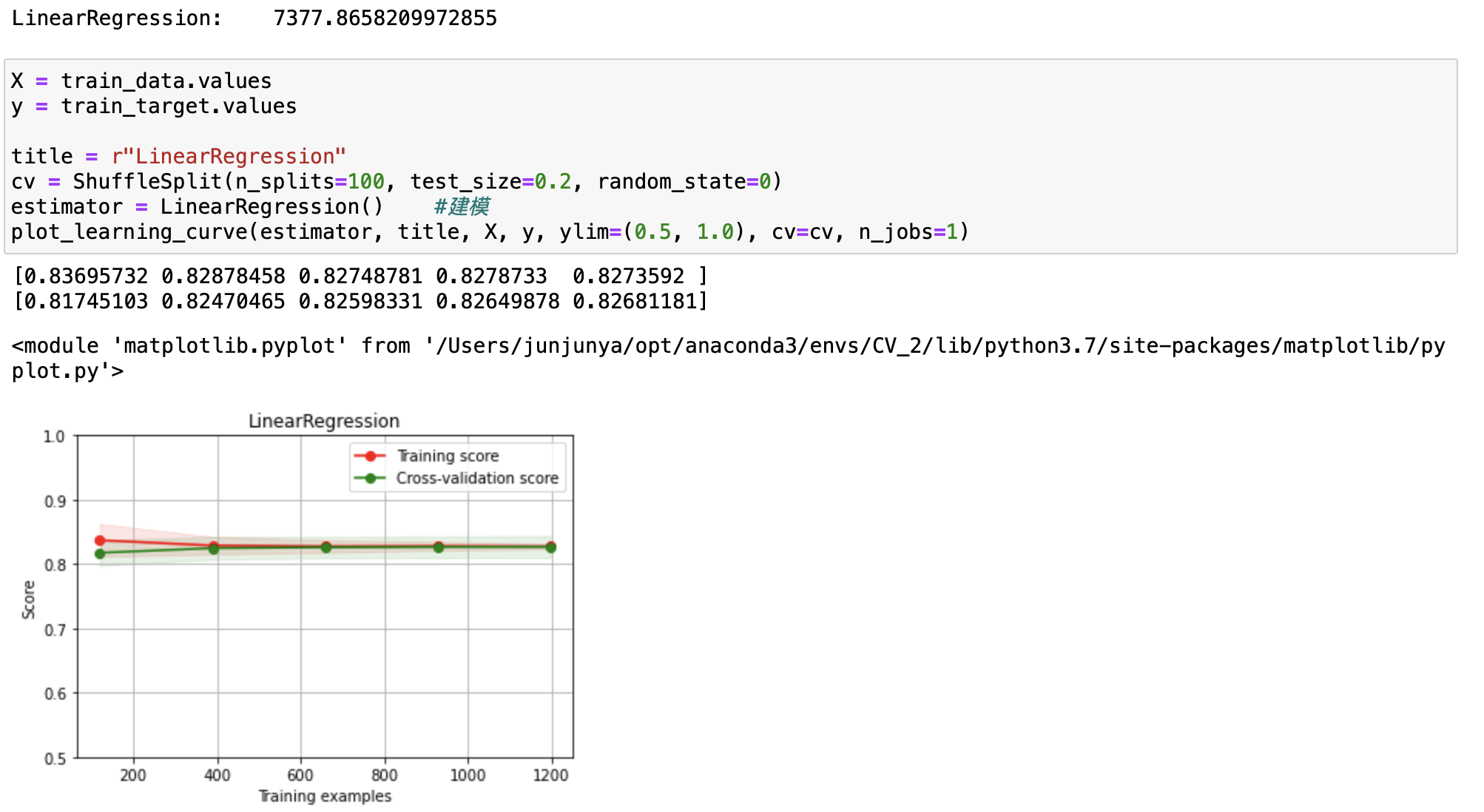


1. 构建模型
2. 机器学习
3. 因为我们数据不是很充足，故使用交叉验证法来训练优化选择模型

评价指标：交叉验证(Cross Validation)用来验证分类器的性能一种统计分析方法，基本思想是把在某种意义下降原始数据(dataset)进行分组，一部分用来为训练集(train set),另一部分做为验证集(validation set)。利用训练集训练分类器，然后利用验证集验证模型，记录最后的分类准确率为此分类器的性能指标。

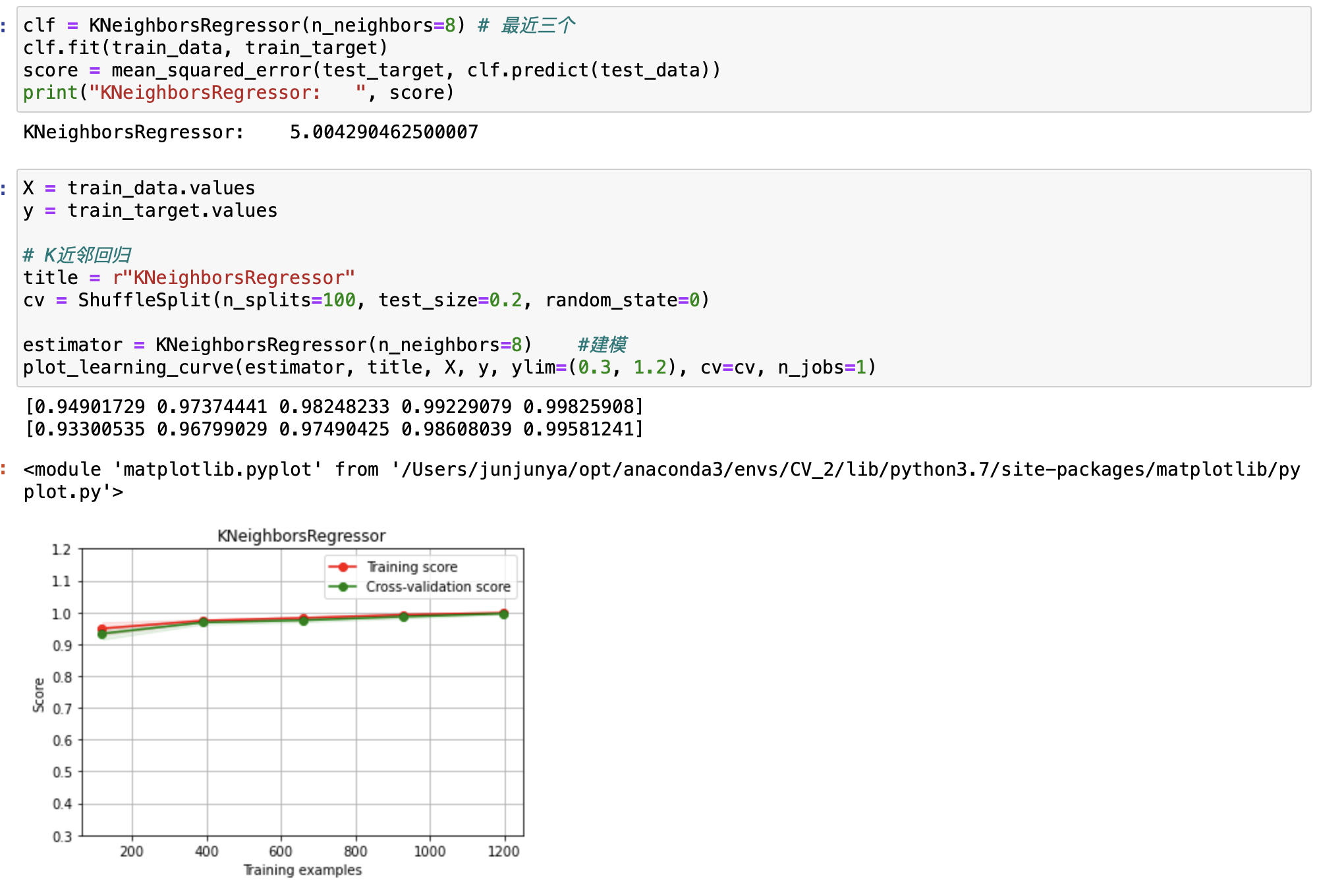
线性回归

利用多个自变量估计因变量，从而解释和预测因变量的值



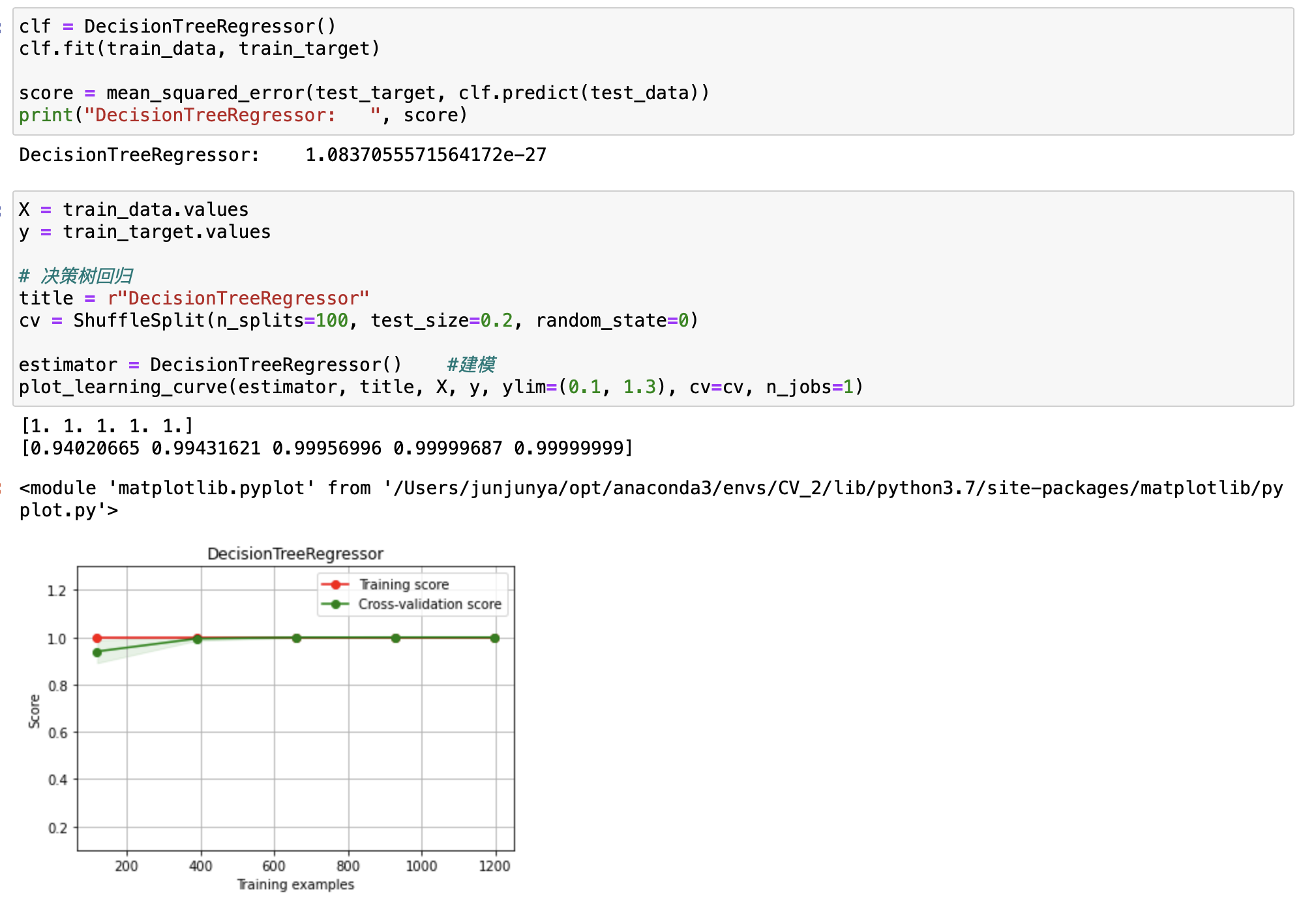
k邻近回归

通过找出某个样本的k个最近邻居，将这些邻居的某个属性的平均值赋给该样本，就可以得到该样本对应属性的值，两个点的公式用欧拉距离公式



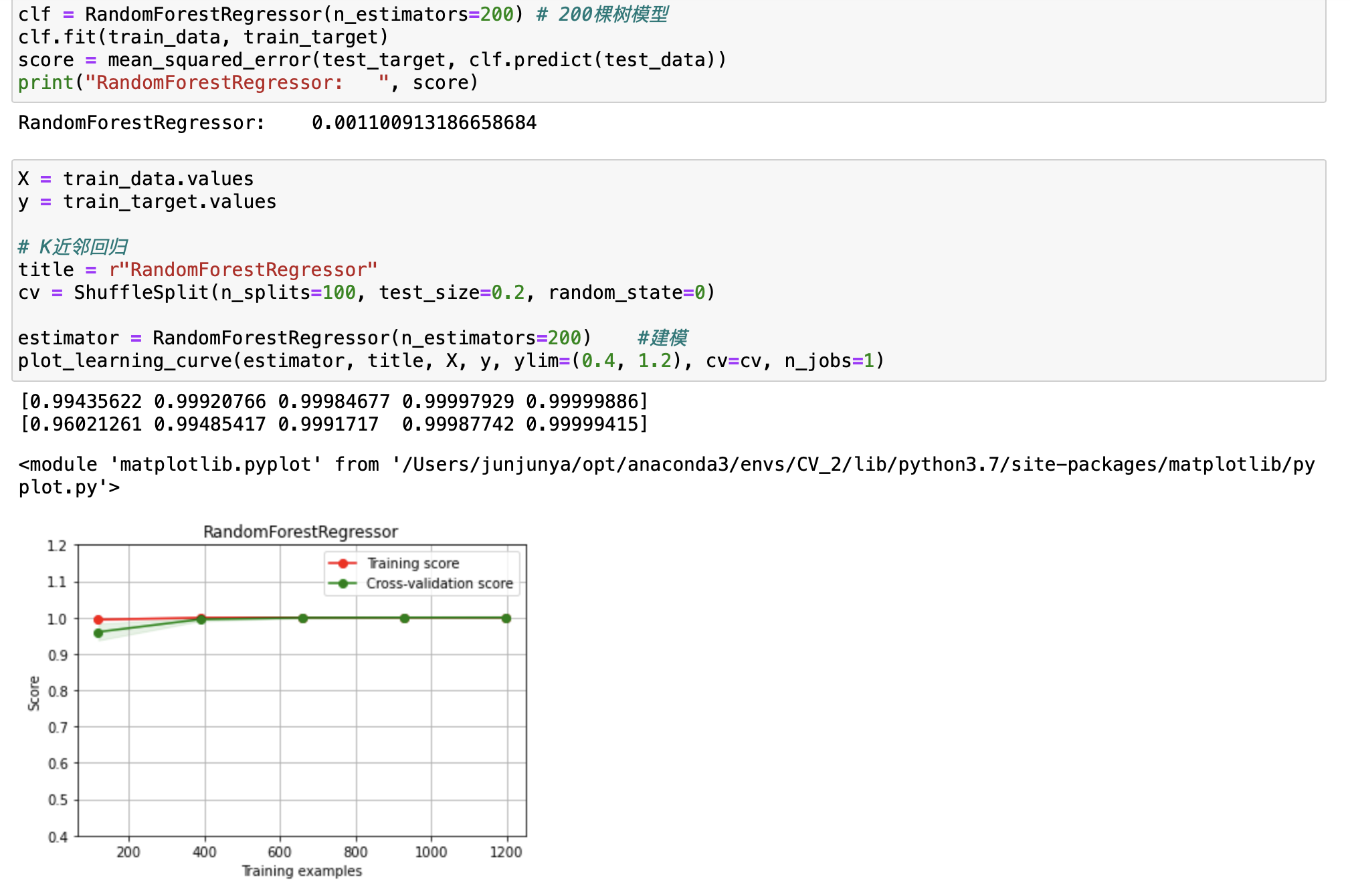
决策树回归

根据一定准则，将一个空间划分为若干个子空间，然后利用这些子空间内所有的点的信息表示这个子空间的值，对于测试数据，只要按照特征将其归到某个子空间，便可得到对应空间的输出值，这里为了得到预测值，一旦有样本落入某一区域，可以用这些划分区域的均值或者中位数代表预测值。



随机森林回归

这是通过集成学习的思想将多棵树集成的一种算法，基本单元是决策树，在回归问题，决策树输出所有决策树的平均值



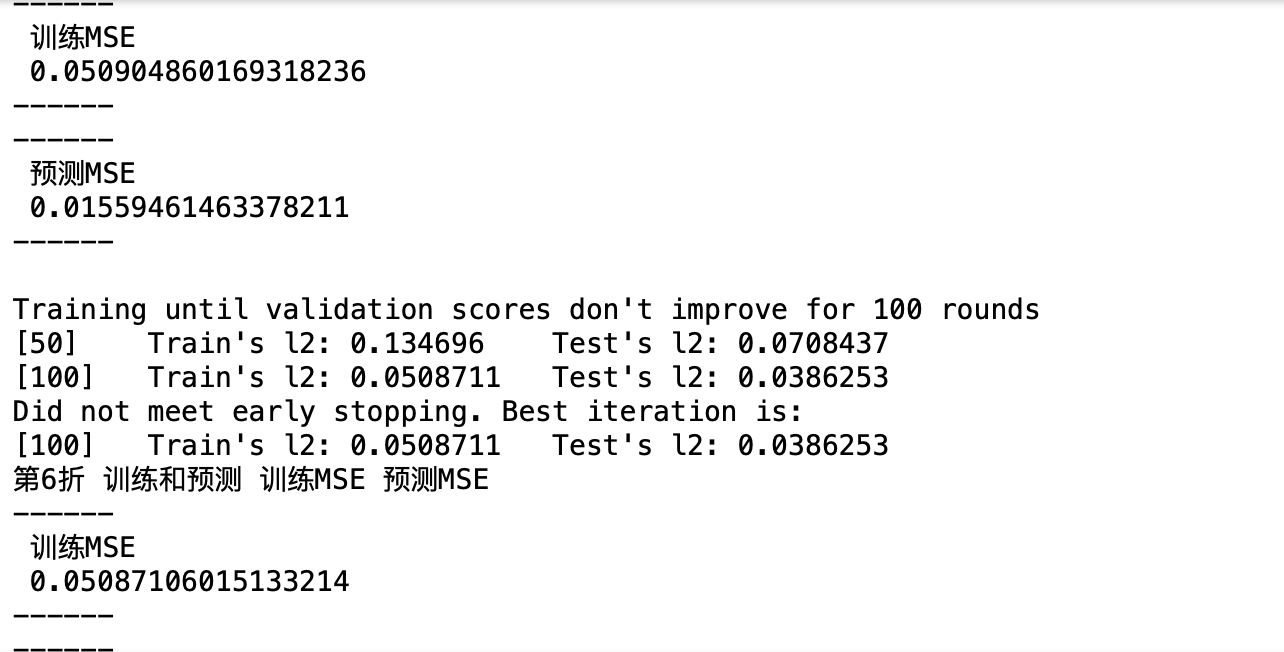
优点：具有极好的准确略，能够有效的运行在大的数据集上，能够处理具有高纬度特征的输入样本，而且不需要降维，能够评估各个特征在分类问题上的重要性，在生成过程中，能够获取内部生成误差的一种无偏估计，对于缺省问题也有好的结果

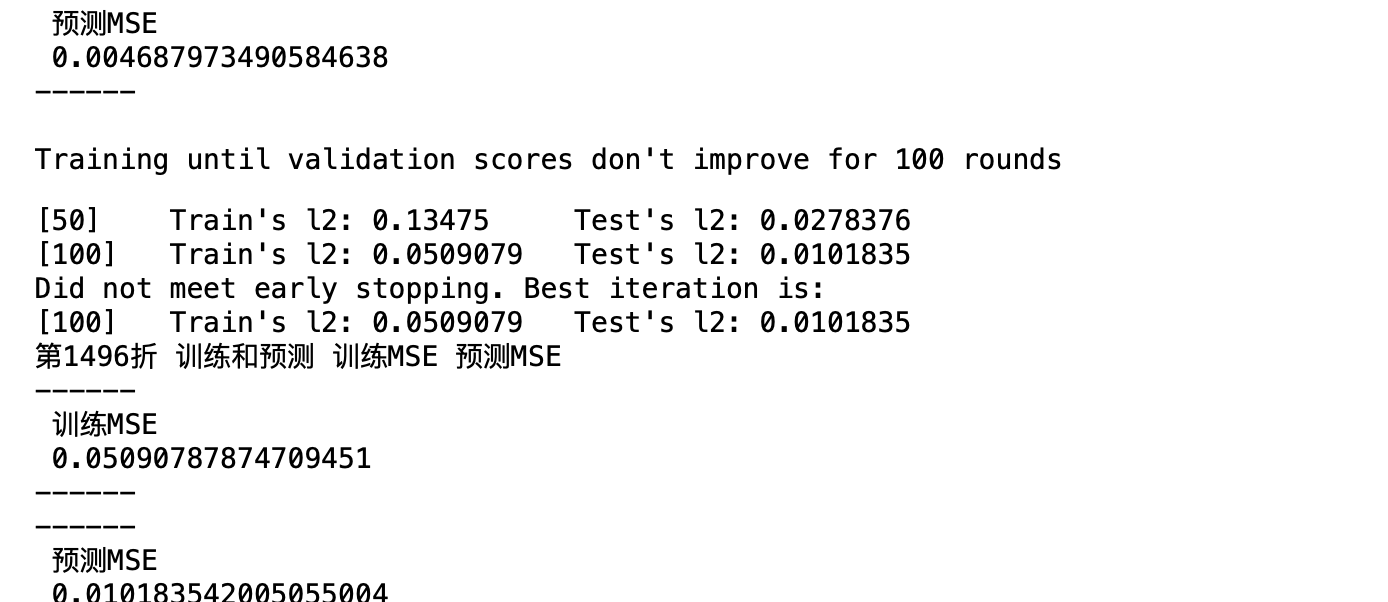
LightGbm模型

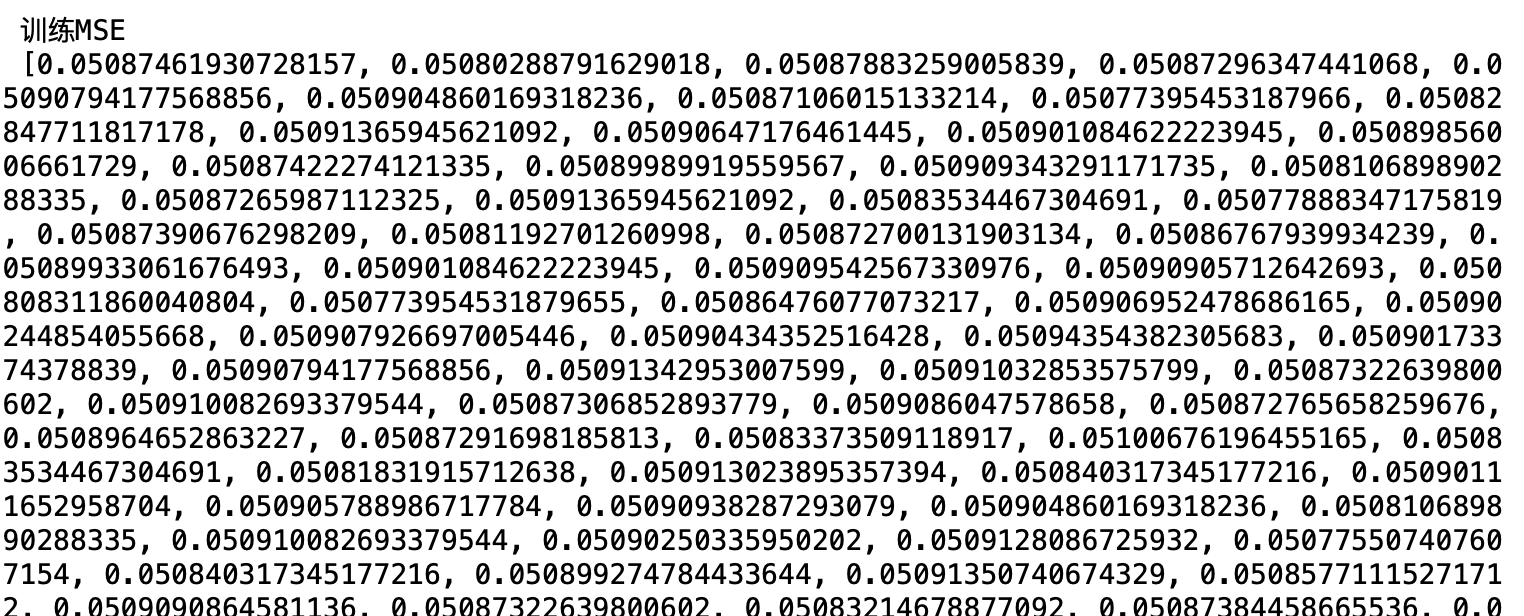
支持高效率的并行训练，具有更快的训练速度，更低的内存消耗，更好的准确略、分布式支持，可以快速处理海量数据等特征



线下验证

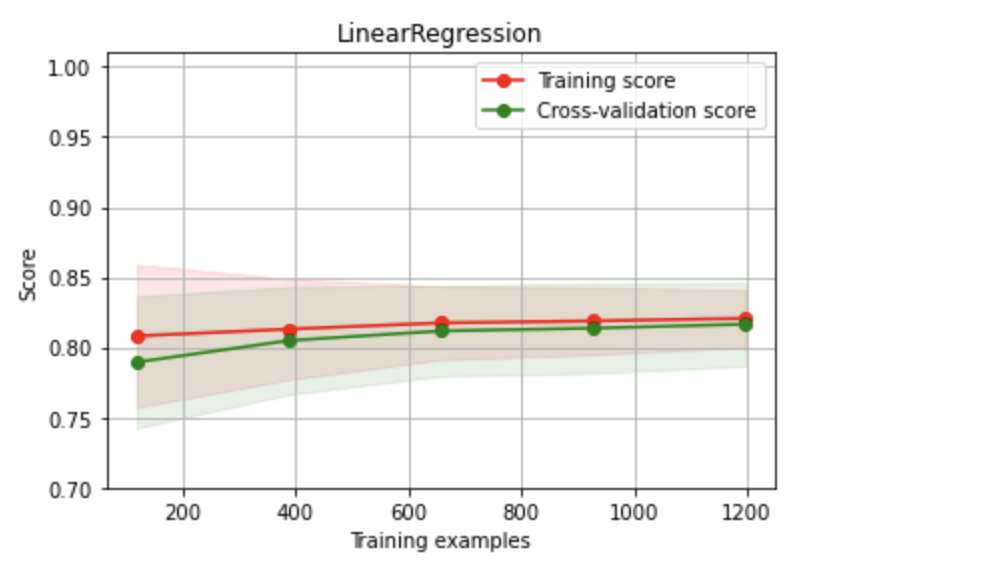




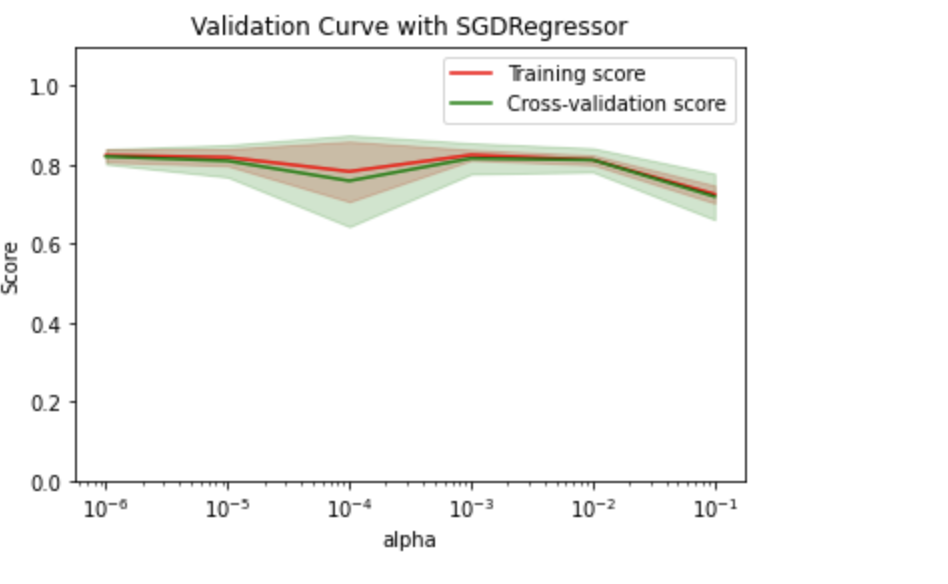


验证结果

线性回归



SGD回归



深度学习

一、读取数据，数据处理

这里我们使用归一化，为了让统一的步长更加合适。特征输入归一化后，不同参数输出的Loss是一个比较规整的曲线，朝哪个方向都一样，学习率可以设置成统一的值 ；特征输入未归一化时，不同特征对应的参数所需的步长不一致，尺度较大的参数w1需要大步长，尺寸较小的参数w2需要小步长，导致无法设置统一的学习率。



1. 网络结构与训练配置

即实现模型“前向计算”（从输入到输出）的过程。深度学习的一个神经元，就可以表示线性回归，去掉激活函数（非线性）即可。以类的方式实现网络结构,可以生成多个模型实例，类成员变量有w,b可以初始化变量



模型设计完成后，需要通过训练配置寻找模型的最优值，即通过损失函数来衡量模型的好坏。

同时计算多个样本的损失函数

在本例中，我们需要有某种指标来衡量预测值Z跟真实值Y之间的差距。对于回归问题，最常采用的衡量方法是使用均方误差作为评价模型好坏的指标，即：Loss=(Y-Z)\*\*2。

Loss就是损失函数，它是衡量模型好坏的指标。在回归问题中，均方误差是一种比较常见的形式，分类问题中通常会采用交叉熵作为损失函数。  
因为均方误差曲线呈现出“圆滑”的坡度。均方误差表现的“圆滑”的坡度有两个好处：曲线的最低点是可导的。越接近最低点，曲线的坡度逐渐放缓，有助于通过当前的梯度来判断接近最低点的程度（是否需要逐渐减少步长，以免错过最低点）。

三、训练过程

接下来介绍如何求解参数w和b的数值，这个过程也称为模型训练过程。  
目标是让定义的损失函数Loss尽可能的小，也就是说找到一个参数解w和b，使得损失函数取得极小值。

模型训练、优化算法、寻解算法

求解损失函数达到极小值时的参数值：梯度下降法

总结：（前向计算+后向传播）  
可分为以下四个步骤，且顺序不能乱！  
1.前向计算：得到模型的输出，即预测值z  
2.损失函数计算：预测值z和真实值y比较计算，得到Loss  
3.梯度计算：z,y,x都需要，才能计算梯度  
4.梯度更新：根据梯度，更新参数



使用numpy进行梯度计算

一方面可以扩展参数的维度，代替for循环来计算1个样本对从w0到w12的所有参数的梯度。，另一方面可以扩展样本的维度，代替for循环来计算样本0到样本403对参数的梯度。

梯度更新（确定损失函数更小的点）

下面我们开始研究更新梯度的方法。首先沿着梯度的反方向移动一小步，找到下一个点P1，观察损失函数的变化。

可以发现沿着梯度反方向走一小步，下一个点的损失函数的确减少了

随机梯度下降法（Stochastic Gradient Descent，SGD）

由于参数每次只沿着梯度反方向更新一点点，因此方向并不需要那么精确。一个合理的解决方案是每次从总的数据集中随机抽取出小部分数据来代表整体，基于这部分数据计算梯度和损失来更新参数，这种方法被称作随机梯度下降法，核心概念如下：

mini-batch：每次迭代时抽取出来的一小批数据被称为一个mini-batch。

batch\_size：一个mini-batch所包含的样本数目称为batch\_size。

epoch：当程序迭代的时候，按mini-batch逐渐抽取出样本，当把整个数据集都遍历到了的时候，则完成了一轮训练，也叫一个epoch。启动训练时，可以将训练的轮数num\_epochs和batch\_size作为参数传入。

将train\_data分成大小为batch\_size的多个mini\_batch，如下代码所示：将train\_data分成 404/10+1=41 个 mini\_batch，其中前40个mini\_batch，每个均含有10个样本，最后一个mini\_batch只含有4个样本。

上面是按顺序读取mini\_batch，而SGD里面是随机抽取一部分样本代表总体。为了实现随机抽样的效果，我们先将train\_data里面的样本顺序随机打乱，然后再抽取mini\_batch。随机打乱样本顺序，需要用到np.random.shuffle函数。  
故我们将将每个随机抽取的mini-batch数据输入到模型中用于参数训练。训练过程的核心是两层循环：  
1.第一层循环，代表整个样本集合被训练遍历的次数，称为“epoch”。  
2.第二层循环，代表每次遍历时，样本集合被拆分成的多个批次，需要全部执行训练，称为“iter (iteration)”。

四、结果

