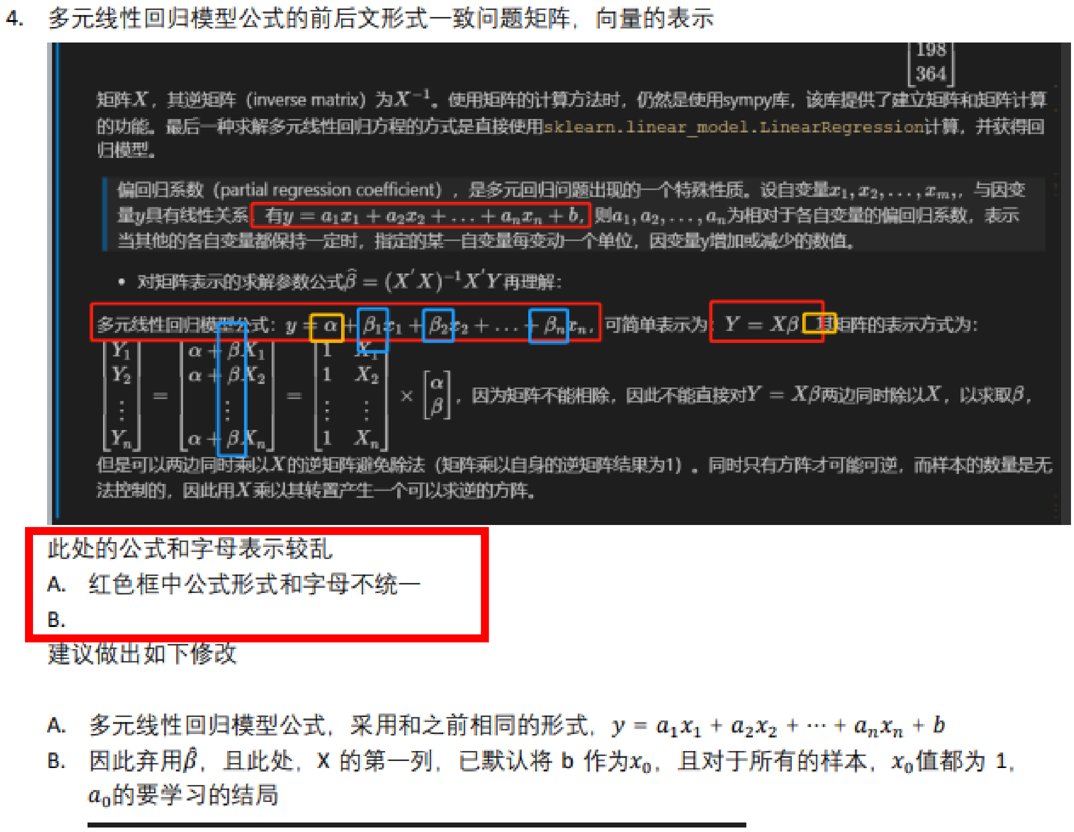
代码内容审核建议 5 （2.1.5）

1. 之前代码与内容审核中的勘误

文件名：代码建议 4.docx 【代码建议四 （章节 2.1.4.3 多元线性回归开始至 2.1.4 结尾)】

A. 列表列举和公式字母表示问题，错别字问题（结局→截距）



4. 多元线性回归模型公式的前后文形式一致问题矩阵，向量的表示

198
364

矩阵X，其逆矩阵（inverse matrix）为X^{-1}。使用矩阵的计算方法时，仍然是使用sympy库，该库提供了建立矩阵和矩阵计算的功能。最后一种求解多元线性回归方程的方式是直接使用`sklearn.linear_model.LinearRegression`计算，并获得回归模型。

> 偏回归系数（partial regression coefficient），是多元回归问题出现的一个特殊性质。设自变量$x_1, x_2, \ldots, x_m$，与因变量y具有线性关系，有$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$，则$a_1, a_2, \ldots, a_n$为相对于各自变量的偏回归系数，表示当其他的各自变量都保持一定时，指定的某一自变量每变动一个单位，因变量y增加或减少的数值。

- 对矩阵表示的求解参数公式$\hat{\beta} = (X'X)^{-1}X'Y$再理解：

多元线性回归模型公式：$y = \alpha + \beta_1x_1 + \beta_2x_2 + \ldots + \beta_nx_n$，可简单表示为：$Y = X\beta$，其矩阵的表示方式为：

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \alpha + \beta X_1 \\ \alpha + \beta X_2 \\ \vdots \\ \alpha + \beta X_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \times \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

，因为矩阵不能相除，因此不能直接对$Y = X\beta$两边同时除以X，以求取β，但是可以两边同时乘以X的逆矩阵避免除法（矩阵乘以自身的逆矩阵结果为1）。同时只有方阵才可能可逆，而样本的数量是无法控制的，因此用X乘以其转置产生一个可以求逆的方阵。

此处的公式和字母表示较乱

A. 红色框中公式形式和字母不统一

B.

建议做出如下修改

A. 多元线性回归模型公式，采用和之前相同的形式，$y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$

B. 因此弃用$\hat{\beta}$，且此处，X 的第一列，已默认将 b 作为x_0，且对于所有的样本，x_0值都为 1，a_0的要学习的结局

此处，A，B 选项不必要，可做如下修改

此处的公式和字母表示较乱，红色框中公式形式和字母不统一，蓝色框中β和β_n等定义不够明确，建议做出如下修改

A. 多元线性回归模型公式，采用和之前相同的形式，$y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$

B. 因此弃用$\hat{\beta}$，且此处，X的第一列，已默认将b作为x_0，且对于所有的样本，x_0值都为 1，要学习的a_0则表示线性回归中的**截距**

B. 错别字问题（表面→避免）

较为合理，暂不做修改，但是保留对简单线性回归章节中的代码截图，作为对比】

简单线性回归章节中的代码修改	此章节注释，暂不修改

`# 对 a 和 b 的微分公式构造二元一次方程组，`
`# 即 diff_S_residual_a = 0, diff_S_residual_a = 0`
`Eq_residual_a=Eq(diff_S_residual_a,0)`
`Eq_residual_b=Eq(diff_S_residual_b,0)`
`# 设置二元一次方程组中的变量为 a 和 b，并求解`
`slop_intercept=solve((Eq_residual_a,Eq_residual_b),(a,b))`
	`diff_SSres_a1=diff(SS_res,a1)`
	`diff_SSres_a2=diff(SS_res,a2)`
	`diff_SSres_b=diff(SS_res,b)`
	`Eq_residual_a1=Eq(diff_SSres_a1,0)`
	`Eq_residual_a2=Eq(diff_SSres_a2,0)`
	`Eq_residual_b=Eq(diff_SSres_b,0)`
	`slop_intercept=solve((Eq_residual_a1,Eq_residual_a2,Eq_residual_b),(a1,a2,b))`

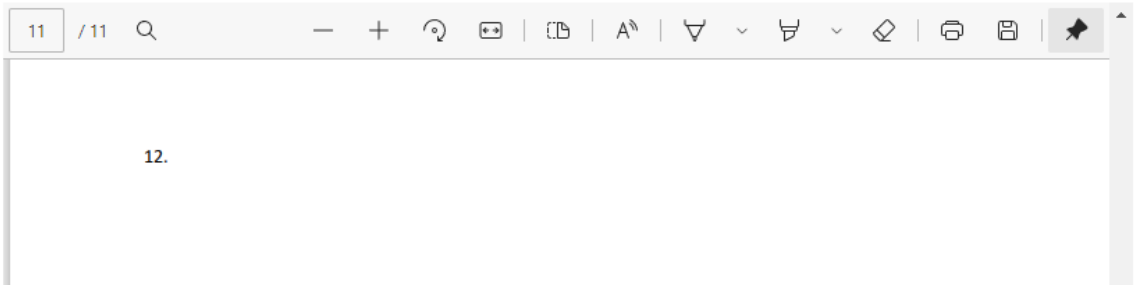
# TypeError: 'NoneType' object is not subscriptable	修改后，可以表面出现这个 error

`#B - 使用矩阵（基于 sympy）求解多元回归方程`
`if 'one' not in storeInfo_df.columns:`
`X_m=Matrix(storeInfo_df.insert(loc=1,column='one',value=1)[['one','area','distance_to_nearestStation']])`
`# TypeError: 'NoneType' object is not subscriptable`
`else:`
`X_m=Matrix(storeInfo_df[['one','area','distance_to_nearestStation']])`
	`## MODIFICATION`
	`if 'one' not in storeInfo_df.columns:`
	`storeInfo_df.insert(loc=1, column='one', value=1)`
	`X_m = Matrix(storeInfo_df[['area', 'distance_to_nearestStation', 'one']])`
	`## 将 one 这一列，放在最后面，与正文文字描述相对应`

文件名：代码建议 3.docx

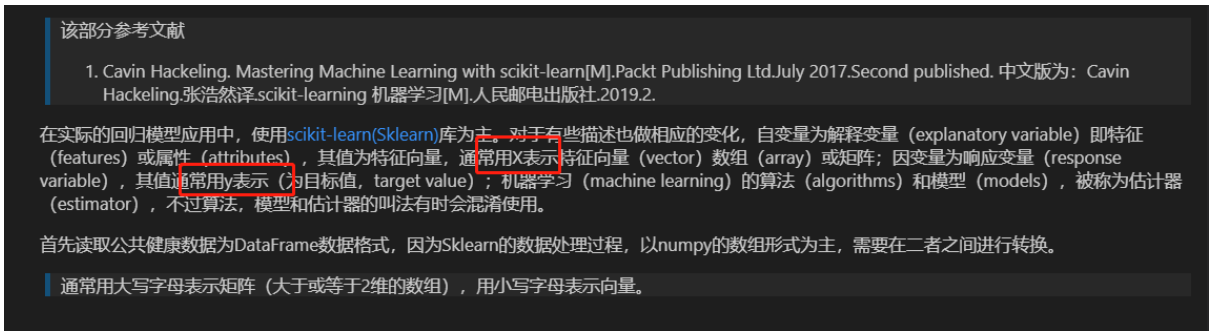
第 11 页，只有空白的序号 12，可以自行删除该序号和空白页

3. 代码建议3

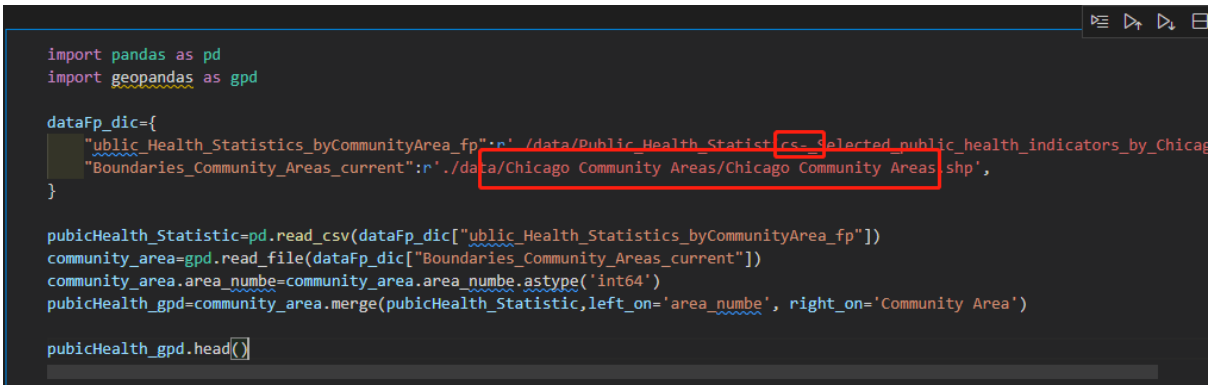


2. 以前提及过的问题

a. 表示为向量或者矩阵的字母，需要采用斜体格式，如下图中的 X 和 y。

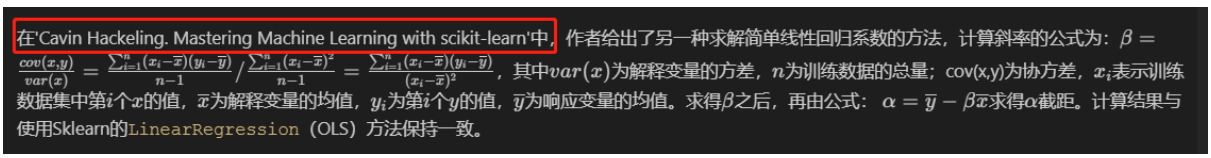


b. 文件夹路径不要有空格，最好采用驼峰命名法或者下划线连接单词



```
import pandas as pd
import geopandas as gpd

dataFp_dic={
    "ublic_Health_Statistics_byCommunityArea_fp":r'./data/Public_Health_Statistics-_Selected_public_health_indicators_by_Chicag
    "Boundaries_Community_Areas_current":r'./data/Chicago Community Areas/Chicago Community Areas.shp',
}

pubicHealth_Statistic=pd.read_csv(dataFp_dic["ublic_Health_Statistics_byCommunityArea_fp"])
community_area=gpd.read_file(dataFp_dic["Boundaries_Community_Areas_current"])
community_area.area_numbe=community_area.area_numbe.astype('int64')
pubicHealth_gpd=community_area.merge(pubicHealth_Statistic,left_on='area_numbe', right_on='Community Area')

pubicHealth_gpd.head()
```

c. 引用格式



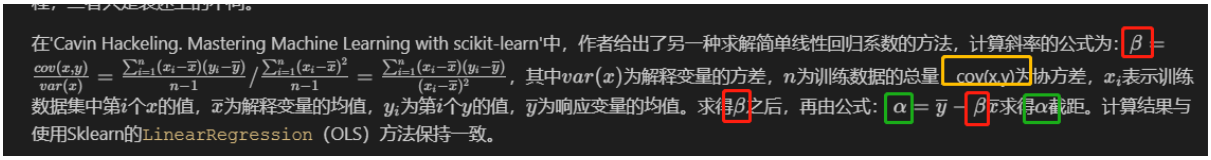
d. 字母表示，最好采用和前面章节对应的方法

Conv(x,y) 没有斜体，

α和β的值可以互换一下，因为数据中已经出现x和y，但是可以采用**加粗**和**不加粗**的x和y来表示，即

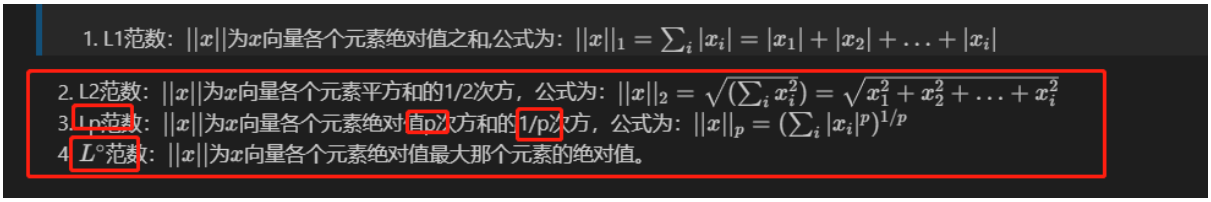
对于一元线性回归方程$y = ax + b$，x和y是回归公式，是标量，

而对于下文中提及的公式，里面的x和y表示数据， x和y都是向量



在'Cavin Hackeling. Mastering Machine Learning with scikit-learn'中，作者给出了另一种求解简单线性回归系数的方法，计算斜率的公式为：$\beta = \frac{cov(x,y)}{var(x)} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{n-1} / \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(x_i-\bar{x})^2}$，其中$var(x)$为解释变量的方差，$n$为训练数据的总量，cov(x,y)为协方差，$x_i$表示训练数据集中第$i$个$x$的值，$\bar{x}$为解释变量的均值，$y_i$为第$i$个$y$的值，$\bar{y}$为响应变量的均值。求得$\beta$之后，再由公式：$\alpha = \bar{y} - \beta\bar{x}$求得$\alpha$截距。计算结果与使用Sklearn的`LinearRegression`（OLS）方法保持一致。

e. 列举时的格式问题



1. L1范数：$||x||$为x向量各个元素绝对值之和,公式为：$||x||_1 = \sum_i |x_i| = |x_1| + |x_2| + \ldots + |x_i|$
2. L2范数：$||x||$为x向量各个元素平方和的1/2次方，公式为：$||x||_2 = \sqrt{(\sum_i x_i^2)} = \sqrt{x_1^2 + x_2^2 + \ldots + x_i^2}$
3. Lp范数：$||x||$为x向量各个元素绝对值p次方和的1/p次方，公式为：$||x||_p = (\sum_i |x_i|^p)^{1/p}$
4. L^∞范数：$||x||$为x向量各个元素绝对值最大那个元素的绝对值。

且，L1，L2，等范数，通常表示时，有特殊的表示方法，如

Feature selection, L_1 vs. L_2 regularization, and rotational invariance

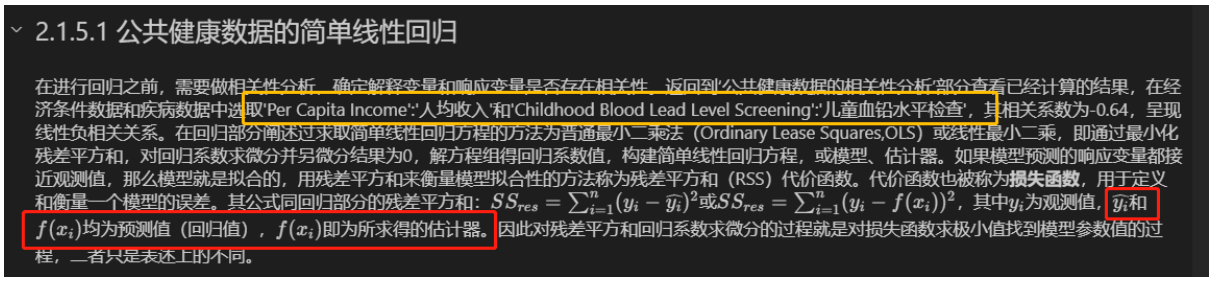
---

Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

354.pdf (icml.cc)【 https://icml.cc/Conferences/2004/proceedings/papers/354.pdf 】

3. 对数据选取列名表示方法的建议

首先，章节名“2.1.5 回归公共健康数据与梯度下降法”可以修改为“2.1.5 公共健康数据的线性回归与梯度下降法”



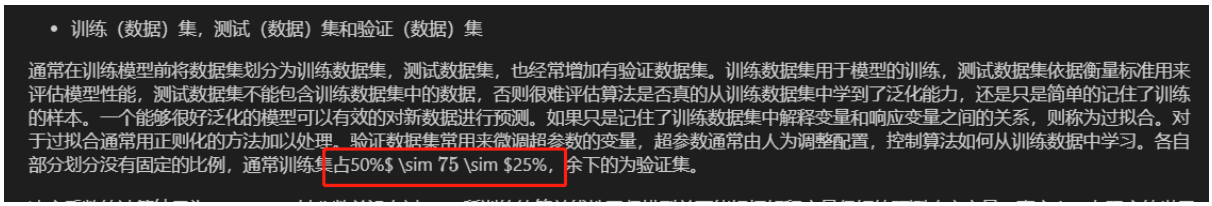
2.1.5.1 公共健康数据的简单线性回归

在进行回归之前，需要做相关性分析，确定解释变量和响应变量是否存在相关性。返回到'公共健康数据的相关性分析'部分查看已经计算的结果，在经济条件数据和疾病数据中选取'Per Capita Income':'人均收入'和'Childhood Blood Lead Level Screening':'儿童血铅水平检查'，其相关系数为-0.64，呈现线性负相关关系。在回归部分阐述过求取简单线性回归方程的方法为普通最小二乘法（Ordinary Lease Squares,OLS）或线性最小二乘，即通过最小化残差平方和，对回归系数求微分并另微分结果为0，解方程组得回归系数值，构建简单线性回归方程，或模型、估计器。如果模型预测的响应变量都接近观测值，那么模型就是拟合的，用残差平方和来衡量模型拟合性的方法称为残差平方和（RSS）代价函数。代价函数也被称为**损失函数**，用于定义和衡量一个模型的误差。其公式同回归部分的残差平方和：$SS_{res} = \sum_{i=1}^{n}(y_i - \bar{y_i})^2$或$SS_{res} = \sum_{i=1}^{n}(y_i - f(x_i))^2$，其中$y_i$为观测值，$\bar{y_i}$和$f(x_i)$均为预测值（回归值），$f(x_i)$即为所求得的估计器。因此对残差平方和回归系数求微分的过程就是对损失函数求极小值找到模型参数值的过程，二者只是表述上的不同。

上文中已经用字典形式，表示了各个列的英文名和中文翻译，此处可以直接提及英文名，且英文名最好采用双引号，然后后面可以用括号来翻译中如，

“Per Capita Income”（人均收入）

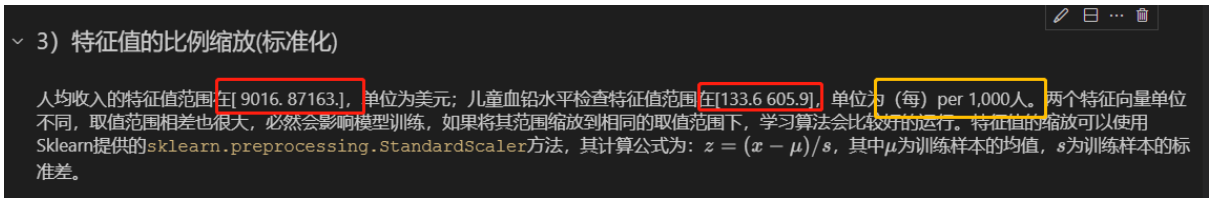
4. 发现一处问题，没有看懂要表达的意思



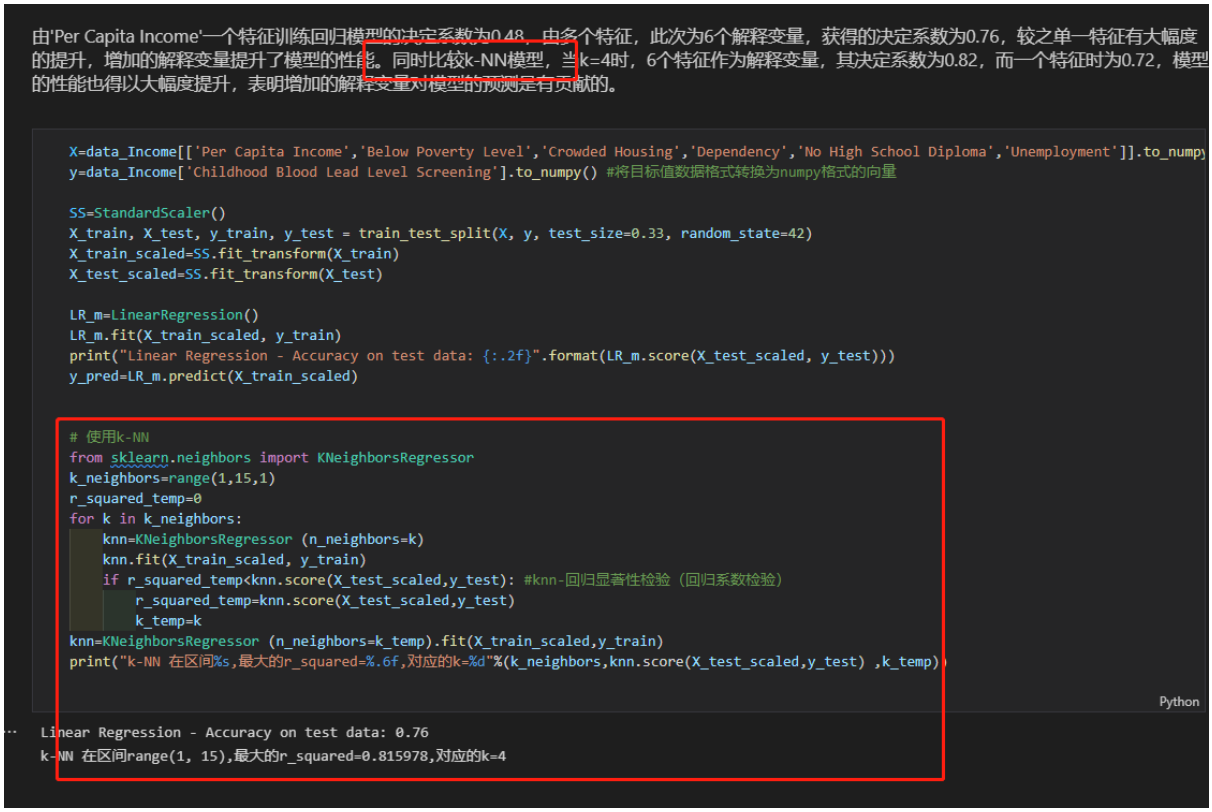
- 训练（数据）集，测试（数据）集和验证（数据）集

通常在训练模型前将数据集划分为训练数据集，测试数据集，也经常增加有验证数据集。训练数据集用于模型的训练，测试数据集依据衡量标准用来评估模型性能，测试数据集不能包含训练数据集中的数据，否则很难评估算法是否真的从训练数据集中学到了泛化能力，还是只是简单的记住了训练的样本。一个能够很好泛化的模型可以有效的对新数据进行预测。如果只是记住了训练数据集中解释变量和响应变量之间的关系，则称为过拟合。对于过拟合通常用正则化的方法加以处理。验证数据集常用来微调超参数的变量，超参数通常由人为调整配置，控制算法如何从训练数据中学习。各自部分划分没有固定的比例，通常训练集占50%$ \sim 75 \sim $25%，余下的为验证集。

5. 特征值范围，可以采用数学里定义域的表示方法：[min, max],即，中括号内最小值最大值之间，使用逗号隔开
   但是后面黄色方框里“单位为（每）per 1,000 人”，觉得这个表达让人费解



6. 提到 KNN，但是没前文并没有介绍
   建议补充介绍，链接，书籍，或者网站



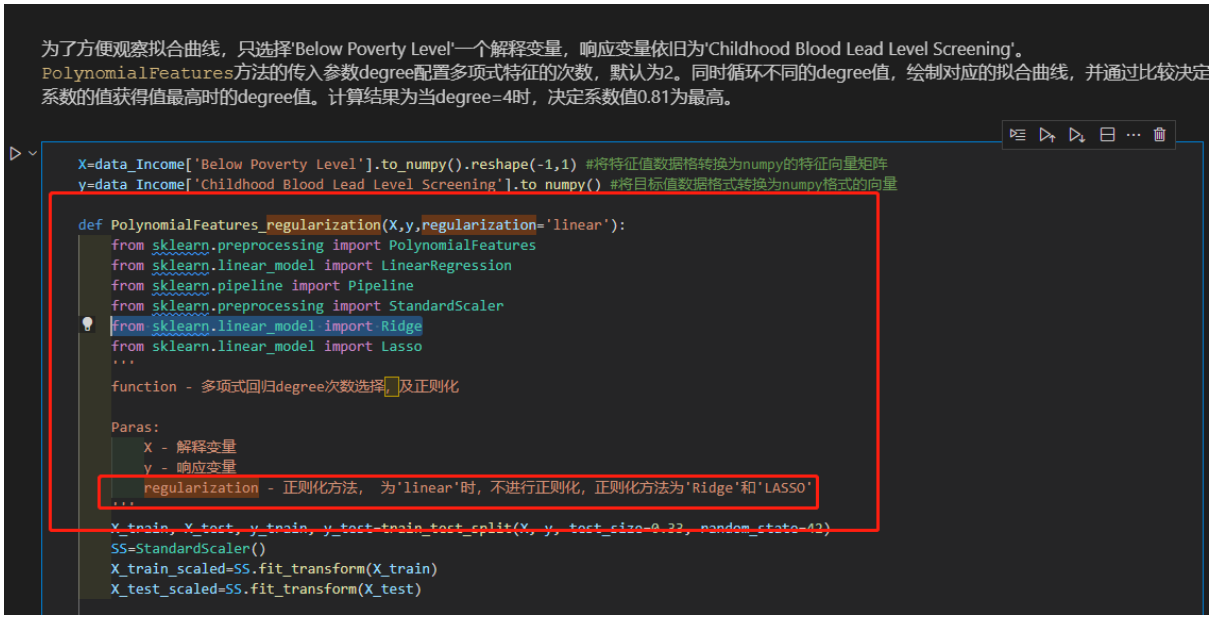
K-近邻算法 - 维基百科，自由的百科全书
https://zh.wikipedia.org/wiki/K-%E8%BF%91%E9%82%BB%E7%AE%97%E6%B3%95

邻近算法_百度百科
https://baike.baidu.com/item/%E9%82%BB%E8%BF%91%E7%AE%97%E6%B3%95/1151153

sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.0.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

后面的**多项式回归**也可以添加一下英文和链接的
多项式回归（Polynomial Regression）
多项式回归 - 维基百科，自由的百科全书
https://zh.wikipedia.org/wiki/%E5%A4%9A%E9%A1%B9%E5%BC%8F%E5%9B%9E%E5%BD%92

7. 定义 **PolynomialFeatures_regularization** 函数方法，正文中缺没有提到



建议在 PolynomialFeatures 之前补充如下内容

下面定义了 PolynomialFeatures_regularization(X,y,regularization='linear')函数，用于多项式回归 degree 次数选择对回归结果影响的比较和讨论，包含了对不同正则化方法的调用。参数 regularization 有三种设置方法，分别是“linear”，“Ridge”和“LASSO”,分别代表不采用正则化，采取 L2 正则化（又称岭回归，Ridge Regression）和 L1 正则化（又称 LASSO）。本小节只对不加正则化的多项式回归方法进行讨论，即，regularization 参数为默认值'linear'。下一节[3)正则化]中将对两种正则化方法对回归结果的影响

进行讨论

并附上 Ridge 和 LASSO 正则化在 sklearn 中的链接

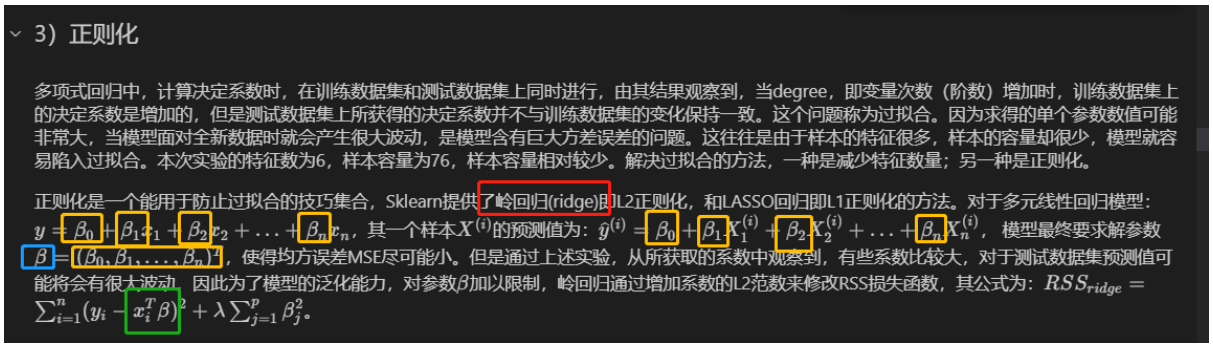
sklearn.linear_model.Ridge — scikit-learn 1.0.2 documentation

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

sklearn.linear_model.Lasso — scikit-learn 1.0.2 documentation

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

8. 后文**[3)正则化]**进一步对这两种正则化进行了描述



3）正则化

多项式回归中，计算决定系数时，在训练数据集和测试数据集上同时进行，由其结果观察到，当degree，即变量次数（阶数）增加时，训练数据集上的决定系数是增加的，但是测试数据集上所获得的决定系数并不与训练数据集的变化保持一致。这个问题称为过拟合。因为求得的单个参数数值可能非常大，当模型面对全新数据时就会产生很大波动，是模型含有巨大方差误差的问题。这往往是由于样本的特征很多，样本的容量却很少，模型就容易陷入过拟合。本次实验的特征数为6，样本容量为76，样本容量相对较少。解决过拟合的方法，一种是减少特征数量；另一种是正则化。

正则化是一个能用于防止过拟合的技巧集合，Sklearn提供了岭回归(ridge)即L2正则化，和LASSO回归即L1正则化的方法。对于多元线性回归模型：$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$，其一个样本$X^{(i)}$的预测值为：$\hat{y}^{(i)} = \beta_0 + \beta_1 X_1^{(i)} + \beta_2 X_2^{(i)} + \ldots + \beta_n X_n^{(i)}$，模型最终要求解参数$\beta = (\beta_0, \beta_1, \ldots, \beta_n)^T$，使得均方误差MSE尽可能小。但是通过上述实验，从所获取的系数中观察到，有些系数比较大，对于测试数据集预测值可能将会有很大波动。因此为了模型的泛化能力，对参数β加以限制，岭回归通过增加系数的L2范数来修改RSS损失函数，其公式为：$RSS_{ridge} = \sum_{i=1}^{n}(y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$。

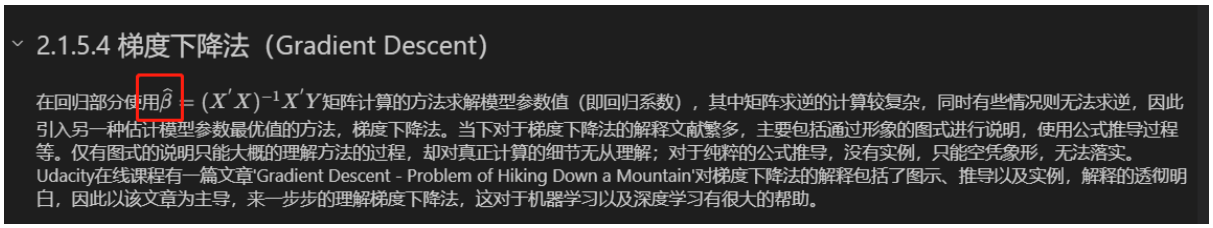
建议将“岭回归(ridge)”改为“Ridge 回归方法”

且此处的β建议全部换成a，作为和前文的对应，且蓝色框标出的β，需要改为加粗的$\boldsymbol{A}$，因为是一个向量

绿色框中的β应该改成a_i，后面的β_i 也改为a_i

添加注意

注意前文的手动实现里，a_0是被放到最后一项的，且值为 1，即$a_0 = 1$，这样是为了和“$y = ax + b$”的形式呼应，希望读者可以注意。

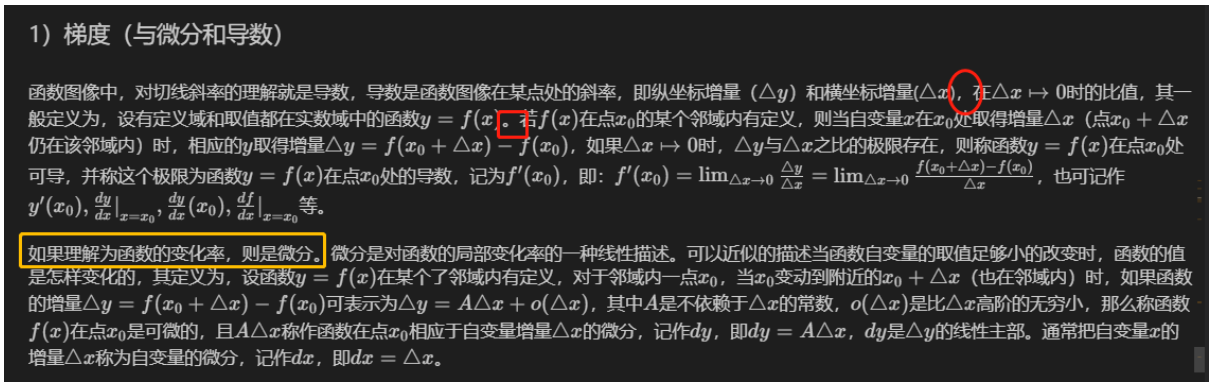


2.1.5.4 梯度下降法（Gradient Descent）

在回归部分使用$\hat{\beta} = (X'X)^{-1}X'Y$矩阵计算的方法求解模型参数值（即回归系数），其中矩阵求逆的计算较复杂，同时有些情况则无法求逆，因此引入另一种估计模型参数最优值的方法，梯度下降法。当下对于梯度下降法的解释文献繁多，主要包括通过形象的图式进行说明，使用公式推导过程等。仅有图式的说明只能大概的理解方法的过程，却对真正计算的细节无从理解；对于纯粹的公式推导，没有实例，只能空凭象形，无法落实。Udacity在线课程有一篇文章'Gradient Descent - Problem of Hiking Down a Mountain'对梯度下降法的解释包括了图示、推导以及实例，解释的透彻明白，因此以该文章为主导，来一步步的理解梯度下降法，这对于机器学习以及深度学习有很大的帮助。

此处的也需要修改为加粗的$\boldsymbol{A}$

9. 标点符号的修改



1）梯度（与微分和导数）

函数图像中，对切线斜率的理解就是导数，导数是函数图像在某点处的斜率，即纵坐标增量（$\triangle y$）和横坐标增量($\triangle x$)，在$\triangle x \mapsto 0$时的比值，其一般定义为，设有定义域和取值都在实数域中的函数$y = f(x)$。若$f(x)$在点x_0的某个邻域内有定义，则当自变量x在x_0处取得增量$\triangle x$（点$x_0 + \triangle x$仍在该邻域内）时，相应的y取得增量$\triangle y = f(x_0 + \triangle x) - f(x_0)$，如果$\triangle x \mapsto 0$时，$\triangle y$与$\triangle x$之比的极限存在，则称函数$y = f(x)$在点$x_0$处可导，并称这个极限为函数$y = f(x)$在点$x_0$处的导数，记为$f'(x_0)$，即：$f'(x_0) = \lim_{\triangle x \to 0} \frac{\triangle y}{\triangle x} = \lim_{\triangle x \to 0} \frac{f(x_0 + \triangle x) - f(x_0)}{\triangle x}$，也可记作$y'(x_0), \frac{dy}{dx}|_{x=x_0}, \frac{dy}{dx}(x_0), \frac{df}{dx}|_{x=x_0}$等。

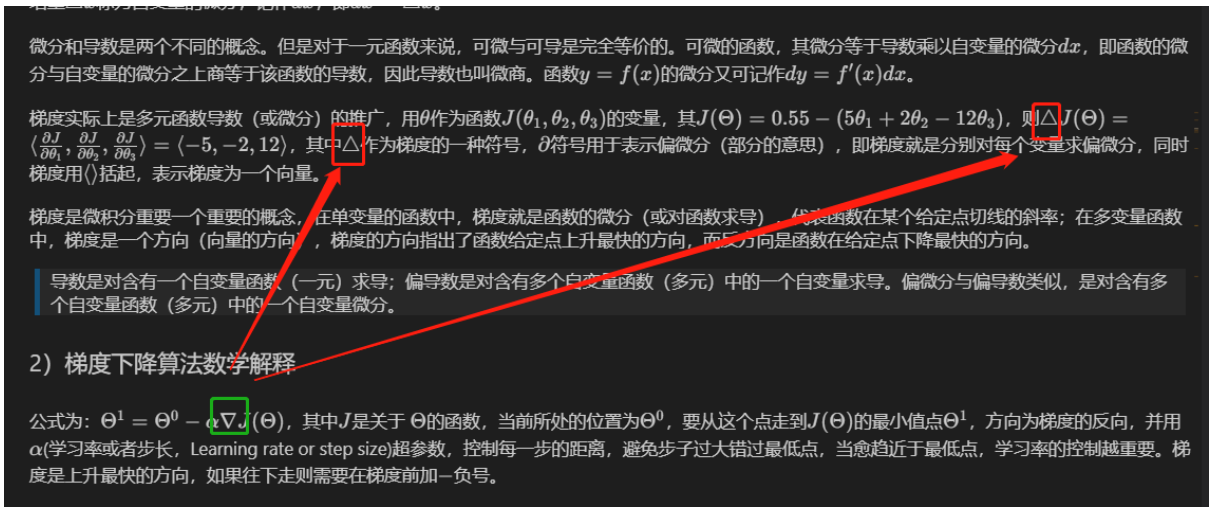
如果理解为函数的变化率，则是微分。微分是对函数的局部变化率的一种线性描述。可以近似的描述当函数自变量的取值足够小的改变时，函数的值是怎样变化的，其定义为，设函数$y = f(x)$在某个了邻域内有定义，对于邻域内一点x_0，当x_0变动到附近的$x_0 + \triangle x$（也在邻域内）时，如果函数的增量$\triangle y = f(x_0 + \triangle x) - f(x_0)$可表示为$\triangle y = A\triangle x + o(\triangle x)$，其中$A$是不依赖于$\triangle x$的常数，$o(\triangle x)$是比$\triangle x$高阶的无穷小，那么称函数$f(x)$在点$x_0$是可微的，且$A\triangle x$称作函数在点$x_0$相应于自变量增量$\triangle x$的微分，记作$dy$，即$dy = A\triangle x$，$dy$是$\triangle y$的线性主部。通常把自变量$x$的增量$\triangle x$称为自变量的微分，记作$dx$，即$dx = \triangle x$。

红色圆圈处，去除逗号，

红色方框处，去除句号，改为逗号

黄色方框处，可删去

10. 梯度的数学表示符号有不对应，建议方框内正三角改为倒三角的统一形式



微分和导数是两个不同的概念。但是对于一元函数来说，可微与可导是完全等价的。可微的函数，其微分等于导数乘以自变量的微分dx，即函数的微分与自变量的微分之上商等于该函数的导数，因此导数也叫微商。函数$y=f(x)$的微分又可记作$dy=f'(x)dx$。

梯度实际上是多元函数导数（或微分）的推广，用θ作为函数$J(\theta_1,\theta_2,\theta_3)$的变量，其$J(\Theta)=0.55-(5\theta_1+2\theta_2-12\theta_3)$，则$\triangle J(\Theta)=\langle\frac{\partial J}{\partial\theta_1},\frac{\partial J}{\partial\theta_2},\frac{\partial J}{\partial\theta_3}\rangle=\langle-5,-2,12\rangle$，其中△作为梯度的一种符号，$\partial$符号用于表示偏微分（部分的意思），即梯度就是分别对每个变量求偏微分，同时梯度用⟨⟩括起，表示梯度为一个向量。

梯度是微积分重要一个重要的概念，在单变量的函数中，梯度就是函数的微分（或对函数求导），代表函数在某个给定点切线的斜率；在多变量函数中，梯度是一个方向（向量的方向），梯度的方向指出了函数给定点上升最快的方向，而反方向是函数在给定点下降最快的方向。

> 导数是对含有一个自变量函数（一元）求导；偏导数是对含有多个自变量函数（多元）中的一个自变量求导。偏微分与偏导数类似，是对含有多个自变量函数（多元）中的一个自变量微分。

2）梯度下降算法数学解释

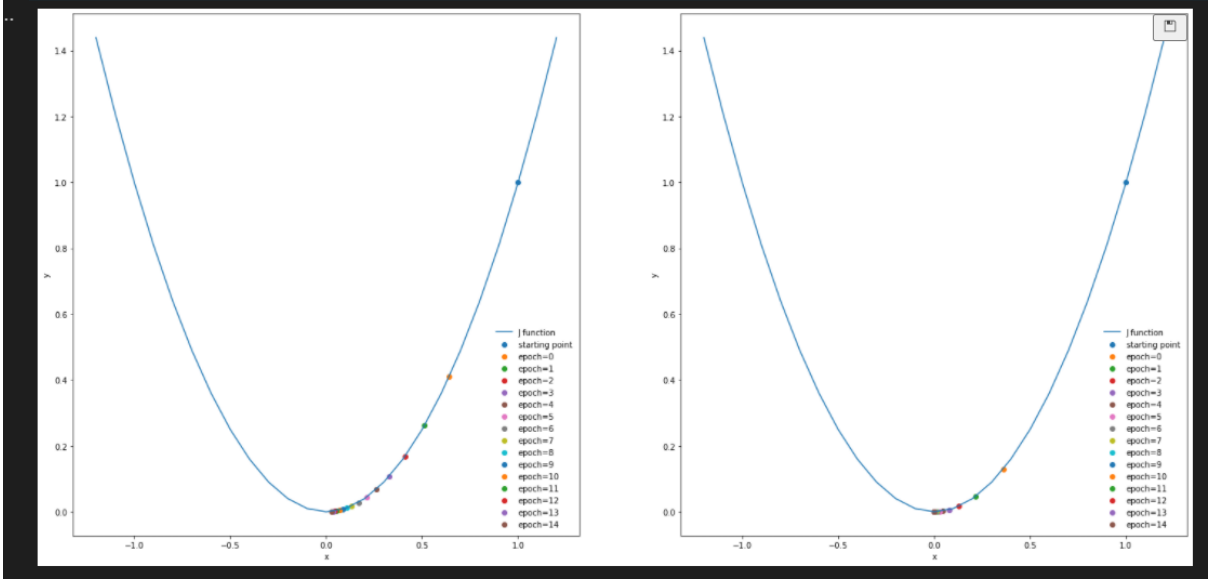
公式为：$\Theta^1=\Theta^0-\alpha\nabla J(\Theta)$，其中$J$是关于Θ的函数，当前所处的位置为$\Theta^0$，要从这个点走到$J(\Theta)$的最小值点$\Theta^1$，方向为梯度的反向，并用$\alpha$(学习率或者步长，Learning rate or step size)超参数，控制每一步的距离，避免步子过大错过最低点，当愈趋近于最低点，学习率的控制越重要。梯度是上升最快的方向，如果往下走则需要在梯度前加−负号。

11. 比较学习率的图像中，并没有绘制表现两幅图学习率不同的信息，
建议在 plt.show()之前添加如下代码

```
ax[0].set_title("learning rate = 0.1")
ax[0].set_title("learning rate = 0.2")
```

```
axs[0].set(xlabel='x',ylabel='y')
axs[1].set(xlabel='x',ylabel='y')
axs[0].legend(loc='lower right', frameon=False)
axs[1].legend(loc='lower right', frameon=False)
plt.show()
```

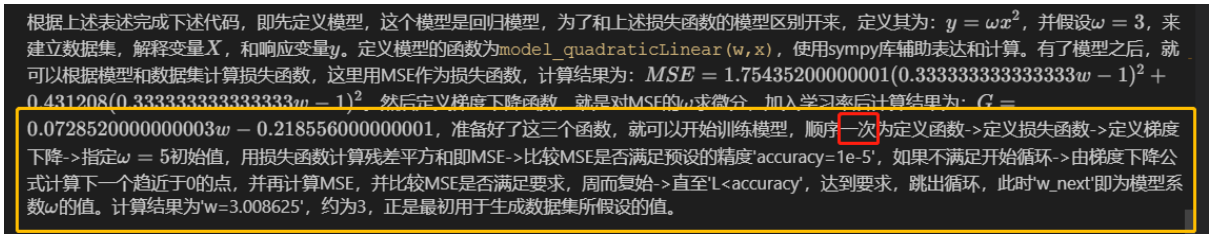
Python



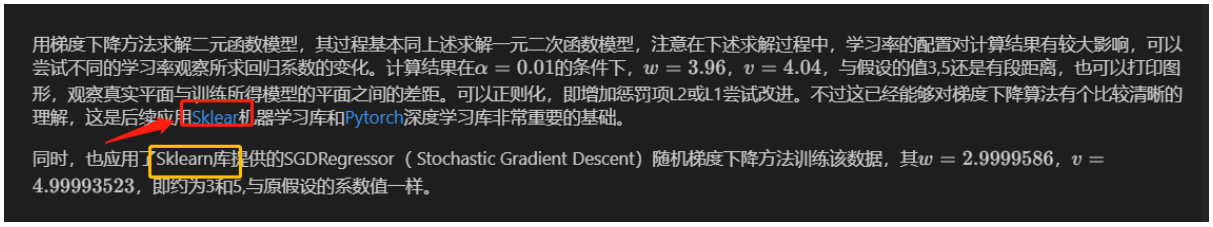
且图片尺寸较大，导致分辨率不够

```
J_=sympy.lambdify(x,J, 'numpy')

fig, axs=plt.subplots(1,2,figsize=(25,12))
x_val=np.arange(-1.2,1.3,0.1)
axs[0].plot(x_val,J_(x_val),label='J function')
axs[1].plot(x_val,J_(x_val),label='J function')

#函数微分
```

12. 发现错别字（一次→依次），且黄色框内内容中的算法流程表示方法不合适，建议更换为 **Visio 流程图**或者**伪代码**

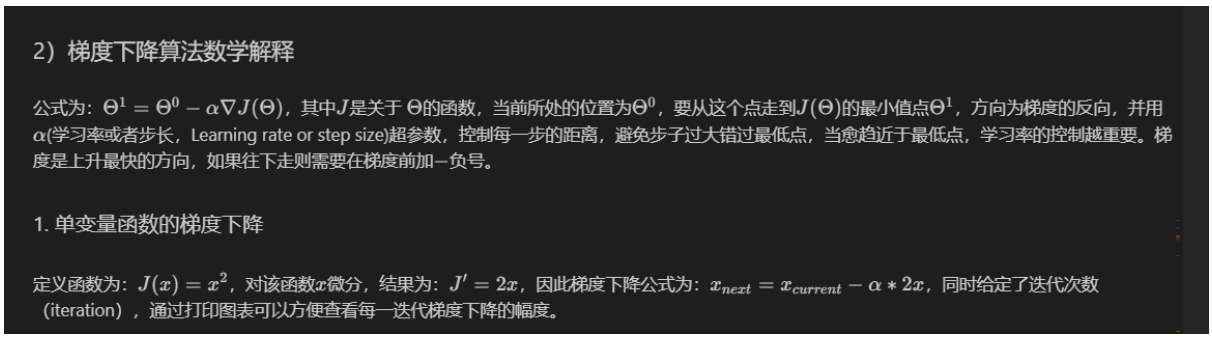

根据上述表述完成下述代码，即先定义模型，这个模型是回归模型，为了和上述损失函数的模型区别开来，定义其为：$y = \omega x^2$，并假设$\omega = 3$，来建立数据集，解释变量X，和响应变量y。定义模型的函数为model_quadraticLinear(w,x)，使用sympy库辅助表达和计算。有了模型之后，就可以根据模型和数据集计算损失函数，这里用MSE作为损失函数，计算结果为：$MSE = 1.75435200000001(0.3333333333333333w - 1)^2 + 0.431208(0.3333333333333333w - 1)^2$。然后定义梯度下降函数，就是对MSE的$\omega$求微分，加入学习率后计算结果为：$G = 0.0728520000000003w - 0.218556000000001$，准备好了这三个函数，就可以开始训练模型，顺序一次为定义函数->定义损失函数->定义梯度下降->指定$\omega = 5$初始值，用损失函数计算残差平方和即MSE->比较MSE是否满足预设的精度'accuracy=1e-5'，如果不满足开始循环->由梯度下降公式计算下一个趋近于0的点，并再计算MSE，并比较MSE是否满足要求，周而复始->直至'L<accuracy'，达到要求，跳出循环，此时'w_next'即为模型系数ω的值。计算结果为'w=3.008625'，约为3，正是最初用于生成数据集所假设的值。

13. 发现错别字（Sklear→Sklearn）


用梯度下降方法求解二元函数模型，其过程基本同上述求解一元二次函数模型，注意在下述求解过程中，学习率的配置对计算结果有较大影响，可以尝试不同的学习率观察所求回归系数的变化。计算结果在$\alpha = 0.01$的条件下，$w = 3.96$，$v = 4.04$，与假设的值3,5还是有段距离，也可以打印图形，观察真实平面与训练所得模型的平面之间的差距。可以正则化，即增加惩罚项L2或L1尝试改进。不过这已经能够对梯度下降算法有个比较清晰的理解，这是后续应用Sklear机器学习库和Pytorch深度学习库非常重要的基础。

同时，也应用了Sklearn库提供的SGDRegressor（Stochastic Gradient Descent）随机梯度下降方法训练该数据，其$w = 2.9999586$，$v = 4.99993523$，即约为3和5,与原假设的系数值一样。

14. 大量重复代码出现时，可以采取包装成函数的方法


2）梯度下降算法数学解释

公式为：$\Theta^1 = \Theta^0 - \alpha\nabla J(\Theta)$，其中$J$是关于$\Theta$的函数，当前所处的位置为$\Theta^0$，要从这个点走到$J(\Theta)$的最小值点$\Theta^1$，方向为梯度的反向，并用$\alpha$(学习率或者步长，Learning rate or step size)超参数，控制每一步的距离，避免步子过大错过最低点，当愈趋近于最低点，学习率的控制越重要。梯度是上升最快的方向，如果往下走则需要在梯度前加−负号。

1. 单变量函数的梯度下降

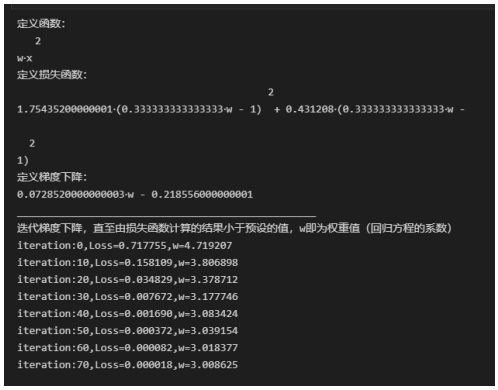
定义函数为：$J(x) = x^2$，对该函数x微分，结果为：$J' = 2x$，因此梯度下降公式为：$x_{next} = x_{current} - \alpha * 2x$，同时给定了迭代次数（iteration），通过打印图表可以方便查看每一迭代梯度下降的幅度。

梯度下降法数学解释一节，对单变量函数和多变量函数的梯度下降以及用 Sklearn 库提供的 SGDRegressor（ Stochastic Gradient Descent）随机梯度下降方法训练分别描述，并用代码展示了不同，但是这几部分代码重复部分较多

建议：可以构造一个可以传入不同个数变量的梯度下降函数，来完成对单变量函数和多变量函数的梯度下降计算，这样，也可以防止内容的冗余，降低读者负担

另一方面，可以保留这样的写法，但是附录中提供这样一个函数（可以传入不同个数变量的梯度下降函数），供读者学习查阅，这样，就可以说，上面的做法虽然略微冗余，但是为了方便读者理解而呈现所有代码。

同时，该部分的信息和变量打印，仍要注意格式上的美观和易读性


```
定义函数:
   2
w·x
定义损失函数:
                                  2
1.75435200000001·(0.3333333333333333·w - 1)  + 0.431208·(0.3333333333333333·w -

  2
1)
定义梯度下降:
0.0728520000000003·w - 0.218556000000001
_____
迭代梯度下降，直至由损失函数计算的结果小于预设的值，w即为权重值（回归方程的系数）
iteration:0,Loss=0.717755,w=4.719207
iteration:10,Loss=0.158109,w=3.806898
iteration:20,Loss=0.034829,w=3.378712
iteration:30,Loss=0.007672,w=3.177746
iteration:40,Loss=0.001690,w=3.083424
iteration:50,Loss=0.000372,w=3.039154
iteration:60,Loss=0.000082,w=3.018377
iteration:70,Loss=0.000018,w=3.008625
```

15. 本节最后，用 Sklearn 库提供的 SGDRegressor （ Stochastic Gradient Descent）随机梯度下降方法训练公共健康数据

只是给出了测试集中的 r-squred score 值和训练出来的参数，此处建议添加对训练学习完的模型，在测试集和训练集中预测的可视化，来帮助读者，学会使用训练的模型

可以参考如下链接

DataTechNotes: Regression Example with SGDRegressor in Python

https://www.datatechnotes.com/2020/09/regression-example-with-sgdregressor-in-python.html