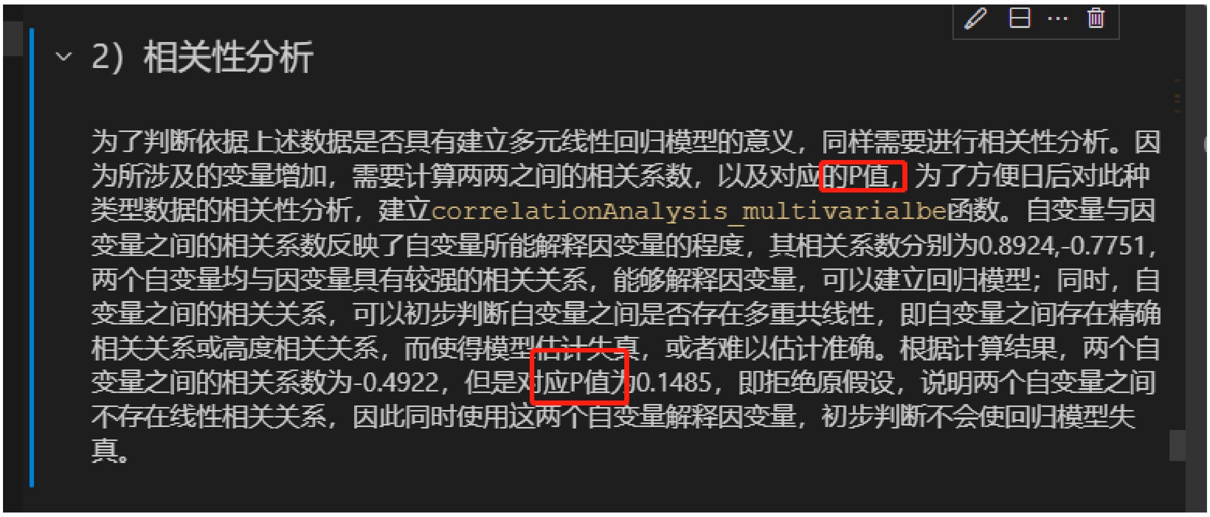
代码建议 四 （章节 2.1.4.3 多元线性回归开始至 2.1.4 结尾）

1. 2.1.4.3 多元线性回归 2）相关性分析中“p 值”的修改



2）相关性分析

为了判断依据上述数据是否具有建立多元线性回归模型的意义，同样需要进行相关性分析。因为所涉及的变量增加，需要计算两两之间的相关系数，以及对应的P值，为了方便日后对此种类型数据的相关性分析，建立correlationAnalysis_multivarialbe函数。自变量与因变量之间的相关系数反映了自变量所能解释因变量的程度，其相关系数分别为0.8924,-0.7751，两个自变量均与因变量具有较强的相关关系，能够解释因变量，可以建立回归模型；同时，自变量之间的相关关系，可以初步判断自变量之间是否存在多重共线性，即自变量之间存在精确相关关系或高度相关关系，而使得模型估计失真，或者难以估计准确。根据计算结果，两个自变量之间的相关系数为-0.4922，但是对应P值为0.1485，即拒绝原假设，说明两个自变量之间不存在线性相关关系，因此同时使用这两个自变量解释因变量，初步判断不会使回归模型失真。

在前文以及之前的修改中，已经提到了对 p-value 值的说明，
建议最后审稿后统一采用“p-value **值为** xx”的说法

2. 对于字母的出现和使用，应定义后再使用或者进行意义含义的解释

因此相关性分析的之后，确定回归方程，在此部分段末，做出如下添加

以上分析可知，对于示例数据，可以采用线性多元回归方程$y=a1x1+a2x2+c$

补充对 2.1.4.2 简单线性回归部分的修改，

二审：

此处更加着重强调了一元简单线性回归的流程，如果不想这样做，可以只保留绿色字的修改

原文	修改
在统计学中，线性回归（linear regression）是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合。只有一个自变量的情况称为简单（线性）回归（simple linear regression），大于一个自变量情况的叫多元回归（multivariable linear regression）。 * 回归分析的流程: 1. 为了讨论是否具有求解回归方程的意义，画出自变量和因变量的散点图（求解相关系数）； 2. 求解回归方程； 3. 确认回归方程的精度； 4. 进行回归系数的检验； 5. 总体回归 Ax+b 的估计； 6. 进行预测	在统计学中，线性回归（linear regression）是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合。只有一个自变量的情况称为简单线性回归（simple linear regression），大于一个自变量情况的叫多元回归（multivariable linear regression）。 * （一元）回归分析的流程: 1. 数据可视化：画出自变量和因变量的散点图，求解相关系数，确定自变量因变量之间是否可以用（一元）线性回归方程$y=Ax+b$解释； 2. 求解回归方程中A和b的值； 3. 确认回归方程的精度； 4. 进行回归系数的检验； 5. 计算总体回归$y=Ax+b$的估计； 6. 进行预测

3. 3）求解多元回归方程中“基本等同于”描述方式的修改

原文	修改
求解多元回归方程的方法基本等同于简单线性回归求解方式	求解多元回归方程的方法与简单线性回归求解方式相类似

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

$$\begin{bmatrix}198\\364\end{bmatrix}$$

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：$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$，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

值和预测值差值的平方和为0，而单个观测值与对应的预测值之间的差值趋于0；二

公式为：$\hat{\beta} = (X'X)^{-1}X'Y$，其中$X = \begin{bmatrix}1 & 10 & 80\\1 & 8 & 0\\1 & 8 & 200\\1 & 5 & 200\\1 & 7 & 300\\1 & 8 & 230\\1 & 7 & 40\\1 & 9 & 0\\1 & 6 & 330\\1 & 9 & 180\end{bmatrix}$，$X' =$

C. 可简单表示为$Y = AX$(完整形式为，$Y = AX + b$，此处处偏置b)，并解释为啥 b 没有不存在于最后公式

对于多元线性回归模型公式：$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$（原公式），$b$首先可以被看做每一个样本中值固定的一个自变量$x_0$，再引入一个可以学习的$a_0$(类似于$a_2 \ldots a_n$)，即，$y = a_0x0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n$，进一步为方便表示和学习，设公式中$x_0$为不需要学习的值，即$x_0 = 1$，而此时$a_0$的值的意义则代表原公式中的偏置$b$，因此多

此部分改动较大，原文截图如下

3）求解多元回归方程

求解多元回归方程的方法基本等同于简单线性回归求解方式，使用最小二乘法对偏回归系数进行求解。求解过程中，使用了三种方法，一是，使用sympy分别对残差平方和SS_res的$a1$、$a2$和b求微分，当各自微分的值等于0时，所反映的残差平方和为0，即观测值和预测值差值的平方和为0，而单个观测值与对应的预测值之间的差值趋于0；二是，使用矩阵计算的方式求解参数，其计算公式为：$\widehat{\beta}=(X'X)^{-1}X'Y$，其中$X=\begin{bmatrix}1&10&80\\1&8&0\\1&8&200\\1&5&200\\1&7&300\\1&8&230\\1&7&40\\1&9&0\\1&6&330\\1&9&180\end{bmatrix}$，$X'=\begin{bmatrix}1&1&1&1&1&1&1&1&1&1\\10&8&8&5&7&8&7&9&6&9\\80&0&200&200&300&230&40&0&330&180\end{bmatrix}$即$X$的的转置，也可记作$X^T,X^{tr}$等，$Y=\begin{bmatrix}469\\366\\371\\208\\246\\297\\363\\436\\198\\364\end{bmatrix}$。对于一个矩阵$X$，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$\widehat{\beta}=(X'X)^{-1}X'Y$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$\beta$，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。

修改后截图如下

3）求解多元回归方程

求解多元回归方程的方法与简单线性回归求解方式相类似，使用最小二乘法对偏回归系数进行求解。求解过程中，使用了三种方法，一是，使用sympy分别对残差平方和SS_res的$a1$、$a2$和b求微分，当各自微分的值等于0时，所反映的残差平方和为0，即观测值和预测值差值的平方和为0，而单个观测值与对应的预测值之间的差值趋于0；二是，使用矩阵计算的方式求解参数，其计算公式为：$\hat{A}=(X'X)^{-1}X'Y$，其中$X=\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix}=\begin{bmatrix}10&80&1\\8&0&1\\8&200&1\\5&200&1\\7&300&1\\8&230&1\\7&40&1\\9&0&1\\6&330&1\\9&180&1\end{bmatrix}$，$X'=\begin{bmatrix}X_1&X_2&\ldots&X_n\end{bmatrix}=\begin{bmatrix}10&8&8&5&7&8&7&9&6&9\\80&0&200&200&300&230&40&0&330&180\\1&1&1&1&1&1&1&1&1&1\end{bmatrix}$即$X$的的转置，也可记作$X^T,X^{tr}$等，$Y=\begin{bmatrix}469\\366\\371\\208\\246\\297\\363\\436\\198\\364\end{bmatrix}$。对于一个矩阵$X$，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$\widehat{A} = (X^{'}X)^{-1}X^{'}Y$再理解：

对于多元线性回归模型公式：$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$（原公式），$b$首先可以被看做每一个样本中值固定的一个自变量$x_0$，再引入一个可以学习的$a_0$(类似于$a_2 \ldots a_n$)，即，$y = a_0x0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n$，进一步为方便表示和学习，设公式中$x_0$为不需要学习的值，即$x_0 = 1$，而此时$a_0$的值的意义则代表原公式中的偏置$b$，因此多元线性回归模型公式可简单表示为：$Y = AX$（对于所有样本$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$和$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$，其矩阵的表示方式为：$Y = AX = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & \ldots & A_n \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} AX_1 \\ AX_2 \\ \vdots \\ AX_n \end{bmatrix}$)，因为矩阵不能相除，因此不能直接对$Y = AX$两边同时除以$X$以求取$A$，但是可以两边同时右乘$X$的逆矩阵$X^{'}$避免除法（矩阵乘以自身的逆矩阵结果为1）。同时只有方阵才可逆，而样本的数量是无法控制的，因此用X乘以其转置$X^{'}$产生一个可以求逆的方阵$X^{'}X$。

完整推导如下

$Y = AX$

首先右乘$X^{'}$,得到$YX^{'} = \widehat{A}XX^{'}$

然后右乘$(XX^{'})^{-1}$,得到$YX^{'}(XX^{'})^{-1} = \widehat{A}XX^{'}(XX^{'})^{-1}$

因为$XX^{'}(XX^{'})^{-1} = 1$，所以化简如下

$YX^{'}(XX^{'})^{-1} = \widehat{A}$

即$\widehat{A} = (X^{'}X)^{-1}X^{'}Y$

附 Markdown 修改，改动较多，**不再标注修改的地方**

原文	修改后
	#### 3）求解多元回归方程 求解多元回归方程的方法与简单线性回归求解方式相类似，使用最小二乘法对偏回归系数进行求解。求解过程中，使用了三种方法，一是，使用 sympy 分别对残差平方和\$SS_res\$的\$a1\$、\$a2\$和\$b\$求微分，当各自微分的值等于 0 时，所反映的残差平方和为 0，即观测值和预测值差值的平方和为 0，而单个观测值与对应的预测值之间的差值趋于 0；二是，使用矩阵计算的方式求解参数，其计算公式为：\$\widehat{ A } = (X^{'} X)^{-1} X^{'}Y\$，其中\$X=\begin{bmatrix} X_{1} \\\\X_{2}\\\\ \vdots\\\\X_{n} \end{bmatrix}=\left[\begin{matrix}10 & 80 &1\\\\ 8 & 0 &1\\\\ 8 & 200 &1\\\\ 5 & 200 &1\\\\ 7 & 300 &1\\\\ 8 & 230 &1\\\\ 7 & 40 &1 \\\\ 9 & 0&1 \\\\6 & 330 &1 \\\\ 9 & 180 &1 \end{matrix}\right]\$，\$X^{'} =\begin{bmatrix} X_{1}& X_{2} & \ldots & X_{n} \end{bmatrix} =\left[\begin{matrix}10 & 8 & 8 & 5 & 7 & 8 & 7 & 9 & 6 & 9\\\\80 & 0 & 200 & 200 & 300 & 230 & 40 & 0 & 330 & 180 \\\\1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{matrix}\right]\$即\$X\$的的转置，也可记作\$X^{T} ,X^{tr} \$等，

$Y=\left[\begin{matrix}469\\366\\371\\208\\246\\297\\363\\436\\198\\364\end{matrix}\right]$。对于一个矩阵$X$，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_{1} x_{1} + a_{2} x_{2}+ \ldots + a_{n} x_{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$\widehat{ A } = (X^{'} X)^{-1} X^{'}Y$再理解：

对于多元线性回归模型公式：$y= a_{1} x_{1} + a_{2} x_{2}+ \ldots + a_{n} x_{n}+b$（原公式），$b$首先可以被看做每一个样本中值固定的一个自变量$x_{0}$，再引入一个可以学习的$a_{0}$(类似于$ a_{2} \ldots a_{n} $)，即，$y=a_{0} x{0} + a_{1} x_{1} + a_{2} x_{2}+ \ldots + a_{n} x_{n}$，进一步为方便表示和学习，设公式中$x_{0}$为不需要学习的值，即$x_{0}=1$，而此时$a_{0}$的值的意义则代表原公式中的偏置$b$，因此多元线性回归模型公式可简单表示为：$Y=AX$（对于所有样本$X =\begin{bmatrix} X_{1}   \\X_{2}\\ \vdots\\X_{n} \end{bmatrix} $和$Y =\begin{bmatrix} Y_{1}   \\Y_{2}\\ \vdots\\Y_{n} \end{bmatrix} $，其矩阵的表示方式为：$Y=AX=\begin{bmatrix} Y_{1}   \\Y_{2}\\ \vdots\\Y_{n} \end{bmatrix} =\begin{bmatrix} A_{1}& A_{2} & \ldots & A_{n} \end{bmatrix}\begin{bmatrix} X_{1}   \\X_{2}\\ \vdots\\X_{n} \end{bmatrix} = \begin{bmatrix} A X_{1} \\ A X_{2}\\ \vdots \\\ A X_{n} \end{bmatrix} $)，因为矩阵不能相除，因此不能直接对$Y=AX $两边同时除以$X$以求取$A$，但是可以两边同时右乘$X$的逆矩阵$X^{'}$避免除法（矩阵乘以自身的逆矩阵结果为 1）。同时只有方阵才可逆，而样本的数量是无法控制的，因此用X乘以其转置$X^{'} $产生一个可以求逆的方阵$X^{'} X$。

完整推导如下

$Y=AX$

首先右乘$X^{'}$,得到$YX^{'}=\widehat{ A } XX^{'}$

然后右乘$(XX^{'})^{-1}$,得到$YX^{'}(XX^{'})^{-1}= \widehat{ A } XX^{'}(XX^{'})^{-1}$

因为$XX^{'}(XX^{'})^{-1}=1$，所以化简如下

$YX^{'}(XX^{'})^{-1}= \widehat{ A } $

```
即$\widehat{ A } = ( X^{'} X)^{-1} X^{'}Y$
```

为了将公式统一成通用常规的形式，正文中修改了*b*列(首列变为最后一列)的位置，对应代码应做出相应修改，且原来代码结果中，对 a1，a2，b（intercept）的解释有错误

```
diff_a1,a2 and intercept:

 ⎧     4073344       -44597         6409648⎫
 ⎨a₁: ───────, a₂: ───────, b: ───────⎬
 ⎩      98121        130828          98121 ⎭
──────────────────────────────────────────
matrix_a1,a2 and intercept:
⎡6409648⎤
⎢───────⎥
⎢ 98121 ⎥
⎢       ⎥
⎢4073344⎥
⎢───────⎥
⎢ 98121 ⎥
⎢       ⎥
⎢-44597 ⎥
⎢───────⎥
⎣ 130828⎦
──────────────────────────────────────────
Sklearn a1=41.51,a2=-0.34,intercept=65.32
linear regression_fx=:

4073344⋅x₁   44597⋅x₂   6409648
────────── - ──────── + ───────
  98121       130828     98121
```

6409648/98121
Out[31]: 65.32391638894833

“ matrix_a1,a2 and intercept ”处，将计算出的最后一个参数误以为是 b，按照原文和代码，第一个参数是 b

且混用 b 和 intercept，这里我们统一使用 b

~~关于 a1，a2，b 的微分值的注释，做出和简单线性回归章节中类似的修改~~【此处代码注释较为合理，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】

<table>
<tr><td>简单线性回归章节中的代码修改</td><td>此章节注释，暂不修改</td></tr>
<tr><td># 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))</td><td>#A- 使用sympy求解多元回归方程
#对残差平方和SS_res关于a1，a1和b求微分，并使微分值为0
diff_SSres_a1=diff(SS_res,a1)
diff_SSres_a2=diff(SS_res,a2)
diff_SSres_b=diff(SS_res,b)

#当微分值为0时，解方程组，获得a1，a2和b的值
Eq_residual_a1=Eq(diff_SSres_a1,0) #设所求a1微分为0
Eq_residual_a2=Eq(diff_SSres_a2,0) #设所求a2微分为0
Eq_residual_b=Eq(diff_SSres_b,0) #设所求a2微分为0
slop_intercept=solve((Eq_residual_a1,Eq_residual_a2,Eq_residual_b),(a1,a2,b))
#计算三元一次方程组</td></tr>
<tr><td># TypeError: 'NoneType' object is not subscriptable</td><td>修改后，可以表面出现这个 error</td></tr>
<tr><td>#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']])</td><td>## 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</td></tr>
</table>

将注释中所有的 intercept 改为 b （红框标出）

```python
#当微分值为0时，解方程组，获得a1，a2和b的值
Eq_residual_a1=Eq(diff_SSres_a1,0) #设所求a1微分为0
Eq_residual_a2=Eq(diff_SSres_a2,0) #设所求a2微分为0
Eq_residual_b=Eq(diff_SSres_b,0) #设所求a2微分为0
slop_intercept=solve((Eq_residual_a1,Eq_residual_a2,Eq_residual_b),(a1,a2,b))
#计算三元一次方程组
print("diff_a1,a2 and intercept:\n")
pprint(slop_intercept)
print("_"*50)

#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']])
else:
    X_m=Matrix(storeInfo_df[['one','area','distance_to_nearestStation']])
y_m=Matrix(storeInfo_df.monthly_turnover)

parameters_reg=(X_m.T*X_m)**-1*X_m.T*y_m #注意在矩阵计算时，矩阵相乘不能任意变化位置
print("matrix_a1,a2 and intercept:\n")
pprint(parameters_reg)

#C - 使用sklearn求解多元回归方程
#B - 使用sklearn库sklearn.linear_model.LinearRegression()，Ordinary least squares Linear Regression-普通最小二乘线性回
from sklearn.linear_model import LinearRegression
X=storeInfo_df[['area','distance_to_nearestStation']].to_numpy()
y=storeInfo_df['monthly_turnover'].to_numpy()

#拟合模型
LR_multivariate=LinearRegression().fit(X,y)
#模型参数
print("_"*50)
print("Sklearn a1=%.2f,a2=%.2f,intercept=%.2f"%(LR_multivariate.coef_[0],LR_multivariate.coef_[1], LR_multivariate.i

#建立回归方程
x1,x2=sympy.symbols('x1,x2')
fx_m=slop_intercept[a1]*x1+slop_intercept[a2]*x2+slop_intercept[b]
print("linear regression_fx=:\n")
pprint(fx_m)
fx_m=sympy.lambdify([x1,x2],fx_m,"numpy")
```

Python

5. 修改公式形式：将一个复杂的分数转换

- **修正自由度的判定系数**

直接使用判定系数时，其自变量的数量越多，判定系数的值越高，但是并不是每一个自变量都是有效的，因此通常使用修正自由度的判定系数，其公式为：$R^2 = 1 - \frac{\frac{SS_{res}}{n_s-n_v-1}}{\frac{SS_{tot}}{n_s-1}}$，其中$n_s$为样本个数，$n_v$为自变量个数，$SS_{res}$为残差平方和，$SS_{tot}$为总的离差平方和。

直接使用判定系数时，其自变量的数量越多，判定系数的值越高，但是并不是每一个自变量都是有效的，因此通常使用修正自由度的判定系数，其公式为：$R^2 = 1 - \frac{SS_{res}}{n_s-n_v-1} / \frac{SS_{tot}}{n_s-1}$，其中$n_s$为样本个数，$n_v$为自变量个数，$SS_{res}$为残差平方和，$SS_{tot}$为总的离差平方和。

原文	修改后
`$R^{2} =1- \frac{ \frac{SS_{res}}{ n_{s} - n_{v} -1} }{ \frac{SS_{tot}}{n_{s} -1} }$`	`$R^{2} =1- \frac{SS_{res}}{ n_{s} - n_{v} - 1} / \frac{SS_{tot}}{n_{s} -1} $`

6. 对于打印信息的修改建议，以此处为例

```
#计算全部回归系数的总体检验统计量
F_total=((SS_tot-SS_res)/dfn)/(SS_res/dfd)
print("F-分布统计量_total=%.6f;p-value=%.6f"%(F_total,f.sf(F_total,dfn,dfd)))

#逐个计算单个回归系数的检验统计量
X=np.insert(X,0,1,1)
X_m=Matrix(X)
M_inverse=(X_m.T*X_m)**-1
C_jj=M_inverse.row(1).col(1)[0]
pprint(C_jj)

F_ai_list=[]
i=0
for a in a_i:
    F_ai=(a**2/C_jj)/(SS_res/dfd)
    F_ai_list.append(F_ai)
    print("a%d=%.6f时,F-分布统计量_=%.6f;p-value=%.6f"%(i,a,F_ai,f.sf(F_total,1,dfd)))
    i+=1

a1_,a2_=LR_multivariate.coef_[0],LR_multivariate.coef_[1]
X=storeInfo_df[['area','distance_to_nearestStation']].to_numpy()
ANOVA_multivarialbe(storeInfo_df.monthly_turnover.to_list(),storeInfo_df.pre.to_list(),2,a_i=[a1_,a2_],X=X)
```

Python

```
F-分布统计量_total=60.410426;p-value=0.000038
6442
─────
98121
a0=41.513478时,F-分布统计量_=44.032010;p-value=0.000110
a1=-0.340883时,F-分布统计量_=0.002969;p-value=0.000110
```

首先是红色方框，只打印 C_jj 变量的值，而没有进行打印提示，这种打印是十分不友好的，且此处，C_jj 的含义，也没有进行注释**【该问题在草稿中还有较多，需要再次审核】**

然后是黄色方框，打印时，仍带有下划线这种多余的字符，且打印没有进行空格，建议最后审核的时候，对所有打印值的格式进行调整，做到美观易读

修改举例 （去除"_"或者"-"，添加 print，添加空格回车"\n"调整输出格式）

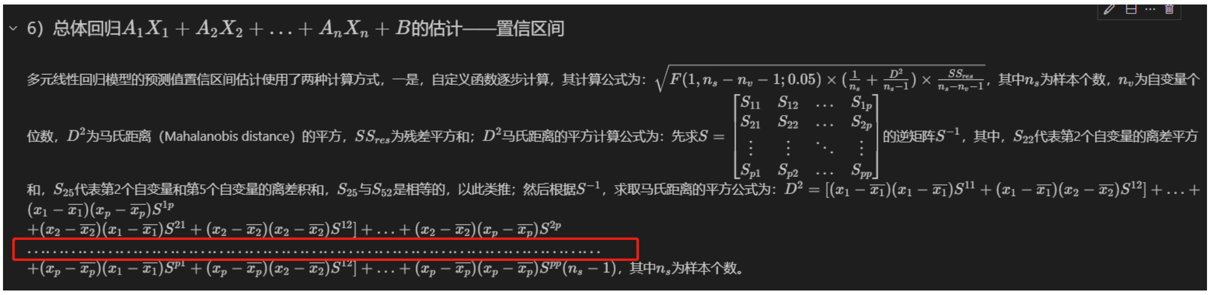
```
print("C_jj值为")
pprint(C_jj)

F_ai_list = []
i = 0
for a in a_i:
    F_ai = (a ** 2 / C_jj) / (SS_res / dfd)
    F_ai_list.append(F_ai)
    print("a%d = %.6f时，F分布统计量 = %.6f;\np-value = %.6f\n" % (i, a, F_ai, f.sf(F_total, 1, dfd)))
    i += 1
```

<table>
<tr><th>原文</th><th>修改后</th></tr>
<tr><td><pre>C_jj=M_inverse.row(1).col(1)[0]
pprint(C_jj)

F_ai_list=[]
i=0
for a in a_i:
    F_ai=(a**2/C_jj)/(SS_res/dfd)
    F_ai_list.append(F_ai)
    print("a%d=%.6f 时，F-分布统计量_=%.6f;p-value=%.6f"%(i,a,F_ai,f.sf(F_total,1,dfd)))
    i+=1</pre></td><td><pre>C_jj = M_inverse.row(1).col(1)[0]
print("C_jj 值为")
pprint(C_jj)

F_ai_list = []
i = 0
for a in a_i:
    F_ai = (a ** 2 / C_jj) / (SS_res / dfd)
    F_ai_list.append(F_ai)
    print("a%d = %.6f 时，F 分布统计量 = %.6f;\np-value = %.6f\n" % (i, a, F_ai, f.sf(F_total, 1, dfd)))
    i += 1</pre></td></tr>
</table>

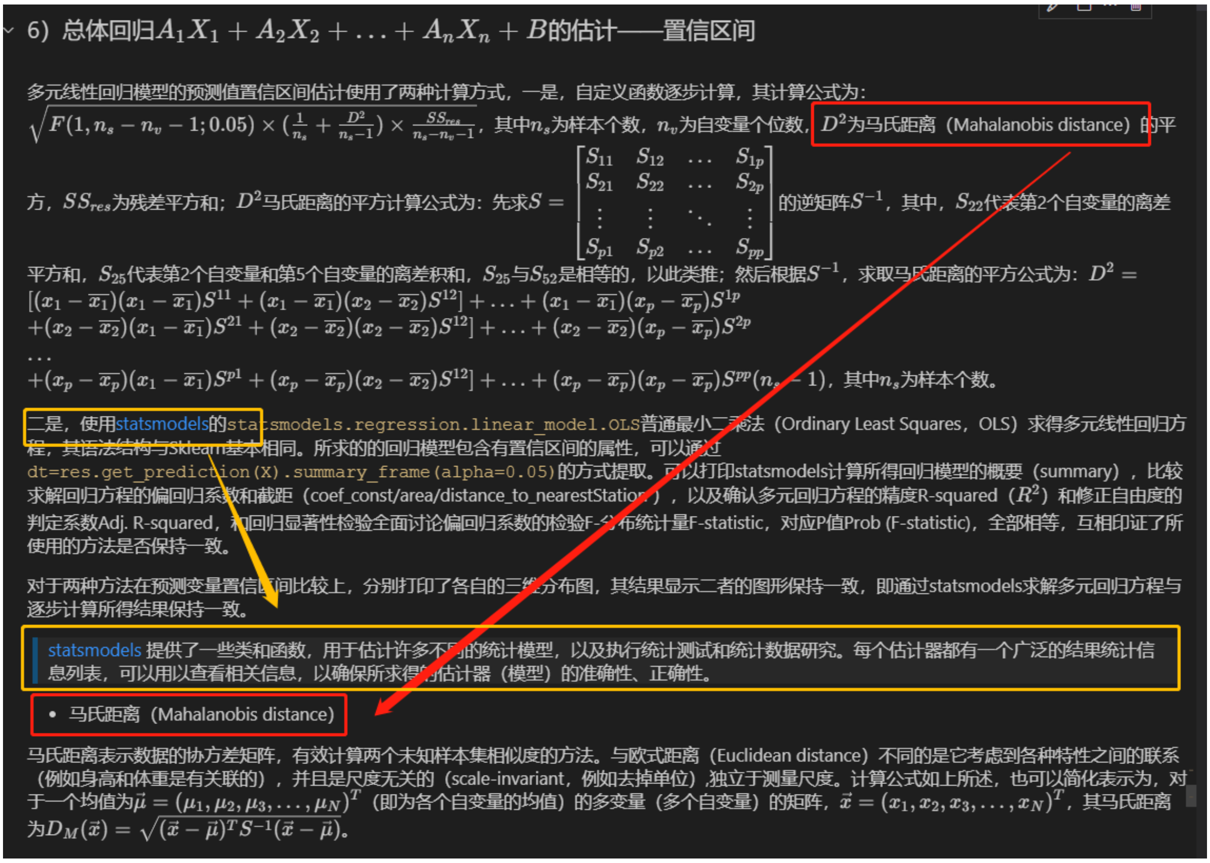
7. “...”过多重复，应该是 typo，建议删除只留一个



6) 总体回归$A_1X_1+A_2X_2+\ldots+A_nX_n+B$的估计——置信区间

多元线性回归模型的预测值置信区间估计使用了两种计算方式，一是，自定义函数逐步计算，其计算公式为：$\sqrt{F(1,n_s-n_v-1;0.05)\times(\frac{1}{n_s}+\frac{D^2}{n_s-1})\times\frac{SS_{res}}{n_s-n_v-1}}$，其中$n_s$为样本个数，$n_v$为自变量个位数，$D^2$为马氏距离（Mahalanobis distance）的平方，SS_{res}为残差平方和；D^2马氏距离的平方计算公式为：先求$S=\begin{bmatrix}S_{11}&S_{12}&\ldots&S_{1p}\\S_{21}&S_{22}&\ldots&S_{2p}\\\vdots&\vdots&\ddots&\vdots\\S_{p1}&S_{p2}&\ldots&S_{pp}\end{bmatrix}$的逆矩阵$S^{-1}$，其中，$S_{22}$代表第2个自变量的离差平方和，$S_{25}$代表第2个自变量和第5个自变量的离差积和，$S_{25}$与$S_{52}$是相等的，以此类推；然后根据$S^{-1}$，求取马氏距离的平方公式为：$D^2=[(x_1-\overline{x}_1)(x_1-\overline{x}_1)S^{11}+(x_1-\overline{x}_1)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_1-\overline{x}_1)(x_p-\overline{x}_p)S^{1p}$
$+(x_2-\overline{x}_2)(x_1-\overline{x}_1)S^{21}+(x_2-\overline{x}_2)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_2-\overline{x}_2)(x_p-\overline{x}_p)S^{2p}$
……………………………………………………………………………………
$+(x_p-\overline{x}_p)(x_1-\overline{x}_1)S^{p1}+(x_p-\overline{x}_p)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_p-\overline{x}_p)(x_p-\overline{x}_p)S^{pp}(n_s-1)$，其中$n_s$为样本个数。

8. 对提及的信息或者术语注释的要保持格式和注释顺序的统一



6) 总体回归$A_1X_1+A_2X_2+\ldots+A_nX_n+B$的估计——置信区间

多元线性回归模型的预测值置信区间估计使用了两种计算方式，一是，自定义函数逐步计算，其计算公式为：
$\sqrt{F(1,n_s-n_v-1;0.05)\times(\frac{1}{n_s}+\frac{D^2}{n_s-1})\times\frac{SS_{res}}{n_s-n_v-1}}$，其中$n_s$为样本个数，$n_v$为自变量个位数，$D^2$为马氏距离（Mahalanobis distance）的平方，SS_{res}为残差平方和；D^2马氏距离的平方计算公式为：先求$S=\begin{bmatrix}S_{11}&S_{12}&\ldots&S_{1p}\\S_{21}&S_{22}&\ldots&S_{2p}\\\vdots&\vdots&\ddots&\vdots\\S_{p1}&S_{p2}&\ldots&S_{pp}\end{bmatrix}$的逆矩阵$S^{-1}$，其中，$S_{22}$代表第2个自变量的离差平方和，$S_{25}$代表第2个自变量和第5个自变量的离差积和，$S_{25}$与$S_{52}$是相等的，以此类推；然后根据$S^{-1}$，求取马氏距离的平方公式为：$D^2=$
$[(x_1-\overline{x}_1)(x_1-\overline{x}_1)S^{11}+(x_1-\overline{x}_1)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_1-\overline{x}_1)(x_p-\overline{x}_p)S^{1p}$
$+(x_2-\overline{x}_2)(x_1-\overline{x}_1)S^{21}+(x_2-\overline{x}_2)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_2-\overline{x}_2)(x_p-\overline{x}_p)S^{2p}$
...
$+(x_p-\overline{x}_p)(x_1-\overline{x}_1)S^{p1}+(x_p-\overline{x}_p)(x_2-\overline{x}_2)S^{12}]+\ldots+(x_p-\overline{x}_p)(x_p-\overline{x}_p)S^{pp}(n_s-1)$，其中$n_s$为样本个数。

二是，使用statsmodels的`statsmodels.regression.linear_model.OLS`普通最小二乘法（Ordinary Least Squares，OLS）求得多元线性回归方程，其语法结构与Sklearn基本相同。所求的的回归模型包含有置信区间的属性，可以通过`dt=res.get_prediction(X).summary_frame(alpha=0.05)`的方式提取。可以打印statsmodels计算所得回归模型的概要（summary），比较求解回归方程的偏回归系数和截距（coef_const/area/distance_to_nearestStation），以及确认多元回归方程的精度R-squared（R^2）和修正自由度的判定系数Adj. R-squared，和回归显著性检验全面讨论偏回归系数的检验F-分布统计量F-statistic，对应P值Prob (F-statistic)，全部相等，互相印证了所使用的方法是否保持一致。

对于两种方法在预测变量置信区间比较上，分别打印了各自的三维分布图，其结果显示二者的图形保持一致，即通过statsmodels求解多元回归方程与逐步计算所得结果保持一致。

> statsmodels 提供了一些类和函数，用于估计许多不同的统计模型，以及执行统计测试和统计数据研究。每个估计器都有一个广泛的结果统计信息列表，可以用以查看相关信息，以确保所求得的估计器（模型）的准确性、正确性。

- 马氏距离（Mahalanobis distance）

马氏距离表示数据的协方差矩阵，有效计算两个未知样本集相似度的方法。与欧式距离（Euclidean distance）不同的是它考虑到各种特性之间的联系（例如身高和体重是有关联的），并且是尺度无关的（scale-invariant，例如去掉单位），独立于测量尺度。计算公式如上所述，也可以简化表示为，对于一个均值为$\vec{\mu}=(\mu_1,\mu_2,\mu_3,\ldots,\mu_N)^T$（即为各个自变量的均值）的多变量（多个自变量）的矩阵，$\vec{x}=(x_1,x_2,x_3,\ldots,x_N)^T$，其马氏距离为$D_M(\vec{x})=\sqrt{(\vec{x}-\vec{\mu})^TS^{-1}(\vec{x}-\vec{\mu})}$。

9. 当打印信息中含有 UserWarning 信息时，建议将这部分删去，不要留在初版的书籍中且，可以采用如下办法去除

```
import warnings
warnings.filterwarnings('ignore')
```

ref: Hide all warnings in ipython - Stack Overflow
https://stackoverflow.com/questions/9031783/hide-all-warnings-in-ipython