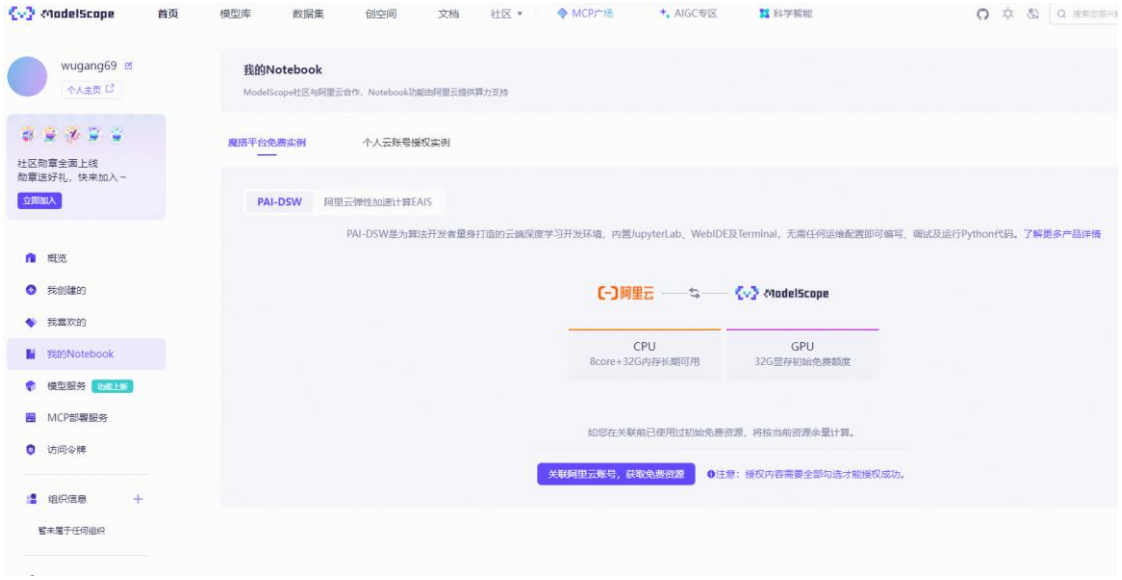
第十章 数据统计分析实验

本章主要介绍了基于 Python 语言的统计分析实验，包括统计检验，线性回归分析和非线性拟合等相关内容。编写过程中参考了文献[20, 21]。对于一些统计分析的理论和思想，通过百度搜索引擎也可以搜索到大量通俗易懂的文章。

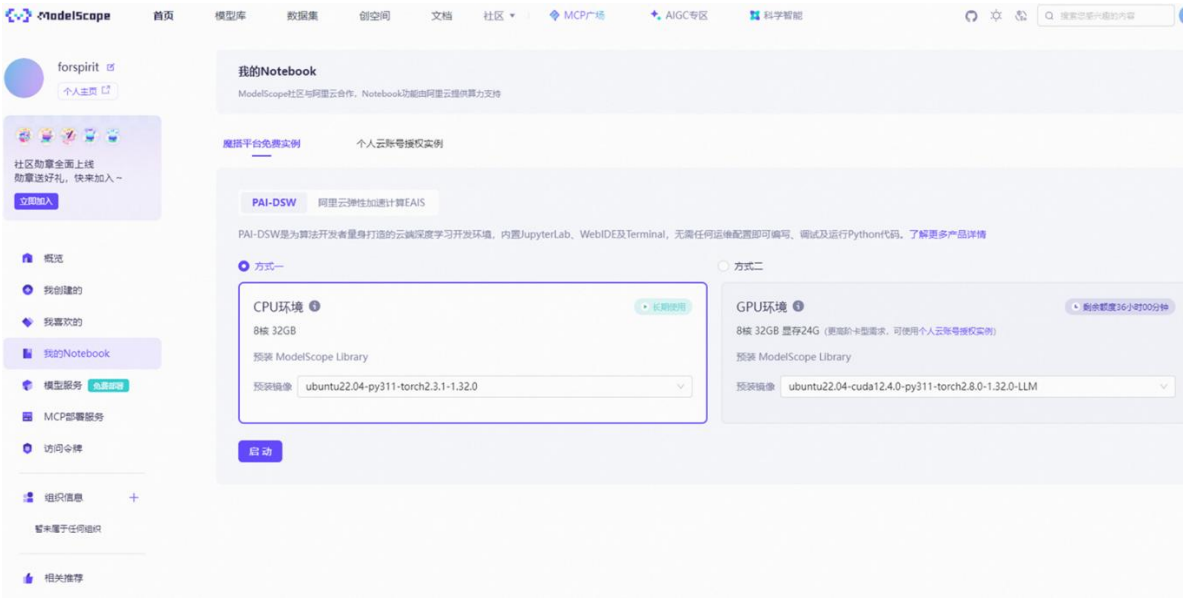
下面在讲述统计分析实验之前，介绍一下 Python 环境的安装。第一个是 webIDE 环境，也就是不需要在本地计算机上安装任何软件，就直接使用第三方提供的线上 python 运行环境，此处介绍阿里云提供的阿里云 ModelScope（魔搭社区 - Notebook）。第二个安装 Anaconda，这个需要到学习通资料贡献中下载安装。此处仅仅介绍魔塔社区提供 Notebook 如何注册使用，具体步骤如下：

(1) 访问 https://modelscope.cn，注册登录后领取免费福利，如下图

(2) 在线开发环境，需要绑定阿里云账号，点击左边“我的 notebook”关联账号，可能支付宝扫码，然后授权具体如下页，关联阿里云账号。

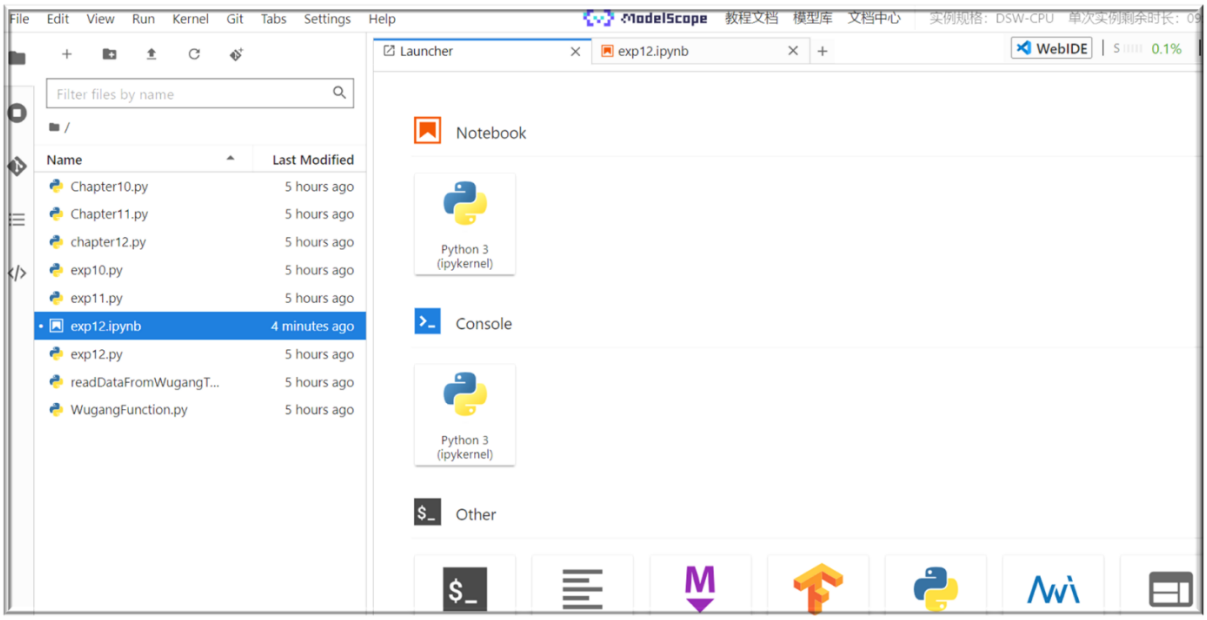


(3) 点击“我的 notebook” 后出现下面界面后，点击启动

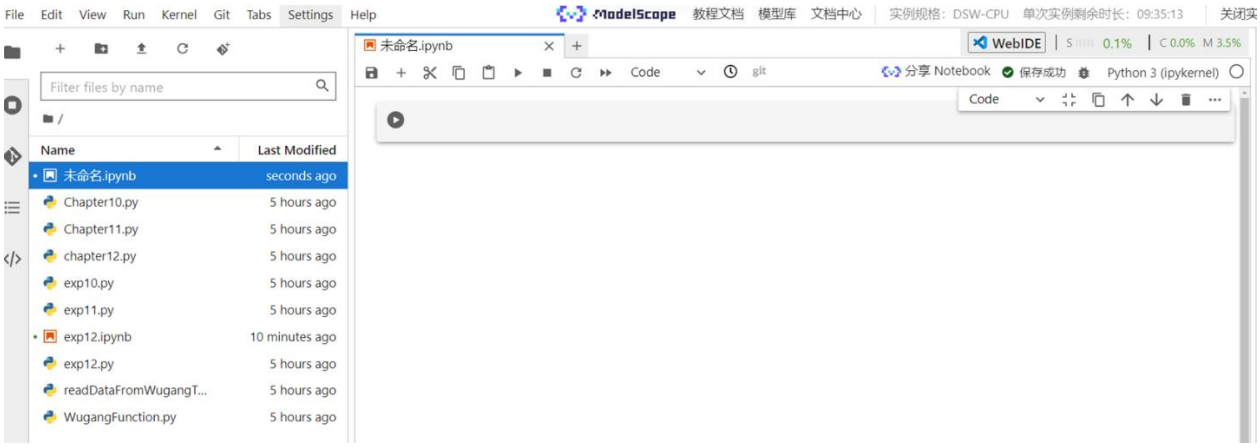


(4) 点击下图中的“查看 Notebook”

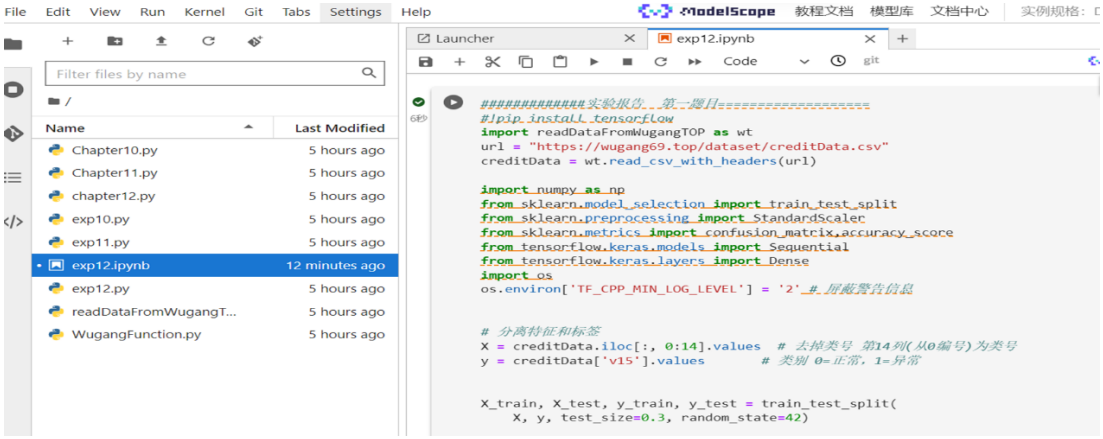
(5) 看到下面界面，左边是我最近上传的文件，你可以点击菜单 run 下面的向上的箭头，上传我提供的两个文件，也就是 readDataFromWugangTOP.py 和文件 WugangFunction.py 这两个文件。然后你点击右边一栏中“Notebook”下面那个“Python3(ipykernel)”按钮，也就是右栏那个红色图标，下面有“Python3(ipykernel)”按钮，点击一下，就可以生成一个文件，具体看第(6)步。右边这一栏的名称是 launcher，如果没有出现这个界面，可以点击左边菜单或者+中的 new launcher 获得这个界面。



(6) 第 5 步点击后会生成一个新的“未命名.ipynb”文件，你可以给这个文件修改名称。最后你将指导书上的代码拷贝到右栏的运行栏即可。你可以修改 settings 菜单中的 themes 中的界面，选择 light 将黑色界面变成下面的白的形式。



(7) 点击右栏代码的左上角那个符号即可运行，运行结果在代码下面观察到，注意如果是运行神经网络代码需要用到 tensorflow 包，可以如下面代码中最上面输入!pip install tensorflow ，一旦安装成功就是用#注释掉，下次再运行就不再安装了，如下图代码中第二行所示。



第一节 基本统计分析

一、频数表和列联表

频数表和列联表主要针对类别型变量，这两个表主要用于独立性检验和相关性度量。创建频数表可以使用 value_counts()函数。创建列联表可以使用 pandas.crosstab()，该函数使用一个变量中的 N 个类型样本，创建一个 N 维列联表。类别变量也就是一个变量里面样本的取值有多种类型，统计每种类型的样本数，也就是频数，如果是两个变量，那么就是一个变量的某一种类型样本对应另一个变量的某种类型样本频数。

现在使用 Arthritis 数据集来做实验，该数据集是一项风湿性关节炎新疗法的双盲临床实验的结果，可以使用这个数据集来做相关实验。下面是 Arthritis 的部分数据，这个数据集类型是 dataframe 数据结构。

```
  ID Treatment Sex   Age Improved
1 57   Treated Male  27     Some
2 46   Treated Male  29     None
3 77   Treated Male  30     None
4 17   Treated Male  32   Marked
5 36   Treated Male  46   Marked
6 23   Treated Male  58   Marked
```

Arthritis["Improved"].value_counts()就是对 Improved 类别型数据进行统计，统计结果是 None :42, Some:14, Marked: 28。其中数值表示是的频数。如果要生成二维列联表，则格式为 pd.crosstab(Arthritis["A"], Arthritis["B"])。其中 A，B 是类别型变量名。表示 A 中每一个类型样本和 B 中每一个类型样本对应的频数。可以使用 crosstab()对 Arthritis [“Treatment”]，Arthritis [“Sex”]生成级联表，比较效果。具体 Python 语言实现：

```
import readDataFromWugangTOP as wt
import pandas as pd
# 下面网址数据集是本教材网站提供的数据集
url = "https://wugang69.top/dataset/Arthritis.csv"
Arthritis = wt.read_csv_with_headers(url) # 个人编写的读取数据函数

oneTable = Arthritis["Improved"].value_counts()
cont_table = pd.crosstab(Arthritis["Treatment"], Arthritis["Sex"])
print("治疗频数表")
print(oneTable)
print("Treatment vs Sex 列联表：")
print(cont_table)
```

显示结果如下

```
治疗频数表
None    42
Some    14
Marked  28
Name: Improved, dtype: int64
Treatment vs Sex 列联表：
Sex        Female  Male
Treatment
Placebo        32    11
Treated        27    14
```

列联表针对的是类别型变量，而 Treatment 取值是安慰剂和治疗剂，Sex 取值是男和女，都是类别型变量，该列联表的意思就是变量为 Treatment 和 Sex 这两个样本集合中，Treatment 取值为 Placebo 时，对应的 Sex，其中取值 Female 的有 32 个，取值 male 的有 11 个；Treatment 取值为 Treated 时，取值 Female 的有 27 个，取值 Male 的有 14 个。取 Arthritics 数据的三个向量，查看它们之间的列联表

```
table = pd.crosstab([Arthritis['Treatment'], Arthritis['Sex']], Arthritis['Improved'])
print(table)
```

显示结果如下

```
Improved          None  Some  Marked
Treatment Sex
Placebo   Female    19     7       6
          Male      10     0       1
Treated   Female     6     5      16
          Male       7     2       5
```

参与列联表计算的有三个变量，分别是 Treatment，Sex 和 Improved。观察上面结果可以得出，这个函数是将 Improved 数据类型为基础，统计其它两个变量中包含的样本类型数目。

二、卡方独立性检验

卡方检验主要用在分析类别型数据与类别型数据之间的关系检验。现在假设治疗和病情改善效果之间是独立的，性别与病情改善之间也是独立的，也就是它们是没有关系。此处实验使用 scipy.stats 包中的 chi2_contingency()函数进行独立性检验，pip install scipy 具体负责安装该包，下面是具体卡方检验实验代码。

```
import pandas as pd
from scipy.stats import chi2_contingency
mytable1 = pd.crosstab(arthritis['Treatment'], arthritis['Improved'])

mytable2 = pd.crosstab(arthritis['Improved'], arthritis['Sex'])

# 对 mytable1 做卡方检验
chi2_1, p_1, dof_1, expected_1 = chi2_contingency(mytable1)
print(f"Chi2 = {chi2_1}, p-value = {p_1}, dof = {dof_1}")

# 对 mytable2 做卡方检验
chi2_2, p_2, dof_2, expected_2 = chi2_contingency(mytable2)
print(f"Chi2 = {chi2_2}, p-value = {p_2}, dof = {dof_2}")
```

显示的结果分别是

```
X-squared = 13, df = 2, p-value = 0.001
X-squared = 4.8, df = 2, p-value = 0.09
```

可以发现治疗和病情改善效果之间独立性的概率为 p=0.001，概率很小，小于 0.05，应该拒绝它们之间相互独立，也就是拒绝没有关系的假设，即是有关系的，也就是这种药物治疗是有效的。性别和病情改善效果检验的概率为 p=0.09，概率较大，也就是大于 0.05，应该接受假设条件，即其独立性可能成立。需要注意的是我们假设是无关的，而概率较大，则说明说明假设是对的，也就是无关的，概率较小则说明假设是错的，也就是相关的。

卡方统计量值计算公式为

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - T_i)^2}{T_i} \tag{1}$$

这个卡方值计算公式很容易理解，就是列联表中，类别型变量所有取值的观测值减去理论值然后和理论值的比率之和。其中O_i表示某一类型样本实际频数，而T_i是假设两个变量相互独立计算对应的理论频数。比如上列中 mytable1 列联表如下

	Improved		
Treatment	None	Some	Marked
Placebo	29	7	7
Treated	13	7	21

我们看上面列联表，显然 Treated 对应的 Marked 数目明显比 Placebo 多，为什么还要检验呢？这就涉及到另外一个问题，那就是如果样本数少，即使频数相对较大也不能说明问题，因此需要在综合考虑 Marked 数目和样本个数的基础上，对假设是否显著进行检验。例如上式(1)，如果T_i很大，O_i和T_i即使相差相对较大，可是计算的卡方值可能仍然很小，也就是这两种类型变量仍然是无关的。现在假设 Treatment 和 Improved 是无关的，这就意味着取 None，Some 和 Marked 的概率取值分别是每一列之和除以样本总数，即取值分别为 42/84，14/84 和 28/84。既然病情改善情况与是否服用 Placebo 和 Treated 无关，那么为什么病人还有 None，Some 和 Marked 的情况出现呢？这可能是由该病情特点决定的，比如天气环境、饮食习惯和个体差异对这种病情都可能造成一段时间以后，有的改善多些，有的改善少些等等。于是理论概率可以这样计算：Placebo 的总样本数是 43，它乘以 None，Some 和 Marked 的概率就可以得到 Placebo 在 None，Some 和 Marked 的取值，这就是理论取值，同样可以计算 Treated 对应的理论取值，于是就可以代入上面的卡方统计量计算公式了。显然如果满足假设，也就是两个变量是独立的，就表明理论和实际值近似相等，根据公式(1)，此时的卡方统计量值就会很小，否则很大。如果随机变量是都是独立的，满足正态分布，那么这些随机变量的平方和满足卡方分布，因此可以计算该分布给定卡方值的概率。Python 语言使用 scipy.stats 包中 chi2.cdf(χ,n)来计算累积分布值，其中 χ 是卡方值，n是自由度，因此可以使用它计算在独立性假设正确的 p 值，也就是概率值。下面代码是据此编写的函数，注意该函数参数是列联表数据。上面算法总结为按列计算

概率值，按行计算理论值。

```
import numpy as np
from scipy.stats import chi2
def kSquare(myDataSet):
    # dataset: numpy array, n行 m列
    myDataSet = np.array(myDataSet)
    rowNum, colNum = myDataSet.shape
    # 总和
    ssValue = myDataSet.sum()
    # 每一列概率
    pValue = myDataSet.sum(axis=0) / ssValue
    # 每一行总数
    rowVector = myDataSet.sum(axis=1).reshape(-1, 1)
    # 理论矩阵（期望值）
    dataSetTheory = rowVector @ pValue.reshape(1, -1)
    # 卡方统计量
    kSquareValue = np.sum((myDataSet - dataSetTheory) ** 2 / dataSetTheory)
    # 自由度
    dfValue = (rowNum - 1) * (colNum - 1)
    # p值
    pChistValue = 1 - chi2.cdf(kSquareValue, dfValue)
    return kSquareValue, pChistValue
```

卡方检验的原理就是独立的标准正态随机变量平方和服从自由度为k的卡方分布，下面就以四个标准正态随机变量取值，然后构建卡方分布。图 13.1 就是自由度取值为 1，2，3，5 后得到的卡方分布图形。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

# 生成随机数，相当于 rnorm(1000000)
n = 1000000
x1 = np.random.normal(size=n)
x2 = np.random.normal(size=n)
x3 = np.random.normal(size=n)
x4 = np.random.normal(size=n)
x5 = np.random.normal(size=n)

# 构造卡方分布模拟
Q1 = x1**2
Q2 = x1**2 + x2**2
Q3 = x1**2 + x2**2 + x3**2
Q5 = x1**2 + x2**2 + x3**2 + x4**2 + x5**2

# 计算核密度估计
kde_Q1 = gaussian_kde(Q1)
kde_Q2 = gaussian_kde(Q2)
kde_Q3 = gaussian_kde(Q3)
kde_Q5 = gaussian_kde(Q5)

# 生成横坐标范围
x = np.linspace(0, 6, 500)

# 绘图
plt.figure(figsize=(8,6))
plt.plot(x, kde_Q1(x), color='blue', lw=1.5, label='df=1')
plt.plot(x, kde_Q2(x), color='black', lw=1.5, label='df=2')
plt.plot(x, kde_Q3(x), color='red', lw=1.5, label='df=3')
plt.plot(x, kde_Q5(x), color='green', lw=1.5, label='df=5')

plt.xlim(0, 6)
plt.ylim(0, 0.6)
plt.title("Chi-square (simulation)", fontsize=14)
plt.xlabel("")
plt.ylabel("")
plt.legend(loc="upper right")
plt.show()
```

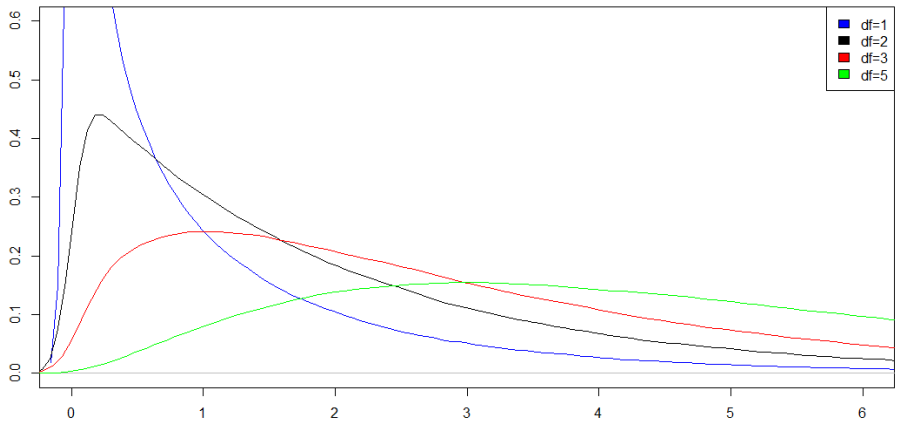


图 13.1 不同自由度下的卡方分布图

通过图 13.1 可以发现，自由度越大，越是趋于正态分布。这是显然的，因为随机变量都是正态分布，样本如果足够大，其平方和也应该满足正态分布。我们看上面例子中，治疗和改善计算的卡方值为 13，可以查看上面的黑色曲线(自由度为 2)，可以发现计算的概率值将小于 0.05，因此应该拒绝独立性假设，但是如果使用置信度的话，比如 95%，也就是概率为大于 95%时接受独立性假设，那么可以查看上面图形，95%对应的卡方值应该在 0.1 左右，也就是计算的卡方值小于 0.1 才能接受假设，而现在计算的卡方值为 13，远远大于 0.1，因此应该拒绝假设。此处要注意使用的是置信度水平还是显著性水平(也就是 p 值)。

至于自由度，通过上面的分析可以看出，就是随机变量的个数。对于列联表而言，列数和行数都可以看作随机变量，因此每一个表格看作一个随机变量取值，但是由于样本均值的原因，最终计算的自由度是行数减去 1 乘以列数减去 1。还有一点需要说明，那就是上例中，Treatment 和 Improved 都属于正态分布，假设是无关的，那么将列表转置进行卡方检验，计算结果是不变的，也就是检验 Treatment 和 Improved 的相关性与检验 Improved 和 Treatment 的相关性是一样的。

假设治疗和病情改善效果之间是独立的，即如果药物无效，就是服用药物和服用安慰剂效果一样，那么所有服用药物的样本和服用安慰剂的样本对治疗效果的概率分布应该是一样的。

三、相关关系计算和检验

相关系数是用来度量定量变量之间的关系，不是定性变量之间的关系，即不是类别型或者其它型变量之间关系。相关系数包含 Pearson 相关系数，Spearman 相关系数和 Kendall 相关系数。Pearson 相关系数衡量两个定量变量之间线性相关程度。Spearman 衡量分级定序变量之间相关程度，Kendall 是一种非参数的等级相关度量。python 语言的 pandas 包里面 corr()函数负责计算相关系数，直接在数据集合后面调用该函数即可。这些函数的参数当然应该是二维数据表，每行是一个观测，每一列是一个变量对应的向量值。本教材提供数据集 state_x77 下载，该数据集含有某个国家 50 个地区 1977 年的人口、收入、文盲率、预期寿命、谋杀率和高中毕业数据，而且有行名称和列名称。我们使用 corr 函数计算这些变量之间的计算相关系数，该函数默认使用 Pearson 相关系数计算方法，其代码和结果如列表 13.1 所示

列表 13.1 某国 50 个地区关于人口和收入等变量的相关系数。

```
import readDataFromWugangTOP as wt
import pandas as pd
url = "https://wugang69.top/dataset/state_x77.csv"
state_x77 = wt.read_csv_with_headers(url)
print(state_x77)

from scipy import stats
# 1. 计算相关系数矩阵
# 删除州名列
state_x77 = state_x77.drop(columns=["State"])
corr_matrix = state_x77.corr()
print("相关系数矩阵：")
print(corr_matrix)
```

运行结果如下：

	Population	Income	Illiteracy	Life Exp	Murder	HS Grad	Frost	Area
Population	1.00000	0.2082	0.10762	-0.06805	0.3436	-0.09849	-0.33215	0.02254
Income	0.20823	1.0000	-0.43708	0.34026	-0.2301	0.61993	0.22628	0.36332
Illiteracy	0.10762	-0.4371	1.00000	-0.58848	0.7030	-0.65719	-0.67195	0.07726
Life Exp	-0.06805	0.3403	-0.58848	1.00000	-0.7808	0.58222	0.26207	-0.10733
Murder	0.34364	-0.2301	0.70298	-0.78085	1.0000	-0.48797	-0.53888	0.22839
HS Grad	-0.09849	0.6199	-0.65719	0.58222	-0.4880	1.00000	0.36678	0.33354
Frost	-0.33215	0.2263	-0.67195	0.26207	-0.5389	0.36678	1.00000	0.05923
Area	0.02254	0.3633	0.07726	-0.10733	0.2284	0.33354	0.05923	1.00000

从列表 13.1 可以发现，Hs Grad 和 Life Exp 的相关系数为 0.5822，说明它们有很强的正相关性，Hs Grad 与 Murder 的相关系数为-0.4880，说明很强的负相关，而 Illiteracy 与 Murder 相关系数为 0.7030，这说明它们正相关性极强。当然 corr()函数也可以计算任意两组变量之间的相关系数。

列表 13.1 计算的相关性是否可靠，需要进行检验，因为如果样本数很少，计算的相关性未必是可靠的。相关性的显著性检验常用的原假设是变量间不相关，也就是相关系数为 0。现在假设 Illiteracy 与 Murder 变量间不相关成立，然后对其进行检验。检验使用的 scipy.stats 模块中函数，即 pearsonr()，spearmanr()和 kendalltau()三种类型的相关系数检验。相关性检验代码为

```
corr, p_value = stats.pearsonr(state_x77["Illiteracy"],state_x77["Murder"])
print("\nIlliteracy 与 Murder 的相关系数:", corr)
print("p 值:", p_value)
if p_value < 0.05:
    print("结论：拒绝原假设（两者相关性显著，不独立）")
else:
    print("结论：不能拒绝原假设（没有显著相关性）")
```

检验的结果如下

```
Illiteracy 与 Murder 的相关系数: 0.70
p 值: 1.25e-08
结论：拒绝原假设（两者相关性显著，不独立）
```

我们观察到，不相关的可能性 P=1e-08，这说明 Illiteracy 与 Murder 变量间不相关的假设不成立，也就是 Illiteracy 与 Murder 高度相关是可信的。

四、t 检验

卡方检验是类别型变量关系的检验，而 t 检验可用于分析连续型变量与类别变量之间关系。这种检验假设自变量是连续型变量，并且符合正态分布，而因变量是类别型变量，也是只能有两个等级。t 检验适用于分析这两个组别上的均值是否存在显著差异。其原假设是“两组的总体均值没有差异”。t 检验目的是检验两组样本关于自变量的概率分布是否一致，也就是这两组样本是否相关，如果相关就说明它们属于同一个概率分布，从卡方检验来说，这就说明自变量和因变量不相关，也就是独立的，由此可见卡方检验和 t 检验的思想是一致的。

单体 t 检验的思想是通过比较给定样本均值与已知或假设的总体分布中的均值来判断样本均值是否有显著差异。也就是单体 t 检验用于检验一个样本的均值是否和已知或假设的总体均值存在显著差异。下面是单体检验的 t 统计量计算公式

$$t = (\bar{x} - \mu)/(s/\sqrt{n})$$

其中$\bar{x}$是样本平均值，μ是总体均值的假设值，s是样本标准差，n是样本容量。双体 t 检验的基本思想是比较两个独立样本的均值，以确定它们是否有显著差异。该检验通常用于研究两组样本之间的均值差异。例如比较两种治疗方法的效果、两个群体的平均成绩等。双体 t 检验的 t 统计量计算公式为

$$t = (\overline{x_1} - \overline{x_2})/\sqrt{s_1^2/n_1 + s_2^2/n_2}$$

其中x_1和x_2分别是两个样本的平均值，s_1和s_2分别是两个样本的标准差，n_1和n_2分别是两个样本的容量。t 分布函数是一个关于给定样本个数的函数，近似正态分布，因此 t 统计量值越大，则在 t 统计分布下得到的值越小，也就是拒绝H_0零假设，也就是两个样本的均值相等的假设不成立，这也意味两组样本显著差异。

t 检验的 python 语言函数可以是 scipy.stats 中的函数 ttest_ind。现在使用数据集 UScrime，对被判监禁的概率和是否与处于某国家的南方地区和非南方地区有关系。显然南方和北方被监禁概率这

两组数据是独立的，也就是独立样本。现在假设没有差异，也就是H_0零假设进行检验。在该国南方地区和不在该国南方地区作为一个自变量，被判监禁的概率为自变量进行 t 检验，具体检验代码如下

```
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/UScrime.csv"
UScrime = wt.read_csv_with_headers(url)
print(UScrime)

import pandas as pd
from scipy import stats

# 2. 分组数据
prob_south = UScrime[UScrime["So"] == 1]["Prob"]   # 南方组
prob_north = UScrime[UScrime["So"] == 0]["Prob"]   # 非南方组

# 3. 独立样本 t 检验。# Welch t-test,取值 False 表示两个变量方差不等。
t_stat, p_value = stats.ttest_ind(prob_north,prob_south, equal_var=False)
print("t 统计量:", t_stat)
print("p 值:", p_value)

# 4. 结论
alpha = 0.05
if p_value < alpha:
    print("结论：拒绝原假设，南方与非南方的犯罪概率有显著差异。")
else:
    print("结论：不能拒绝原假设，南方与非南方的犯罪概率没有显著差异。")
```

运行结果如下

```
t 统计量: -3.8953717090736655
p 值: 0.0006505783178002
结论：拒绝原假设，南方与非南方的犯罪概率有显著差异。
```

其中 UScrime["Prob"]是犯罪概率，UScrime["So"]取值为 0 或者 1，取 0 时候不在南方，取 1 是表示在南方，因此是取两个等级的类别型变量。从检验结果可以观察到这种没有差异的概率为 0.0007，因此可以排除这种假设，这就意味着在南方犯罪和在其它地方犯罪，被监禁的可能性是不一样的，南方更可能被监禁。

如果两组样本属于同一种概率分布，意味着这两组样本是相关的，这就表明监禁概率和地理位置无关，也就是监禁概率和地理位置是独立的

第二节 线性回归分析

回归分析通常是指使用多个预测变量(自变量)来预测响应变量(因变量)的方法，但是更为重要的是通过回归分析来挑选与响应变量相关的解释变量，也就是所有可用的预测变量中，哪些变量对因变量影响最大，哪些变量次之，最终哪些预测变量进入回归模型中来预测因变量。并不是所有的预测变量都拿过来做自变量进而和因变量建立回归模型，因为有的预测变量可能对因变量没有影响，有的几个预测变量之间有明显的相关性，如果都作为预测变量的话，会干扰对因变量的预测，所以就存在几个问题：(1)预测变量和因变量相关关系问题；(2)预测变量之间的独立性问题；(3)回归模型中预测变量的选择问题；(4)预测因变量的准确度问题等等。

一、普通最小二乘回归(OLS)

1. OLS 的应用条件

普通最小二乘回归的思想就是使得所有的观测向量和预测向量距离平方之和最小。使用最小二乘法很容易推导出回归模型公式。OLS 构建回归模型不是没有条件的，正如上面分析的那样，对预测变量之间和预测变量和因变量之间都要满足一定的关系才能使用回归模型，进而解释得到回归系数的物理意义。这些条件就是(1)依赖正态性；(2)因变量独立性；(3)关系线性；(4)方差齐性。依赖正态性指的是对于固定自变量值，因变量值呈正态分布，也就是因变量取值的误差应该呈均值为 0 的正态分布；因变量独立性就是因变量取值是相互独立的；关系线性指的是因变量和自变量之间

是线性关系；方差齐性指的是因变量误差的方差恒定，不随自变量取值变化(即不固定自变量)，即模型对不同大小、范围的 x 预测得一样稳定，残差的波动程度不会随着 x变大而变得更大或更小，常使用标准化残差来衡量。如果这些假设不成立，那么统计显著性检验和所得的置信区间就可能不精确。

2. 线性回归的 Python 语言函数实现

普通线性回归在 Python 语言中可以使用 statsmodels.api 的 OLS 对象来实现。如果没有安装该模块，则要在 spyder 的 console 中运行 pip install statsmodels。使用这个函数的时候，需要使用该包的 add_constant()给自变量加截距项，否则默认线性函数通过坐标原点。

现在有一个数据集 women，提供了 15 个年龄在 30-39 岁女性身高和体重信息。希望通过身高预测体重，进而分辨出哪些女性胖了，哪些女性瘦了。这是一个最简单的线性回归，因为只有一个预测变量，那就是身高。具体代码如下

```
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/women.csv"
women = wt.read_csv_with_headers(url)
print(women)

import pandas as pd
import statsmodels.api as sm

x = sm.add_constant(women["height"])  # 在 X 前添加一列常数 1，对应截距项
y = women["weight"]
# 4. 拟合线性回归模型
model = sm.OLS(y, x).fit()

# 5. 查看回归结果
print(model.summary())
```

拟合的部分结果如下

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 weight   R-squared:                       0.991
Model:                            OLS   Adj. R-squared:                  0.990
Method:                 Least Squares   F-statistic:                     1433.
Date:                Wed, 03 Sep 2025   Prob (F-statistic):           1.09e-14
Time:                        14:57:36   Log-Likelihood:                -26.541
No. Observations:                  15   AIC:                             57.08
Df Residuals:                      13   BIC:                             58.50
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -87.5167      5.937    -14.741      0.000    -100.343     -74.691
height         3.4500      0.091     37.855      0.000       3.253       3.647
```

从上面回归的参数可以写出回归模型，就是 weight = - 87.5167 + 3.45*Height。对这个模型的解释是截距不可能为 0，因为身高不可能为 0，因此截距项没有物理意义，仅仅是常量调整项，这是因为身高和体重不是同一个量纲，故要有调整项。回归系数(3.45)显著不为 0，因为 t 统计量值的绝对值为 37.9，而 Pr 很小，一般小于 0.05 即认为系数检验显著了(拒绝假设系数参数为 0)。很显然回归系统取零和不取零进行 t 检验，检验结果是拒绝取零和不取零没有差异，即认为回归系数参数取值是可以接受的，这意味着每增加 1 英寸，体重增加 3.45 磅是可信的。R-squared 为 0.991，这个指标是 1-SSE/SST，就是 1 减去估计误差平方和/均值误差平方和，该值在 0 和 1 之间，越大说明估计的越好。这个模型可以使用图形绘制出来，具体代码是：

```
import matplotlib.pyplot as plt
plt.scatter(women["height"], women["weight"], color="blue", label="Data")
# 下面直接调用拟合的结果 model 的函数 predict()实现预测
plt.plot(women["height"], model.predict(x), color="red", label="Fitted line")
plt.xlabel("Height (inches)")
plt.ylabel("Weight (lbs)")
plt.title("Linear Regression: Weight ~ Height")
plt.legend()
plt.show()
```

注意 model.predict(x)函数实现线性回归的泛化功能。具体绘制结果读者可以自行绘制。这个模

型只有一个变量，而且是线性的，也就是一条直线来拟合离散点，如果使用曲线拟合，那也是线性回归，比如使用多项式回归。

二、回归诊断

1. 回归诊断的典型方法

OLS 回归模型模型是否正确可靠，需要回归诊断。主要是线性回归需要满足上面提到的四个假设条件，如果不满足，模型可能就有问题。线性回归以后返回的结果，R 语言使用 plot 可以直接绘制出四幅图形，分别是拟合残差图，正态 QQ 图，尺度位置图，残差杠杆图。但是 Python 语言需要自己绘制，本教材提供函数 plot_lm_diagnostics(model)类似 R 语言的 plot()函数，此处仍然以体重和身高的线性回归为例，绘制这四种图形，具体代码如下，生成的图形如图 13.2 所示。

```
import WugangFunction as wuFunction
wuFunction.plot_lm_diagnostics(model) # model 就是上面线性回归的结果。
```

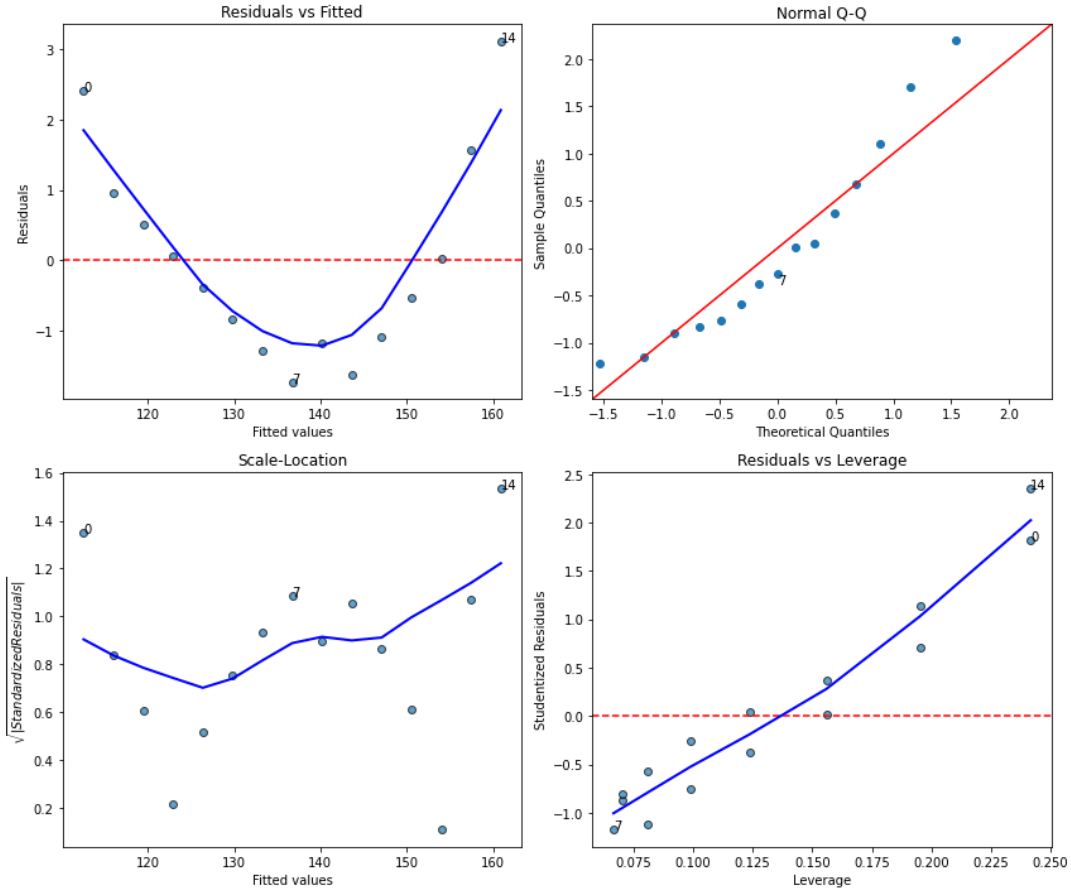


图 13.2 回归诊断图形

从图 13.2 可以观察到 Resuduals vs Fitted 图中拟合值与残差之间的关系，显然它们是曲线关系，不是线性关系，这说明可能需要在回归模型中加一个二次项，以提高拟合精度；Normal Q-Q 图是标准残差对期望理论值的概率图，满足正态分布的图形点应该在一条 45 度角的直线上，从图上可以发现的确如此，这也就说明**因变量误差成正态分布**，其残差值是一个均值为 0 的正态分布。Scale-Location 图可以用来**检验是否方差同性**，如果满足，水平线点周围应该随机分布，现在发现整体有这个性质。最后一幅是残差杠杆图，用来关注单个点的信息，从图形鉴别离群点，高杠杆点以及强影响点的情况。此处第 14 点可能就是强影响点，可以考虑删除等操作。读者可以添加二次项进行线性回归，然后进行回归诊断试一试。需要注意的所有的线性回归，绘制图 13.2 中的四种图形的类型，为线性回归的改进和评估提供依据。

2. 多项式线性回归

上面回归诊断中，数据集 women 体重和身高回归的拟合残差图不在一条直线上，说明拟合精度不够，我们可以增加一个二次项来提高回归精度，具体代码就是

```
# 重置 x 的数值
x = women["height"]
# 构造二次多项式项
X_poly = pd.DataFrame({
    "height": x,
    "height2":x**2
})
# 加常数项（截距）
X_poly = sm.add_constant(X_poly)
# OLS 拟合
model = sm.OLS(y, X_poly).fit()
print(model.summary())
# 绘制拟合曲线
X_fit = np.linspace(58, 72, 200)
X_fit_poly = pd.DataFrame({
    "height": X_fit,
    "height2": X_fit**2
})
X_fit_poly = sm.add_constant(X_fit_poly)
y_pred = model.predict(X_fit_poly)
plt.scatter(x, y, color="black", label="Data")
plt.plot(X_fit, y_pred, color="red", label="Quadratic fit (OLS)")
plt.xlabel("Height")
plt.ylabel("Weight")
plt.legend()
plt.show()
```

上面代码中的二次多项式的实现是通过直接对原始数据进行运算完成的。运行结果发现 R-squared 值为 0.999，这说明拟合的非常好。我们可以本教材提供的函数在一个画板上查看回归诊断的结果，核实拟合残差是否大体在一条水平线上，读者可以自行运行查看。

我们使用的多项式回归仍然是线性回归，这里面关键之处在于回归系数都是线性的，没有出现平方等项，因此可以看出线性回归的线性是针对回归系数而言的，这是因为因变量仍然和自变量的函数呈线性关系，比如$y = klog(x)$，那就是 y 和 log(x)呈线性变化，当然和 x 不是线性变化的，这一点需要在概念上弄清楚。事实上，在使用最小二乘法求解系数参数时，都是求解关于系数参数的线性方程组。如果回归方程是 $y = ae^{bx}$，其中a,b是系数参数，那么就不是线性回归了，因为系数参数出现在指数项上面了。

3．线性回归的改进措施

根据残差杠杆图识别出离群点、强影响点和高杠杆点，可以通过删除和替换等方法改善线性回归模型。当模型不符合正态性、线性或者方差同性假设时，可以通过变量替换加以解决，这方面内容可以进一步参考相关书籍。

三、逐步线性回归

1. 逐步线性回归的含义

在使用线性回归的过程中，有一个有趣的问题存在，那就是分析一个问题时，找了一大堆指标，而且观察了很多指标对应的样本数，现在就希望使用线性回归进行建模了，但是这些指标真的对问题有影响吗？如果没有影响，就使用线性回归来建模，显然是不可能真正解决问题的。例如上一节提到的数据集 state_x77 中，某国犯罪率和各地区的天气之间实际上没有直接关系，如果仍然参与线性回归，就会造成对其它指标的干扰，使犯罪率问题的研究失真。于是一个显而易见的问题就是要去除那些对问题没有影响或者影响很小的指标，这就是涉及到预测变量的挑选问题，也就是只挑选对因变量有较大影响的预测变量作为回归的自变量。另一方面去除影响不大的预测变量，使得回归模型变得更为简洁同样也很重要，这就是逐步线性回归要解决的问题。逐步线性回归分为向前逐步回归，向后逐步回归，顾名思义，向前逐步回归就是每次增加预测变量到模型中，直到再添加预测变量不会使模型有所改进，向后逐步回归恰恰相反，刚开始所有预测变量都参与回归建模，然后逐步删除直到再继续删除预测变量会降低模型质量为止。

2. 逐步线性回归的 Python 语言实现

Python 语言的包 statsmodels.api 的 OLS.fit 生成 AIC 值。AIC(Akaike Information Criterion)指的是衡量统计模型拟合优良性的一种标准，它考虑了模型的统计拟合度以及拟合参数数目，AIC 值小的模型要优先选择。回归后的 model.aic 提供模型的 AIC 值来实现这种模型比较。下面代码是数据集 state_x77 中犯罪率分析的回归模型 AIC 值比较代码如下

```
import WugangFunction as wuFunction
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/state_x77.csv"
state_x77 = wt.read_csv_with_headers(url)
Murder = state_x77["Murder"]
fourData = state_x77[["Population","Illiteracy","Income","Frost"]]
twoData  = state_x77[["Population","Illiteracy"]]
model1 = sm.OLS(Murder, sm.add_constant(fourData)).fit()
# 模型 2: x1 + x2
model2 = sm.OLS(Murder, sm.add_constant(twoData)).fit()
print("Model1 AIC:", model1.aic)
print("Model2 AIC:", model2.aic)
```

比较结果如下

```
Model1 AIC: 239.642948715101
Model2 AIC: 235.65652123237624
```

可以发现第一个回归模型的 AIC 值是 239.6429，第二个回归模型为 235.6565。显然第 2 个回归模型的 AIC 值更小，也就是第二个回归模型更好，这说明将预测变量“收入”和“天气”去除建立的线性回归模型更好。逐步线性回归是自动变量挑选，不用人为去选择预测变量，具体 python 语言实现代码如下

```
import WugangFunction as wuFunction
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/state_x77.csv"
state_x77 = wt.read_csv_with_headers(url)
#注意两个中括号即提取数据再形成 dataframe 类型数据
x_state_x77 = state_x77[["Population","Illiteracy","Income","Frost"]]y_state_x77 =  state_x77["Murder"]
# 返回的是一个字符串数组，因此下面加上中括号可以直接提取数据。

selected_features,aicValue=wuFunction.stepwise_selection_aic(x_state_x77,y_state_x77)
print("Selected features:", selected_features)
print("逐步回归的 AIC 值:")
print(aicValue)
# --- 拟合最终模型 ---
final_model = sm.OLS(y_state_x77, sm.add_constant(state_x77[selected_features])).fit()
print(final_model.summary())
```

本自编程序显示的不够完善，只显示 AIC 值，下面是 R 语言中 MASS 包的 stepAIC 运行结果，更清晰的显示了逐步回归过程。为了方便说明，此处给以列出。

```
Start: AIC=97.75
Murder ~ Population + Illiteracy + Income + Frost
                 Df    Sum of Sq      RSS      AIC
- Frost           1         0.02   289.19    95.75  (说明去掉 Frost 时，AIC 最小为 95.75)
- Income          1         0.06   289.22    95.76
<none>                             289.17    97.75  (都不去掉时，AIC 为 97.75)
- Population      1        39.24   328.41   102.11
- Illiteracy      1       144.26   433.43   115.99  (去掉 Illiteracy 时，AIC 最大)

Step: AIC=95.75
Murder ~ Population + Illiteracy + Income (下面就是去掉 Frost 以后继续做逐步回归)
                Df      Sum of Sq      RSS      AIC
- Income         1           0.06   289.25    93.76 (去掉 Income 时，AIC 最小)
<none>                              289.19    95.75
- Population     1          43.66   332.85   100.78
- Illiteracy     1         236.20   525.38   123.61

Step:  AIC=93.76
Murder ~ Population + Illiteracy
               Df    Sum of Sq       RSS       AIC
<none>                           289.25     93.76  (不去掉任何一个自变量时候，回归的 AIC 最小)
- Population    1       48.517    337.76     99.51
- Illiteracy    1      299.646    588.89    127.31

Call:
lm(formula = Murder ~ Population + Illiteracy, data = states)
Coefficients:
(Intercept)     Population      Illiteracy
```

```
1.6515497          0.0002242          4.0807366
```

上面逐步回归的步骤中，每一个预测变量对应的 AIC 值指的是删除掉该预测变量后回归模型的 AIC 值，<none>对应的是没有删除预测变量时回归模型的 AIC 值。很显然第一步回归时，<none>的值为 97.75。删除掉 Frost 时，回归模型的 AIC 值最小，为 95.75，因此应该选择删除 Frost 这个预测变量，同样第二步应该删除 Income，这样会使得回归模型 AIC 变为 93.76，而如果再删除其它预测变量会使得回归模型的 AIC 值变大，因此逐步回归结束，最终回归的结果只有“人口”和“文化程度”被选为预测变量参与回归建模。

第三节 非线性拟合

上面两节讨论的都是线性回归，也就是拟合参数和因变量都是线性关系，但是有时给定的数据集不能使用线性回归建模，而使用多项式回归建模会使得模型复杂或者不稳定，因此可能使模型参数位于指数或者其它函数内建立的拟合函数更为有效。这时候建立的模型主要是希望精确描述数据集，能够实现对数据的泛化和内插。Python 语言中 scipy.optimize.curve_fit()函数能够实现这种非线性拟合，该函数具体格式如下

```
scipy.optimize.curve_fit(formula, xdata, ydata, p0=None, sigma=None, absolute_sigma=False,
                         check_finite=True, bounds=(-inf, inf), method=None, jac=None, **kwargs)
```

其中 formula 是选择的数学模型公式，xdata 和 ydata 是拟合的数据，p0 是初始值。下面是一元函数拟合的例子，如列表 13.2 所示。

列表 13.2 一元函数的非线性拟合示例

```
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

# --- 构造数据 ---
np.random.seed(0)
x = -np.arange(1, 101) / 10          # 对应 -(1:100)/10
y = -100 + 10 * np.exp(x / 2) + np.random.normal(0, 0.1, size=x.shape)  # 添加噪声

# --- 定义拟合函数 ---
def model(x, c, a, b):
    return c + a * np.exp(b * x)

# --- 非线性拟合 ---
popt, pcov = curve_fit(model, x, y, p0=[-100, 10, 0.5])  # 初始值 p0 可以提高拟合稳定性
c_fit, a_fit, b_fit = popt

print("拟合参数:")
print(f"c = {c_fit:.4f}, a = {a_fit:.4f}, b = {b_fit:.4f}")

# --- 绘图 ---
# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.scatter(x, y, label='数据', s=10)
x_fit = np.linspace(x.min(), x.max(), 200)
y_fit = model(x_fit, *popt) #将拟合的参数传入，返回计算的值
plt.plot(x_fit, y_fit, color='red', label='拟合曲线')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

p0 初始值列表默认为 None，如果拟合失败需要调整初始值，本例子选择的初始值列表为[-100, 10, 0.5]，也就是三个参数 c,a,b 的初始值，显示的图形如图 13.3 所示。

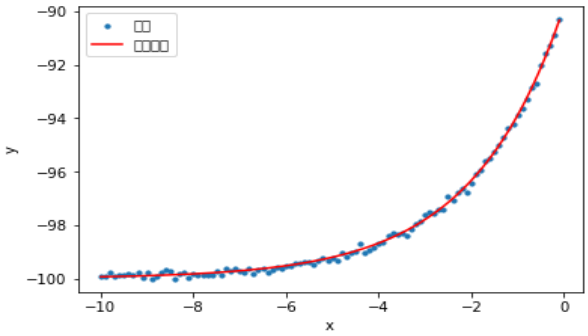


图 13.3 非线性拟合的结果

显示拟合的结果列表 13.3 所示

列表 13.3 非线性拟合结果

```
拟合参数估计与统计量:
  c =   -99.9871   std.err =     0.0186   t = -5364.3236   p = 0.0000e+00
  a =    10.1524   std.err =     0.0461   t =   220.3012   p = 0.0000e+00
  b =     0.5095   std.err =     0.0048   t =   106.0185   p = 0.0000e+00

拟合优度:
RSS  = 0.9097
R^2  = 0.9985
自由度 dof = 97, MSE = 0.009378
```

可以观察到拟合的效果很好。可以使用这个非线性模型来表达该数据集。事实上，列表 13.3 正是使用 c=-100，a=10，b=0.5 生成的数据集合。从列表 13.3 可以发现非线性拟合获得的参数分别为 c=-99.9871，A=10.1524，B=0.5095，它们已经十分接近原始数值，从 R^2 接近 1 也可以发现拟合效果很好。另外注意的就是 model 函数的参数是通过 p0 给出参数列表的初始值，使得函数知道哪些是参数，哪些是自变量。另外，非线性优化一般都是给定初始值，然后使用迭代算法逐步求得最优解，但是初始值如果选取的不好，可能造成耗费函数陷入局部最优解或者出现奇异值，因此可以通过改变初始值克服这些问题。具体这方面的知识可以参考有关最优化理论或者运筹学方面的书籍。

第四节 综合实验十

一、实验目的

1. 能够使用 Python 语言进行基本的统计检验
2. 掌握线性回归的 Python 语言实现以及回归检测和回归模型的改进
3. 掌握数据集非线性拟合的 Python 语言实现

二、实验平台和软件

1. 互联网平台
2. 计算机系统
3. Spyder 环境和 python 语言编译系统

三、实验内容和步骤

1. 基本统计检验方法

(1) 对数据集 Arthritis 关于 Treatment 和 Improved 生成列联表，关于 Improved 和 Sex 生成列联表

(2) 使用卡方检验分析 Treatment 和 Improved 以及 Improved 和 Sex 的独立性问题。

(3) 使用 UScrime 数据集对某国南方地区和被监禁概率的关系进行 t 检验，并分析检验结果。

(4) 分析卡方检验和 t 检验的使用条件

2. 线性回归分析

(1) 使用数据集 women，对 weight 关于 height 做线性回归

(2) 对回归结果进行分析，查看回归系统是否显著

(3) 进行回归诊断，如果回归效果不好，给出进一步改进措施

(4) 对数据集 state_x77 中的 Murder 关于 Population, Illiteracy, Income 和 frost 进行逐步线性回归，给出最优回归模型；

3. 非线性回归分析

(5) 对于函数$y = c + be^{ax}$,其中 x 是自变量，y 是因变量，a,b,c 是参数。给出 x=-10:0:0.5，生成对应的 y 值，使用非线性拟合函数拟合生成的数据集，选取的非线性函数模型可以是原函数，也可以是多项式函数

(6) 分析非线性拟合的结果。

四、实验报告(总分 10 分)

1.对数据集 state_x77 中的 Murder 关于 Population, Illiteracy, Income 和 frost 进行多元线性回归，并进行回归诊断和分析，说明对因变量和预测变量进行统计检验的结果 (4 分)

2.给定一个数据集如下，其中因变量是药物扩散的浓度，自变量是时间

```
timeData = [0,2,4,6, 8,24,49,72,96]
sampleData = [0, 0.0682, 0.0816, 0.0999, 0.1182, 0.1593, 0.1809, 0.2139, 0.2258]
```

请绘制其散点图，根据散点图特征，至少选用三个数学模型对该数据集进行线性和非线性拟合，并分析和评价拟合结果(可以是多项式函数，指数函数，有常数项的幂函数以及对数函数等，注意对数自变量不能取值为 0，因此拟合表达式中可以考虑自变量+1) (5 分)

3.记录实验出现的问题，分析并给出最后解决办法 (1 分)