2.7.Python编程挑战

▲2.6.统计学面临的挑战

▼2.8.继续学习本章知识

Python编程挑战

【分析对象】 txt文件—文件名为“Pearson.txt”，数据内容来自卡尔·皮尔逊（Karl Pearson）在 1903 年对 1078 对夫妇的观察，取父亲的身高为自变量，取他的成年儿子的身高为因变量（注：数据文件见本书配套资源）。

【分析目的与任务】 理解统计建模在数据科学中的应用—以一元线性回归方法进行分析。

首先，取父亲的身高为自变量，取他的成年儿子的身高为因变量。 其次，训练模型并查看其统计量。 接着，评价模型拟合优度及假设检验。最后，将模型用于预测新数据。

【方法及工具】Python 及 statsmodels 包。

【主要步骤】要实现上述目的主要步步如下：数据读入、数据理解、数据准提、模型类型的选择与超级参数的设置、训练具体模型及查看其统计量、拟合优度评价、建模前提假定的讨论和模型的应用。

Step 1：数据读入

- 确前工作目录是 Python 数据分析中常用的基本概念，它代表的是 Python 解释器直接读写数据文件的路径。通常，我们采用 Python 模块 os 中的函数 chdir()和 getcwd() 分别进行确前工作目录的查看和更改操作。
- 此外，按照 Python 代码编写的惯例，在程序开始处本例所需的 Python 包—Pandas、Numpy 和 matplotlib.pyplot 也导入到确前会话之中，并在后续代码中分别用于数据框（关系表）处理、矩阵计算和数据可视化。
- 本例所涉及的数据文件为“Pearson.txt”，读者可以从本书配套资源中找到此文件并复制到自己的确前工作目录。
- 示例如下。

（1）导入所需 Python 包

```
In[1] #导入所需 Python 包

      import numpy as np
      import matplotlib.pyplot as plt
      import os

      #更改当前工作目录
      os.chdir(r'C:\Users\soloman\clm')
      【提示】此处，路径“C:\Users\soloman\clm”是示例，读者可自行设置其他路径
      #查看当前工作目录
      print(os.getcwd())
```

对应输出结果为：

```
C:\Users\soloman\clm
```

（2）显示数据

接着，用 Python 第三方包 Pandas 提供的函数 read_table()将确前工作目录中的数据文件 Pearson.txt 读入至 Pandas 数据框 df_heights 中，并用 Pandas包中的函数 head()或 tail()显示数据框 df_heights 的前 5 行或后 5 行数据。示例如下。

```
In[2] #”
      #从当前工作目录读入数据文件  Pearson.txt
      df_heights = pd.read_table("Pearson.txt")
      import os

      #查看数据框 df_heights 的后 5 行
      df_heights.tail()
      #【提示】默认情况下，tail()函数显示的是数据框的后 5 行
```

对应输出结果为：

Father	Son
107367.0	70.8
107471.3	68.3
107571.8	69.3
107670.7	69.3
107770.3	67.0

Step 2：数据理解

- **数据理解**是执行数据科学项目的前提。
- 数据理解以**业务理解**（ Business Understandings）为前提。由于本例所涉及的业务非常简单，因而不再赘述其业务理解问题。
- 常用的**数据理解方法**有查看数据形状、显示列名或模式信息、查看描述性统计 信息等。
- 示例如下。

数据理解方法

In[3]
```
# 查看数据形状
df_heights.shape
```

对应输出结果为：

```
(1078,2)
```

In[4]
```
# 显示列名或模式信息
print(df_heights.columns)
```

对应输出结果为：

```
Index(['Father', 'Son'], dtype='object')
```

In[5]
```
#查看描述性统计信息
df_heights.describe()
```

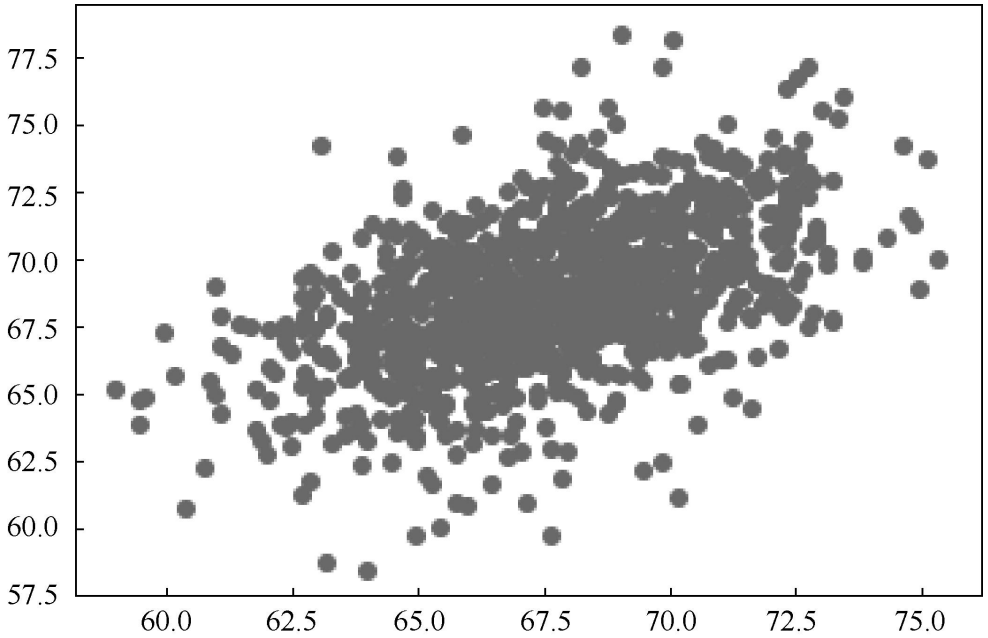
对应输出结果为：

	Father	Son
count	1078.000000	1078.000000
mean	67.686827	68.684230
std	2.745827	2.816194
min	59.000000	58.500000
25%	65.800000	66.900000
50%	67.800000	68.600000
75%	69.600000	70.500000
max	75.400000	78.400000

数据理解方法

```
In[6]  # 采用数据可视化方法理解数据
       %matplotlib inline
       plt.scatter(df_heights['Father'], df_heights['Son’])
       plt.show()
```

对应输出结果为：



Step 3：数据准备

- 在基于 Python 的统计建模中，人们通常采用 Python 第三方包 statsmodels、statistics 和 sciKit-learn 等实现**数据建模**的目的。
- 值得一提的是，上述统计建模的 Python 包通常对所需建模的数据模态均有特殊要求—需要事先将原始数据集分解成**自变量和因变量**。其中，自变量需要用“特征矩阵”表示，因变量需要要“目标向量”表示。
- 为此，我们需要对**数据框 df_heights 进行数据准提、数据预处理**工作。
- 示例如下。

数据准备

```
In[7]: #准备线性回归的特征矩阵（X）和目标向量(y)

       x= df_heights['Father’]
       y = df_heights['Son‘]
       X[:10]
```

对应输出结果为：

0	65.0
1	63.3
2	65.0
3	65.8
4	61.1
5	63.0
6	65.4
7	64.7
8	66.1
9	67.0

Name:Father,dtype:float64

```
In[8]: y[:10]
```

对应输出结果为：

0	59.8
1	63.2
2	63.3
3	62.8
4	64.3
5	64.2
6	64.1
7	64.0
8	64.6
9	64.0

Name:Son ,dtype:float64

Step 4：模型类型的选择与超级参数的设置

- 从本例题 Step 2 的数据可视化结果中可看出，我们可以尝试采用简单线性回归分析方法进行建模。为此，我们导入常用于**统计建模**的 Python 第三方包 statsmodels。
- 示例如下。

```
In[9] | #导入 Python 第三方包 statsmodels
      | import statsmodels.api as sm
```

- 接下来，采用 statsmodels 包提供的函数 OLS()进行简单线性回归。需要注意的是， 在默认情况下，OLS()函数的训练结果中并不含截距项（Intercept）。
- 如果在数据建模中需要截距项，必须进一步预处理数据，即在自变量所对应的特征矩阵中增加这样一个**特殊的预定义列**—列名为 const，该列上的每一行的取值均为1。
- 为此,statsmodels 包提供了另一个函数.add_constant()。我们通过调用.add_constant()函数可以很轻松地实现向特征矩阵插入 const 列。

（1）向特征矩阵插入 const 列

In[10]:

```
#向特征矩阵插入 const 列
X_add_const=sm.add_constant(X)

#【提示】给 X 新增一列，列名为 const，每行取值 1.0
#【注意】建议不要将此行代码写成 “X=sm.add_constant(X)”，否则，第一次执行该 cell 时才能得到正确结果；当多次执行时，每次执行该 cell 时 X 的值发生改变
#显示插入 const 列后的自变量
X_add_const.head()
```

对应输出结果为：

	const	Father
0	1.0	65.0
1	1.0	63.3
2	1.0	65.0
3	1.0	65.8
4	1.0	61.1

（2）将超级模型和数据集提交给 Python 解释器

最后，通过调用 statsmodels 包提供的 OLS()函数，对已预处理好的数据集（此处 自变量为 X_add_const 和因变量 y）进行**简单线性回归**，并存放在 Python 对象 myModel 中，以便后续对其进行拟合和预测活动。
示例代码如下。

```
In[11] | #将超级模型和数据集提交给 Python 解释器
       | myModel = sm.OLS(y, X_add_const)
```

Step 5：训练具体模型及查看其统计量

需要注意的是，运行本例 Step 4 中的代码行 “myModel = sm.OLS(y, X_add_const)”时，并未完成线性回归的拟合操作，拟合操作的执行需要调用 statsmodels 提供的另一个函数 **fit()** 实现。

拟合完成后，可以用 statsmodels 提供的函数 summary()查看拟合结果。

示例如下。

In[12]:
```
#模型拟合
results = myModel.fit()

#查看拟合结果
print(results.summary())
```

OLS Regression Results

Dep. Variable:	Son	R-squared:	0.251		
Model:	OLS	Adj. R-squared:	0.250		
Method:	Least Squares	F-statistic:	360.9		
Date:	Wed, 18 Dec 2019	Prob (F-statistic):	1.27e-69	Time:	
	11:01:32	Log-Likelihood:	-2489.4		
No. Observations:	1078	AIC:	4983.		
Df Residuals:	1076	BIC:	4993.		
Df Model:	1				
Covariance Type:	nonrobust				

	coef	std err	t	P>|t|	[0.025	0.975]
const	33.8928	1.833	18.491	0.000	30.296	37.489
Father	0.5140	0.027	18.997	0.000	0.461	0.567

Omnibus:	17.527	Durbin-Watson:	0.765
Prob(Omnibus):	0.000	Jarque-Bera (JB):	30.642
Skew:	-0.052	Prob(JB):	2.22e-07
Kurtosis:	3.819	Cond. No.	1.67e+03

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.67e+03. This might indicate that there are strong multicollinearity or other numerical problems.

查看统计量

在 summary()函数的返回结果中可以找到，统计建模结果的关键参数，如拟合系数（Coef）、拟合优度（R 方，R-squared）、F 值（F-statistic）、DW 统计量（Durbin-Watson） 和 JB 统计量(Jarque-Bera)及它们的 P 值。确然，我们可以用 statsmodels 包提供的其他函数或属性，分别显示上述参数。以查看回归系数，即斜率和截距项为例。
示例如下。

```
In[13]: #查看斜率和截距项
        results.params
```

对应输出结果为：

const	33.892801
Father	0.514006
dtype: float64	

Step 6： 拟合优度评价

除了调用 summary()函数外，我们还可以用 statsmodels 包提供的 rsquared 属性查看拟合结果的优度—R方。

In[14]:
```
#【显示】 R 方（决定系数）
results.rsquared

#【提示】R 方的取值范围为[0,1]，越接近 1，说明“回归直线的拟合优度越好”。
```

对应输出结果为：

```
0.25116403263425147
```

Step 7：建模前提假定的讨论

在基于统计学方法完成数据分析、数据科学任务时，不仅要进行模型优度的评价，而且还需要重点分析统计分析方法的应用前提假定是否成立。因为统计分析都是建立在一个或多个“**前提假定条件**”之上。

以简单线性回归为例，其“前提假定条件”有：

（1）x 和 y 之间存在线性关系→检验方法为计算“F 统计量”，示例如下。

In[15]:

```
#查看 F 统计量的 P 值
results.f_pvalue

#【提示】F 统计量的 p 值——用于检验【X 和 y 之间是否存在线性关系】。自变量（X）和因变量（y）之间存在线性关系是“线性回归分析”前提假设之一。
#【提示】查看 P 值是否小于 0.05。
```

对应输出结果为：

```
1.2729275743657959e-69
```

（1）前提假定条件

（2）残差项（的各期）之间不存在的自相关性→检验方法为计算“Durbin-Watson 统计量”，示例如下。

In[16]

```
# 查看 Durbin-Watson 统计量
sm.stats.stattools.durbin_watson(results.resid)

#查看 Durbin-Watson 统计量 sm.stats.stattools.durbin_watson(results.resid)
#【提示】通常，Durbin-Watson 统计量用于检查“残差项之间不存在的自相关性”，残差项（的各期）之间相互独立是线性回归分析的另一个前提假定
```

对应输出结果为：

```
0.764609072811116
```

（1）前提假定条件

（3）残差项为正态分布的随机变量—>检验方法为计算“JB统计量”，示例如下。

In[17]

```
#查看 JB 统计量及其 P 值
sm.stats.stattools.jarque_bera(results.resid)

# 【提示】此函数的返回值有 4 个，分布为 JB 值，JB 的 P 值，峰度和偏度
# 【思路】JB 统计量用于检验“残差项为正态分布”，残差项属于正态分布是线性回归
  分析的前提假设之一
```

对应输出结果为：

```
(30.64219867294703,
2.2188661329538247e-07,
-0.05180726198181561,
3.819429749308355)
```

（2）用新模型 results 进行预测

模型优度和模型前提假定均成立时，我们可以通过调用 statsmodels 包提供的 predict()函数实现“**模型预测**”的目的。需要注意的是，predict()函数的参数为空时， 训练数据为自变量（X）预测对应的因变量（y）。示例如下。

```
In[18]  #用新模型 results 进行预测

        y_predict=results.predict()
        y_predict
```

对应输出结果为：

```
array([67.30318486,66.4293748,
67.30318486, ..., 70.79842506,
70.23301856, 70.02741619])
```

Step 8：模型的应用

- 通常，模型的应用主要采用 statsmodels 包提供的 **predict()函数或 tranform()函数**实现。
- 二者的区别在于，predict()函数另生成一个数据对象来记录目标向量（y），而 tranform()函数直接替换原有数据框中的目标向量（y）的值。
- 需要注意的是，predict()函数和 tranform()函数的实参必须为训练该模型是的自变量的数据模型一致。
- 例如，本例题中，特征矩阵中增加了一个常数项列 const，取值为 1。

（1）可视化显示回归效果

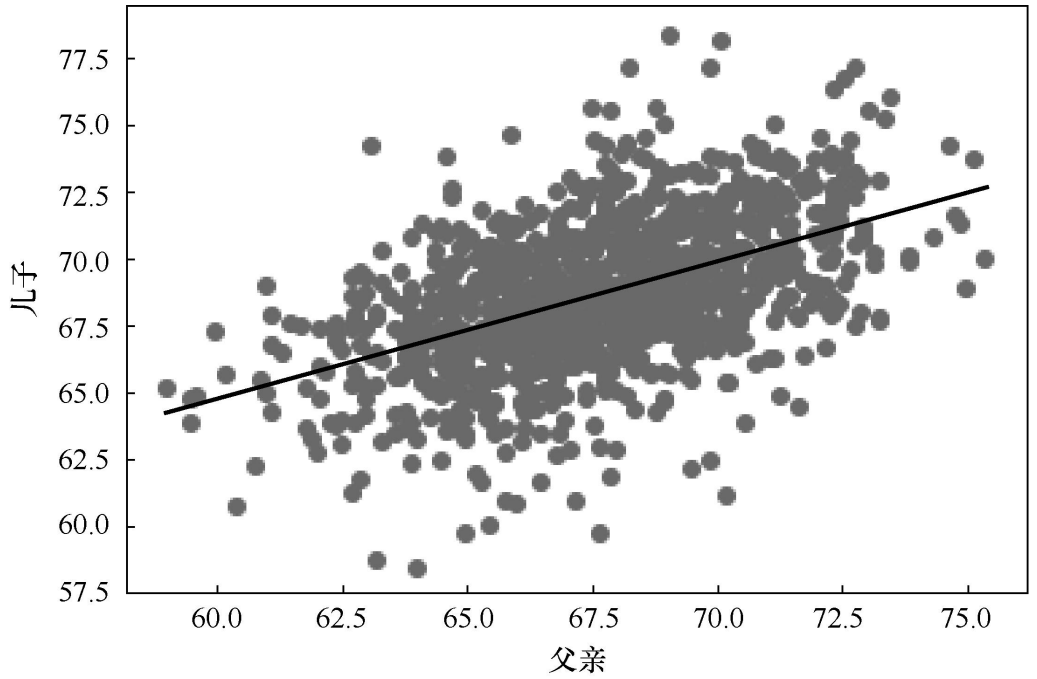
In[19]

```
#可视化显示回归效果

plt.rcParams[‘font.family’]=simHei‘
plt.plot(df_heights['Father'],df_heights['Son'],'o‘)
plt.plot(df_heights['Father'],y_predict)
plt.title('父亲身高与儿子身高的线性回归分析’)
plt.xlabel('父亲‘)
plt.ylabel('儿子')
```

对应输出结果为：

Text(0,0.5,'儿子')



（2）模型预测

最后，将模型应用于训练集和测试集之外的“新数据”，父亲的身高 70 英寸时，预测其儿子的身高。

In[20]
```
# 模型预测
h = 70
results.predict([1,h])
```

对应输出结果为：

```
array([69.87321442])
```

可见，父亲的身高为 70 英寸时，该模型预测的其儿子身高为 69.87321442 英寸。