

8.2.2 线性回归的实现

2023 年 9 月 8 日

线性回归的 Python 实现

1. 相关性系数的实现

调用 `scipy.stats.pearsonr` 计算相关系数：

```
import scipy.stats as stats
r = stats.pearsonr(x1, x2)
```

也可以使用 `pd.DataFrame.corr()` 来计算相关系数矩阵，然后绘制图形，详细见下一节。

```
df = pd.DataFrame([(0.2, 0.3), (0.0, 0.6), (0.6, 0.0), (0.2, 0.1)],
                  columns=['dogs', 'cats'])

df.corr(method='pearson')
      dogs  cats
dogs    1.0   0.3
cats    0.3   1.0
```

2. 线性回归的实现

线性回归的 Python 实现主要使用 `sklearn` 机器学习库 `linear_model` 模块下的 `LinearRegression` 类。

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True, copy_X=True, n_jobs=None, p
```

类别	名称	含义
参数	<code>fit_intercept</code>	默认为 <code>True</code> ，是否计算模型的截距，如果设为 <code>False</code> ，是指数据是以原点为中心的，不会计算截距。
属性	<code>coef_</code>	线性回归问题的估计系数。

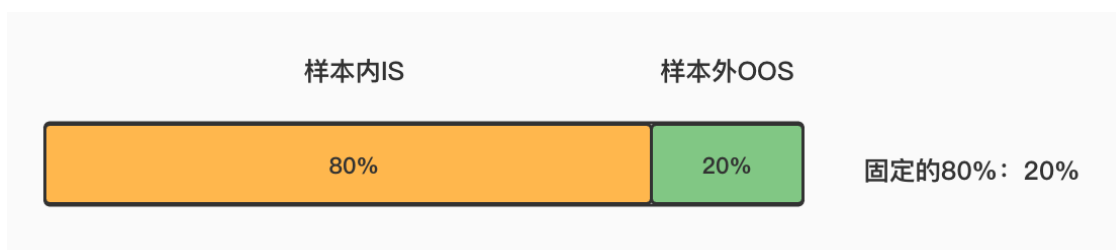
类别	名称	含义
方法	<code>intercept_</code>	线性模型中的独立项，也就是截距
	<code>fit(X, y)</code>	训练模型（或称估计器、学习器）
	<code>predict(X)</code>	在训练后，使用模型预测
	<code>score(X, y)</code>	用来计算模型的精度
	<code>get_params()</code>	获得模型的参数

模型输入 X 和 y

上述回归模型中使用 X 和 y 的数据结构如下， X 包含多个样本，以及每个样本的属性，也就是自变量，和 X 的每个样本对应的就是我们的预测目标 y ，也就是因变量。

在实际编程中，一般使用 `pd.DataFrame` 来表示 X 和 y 。

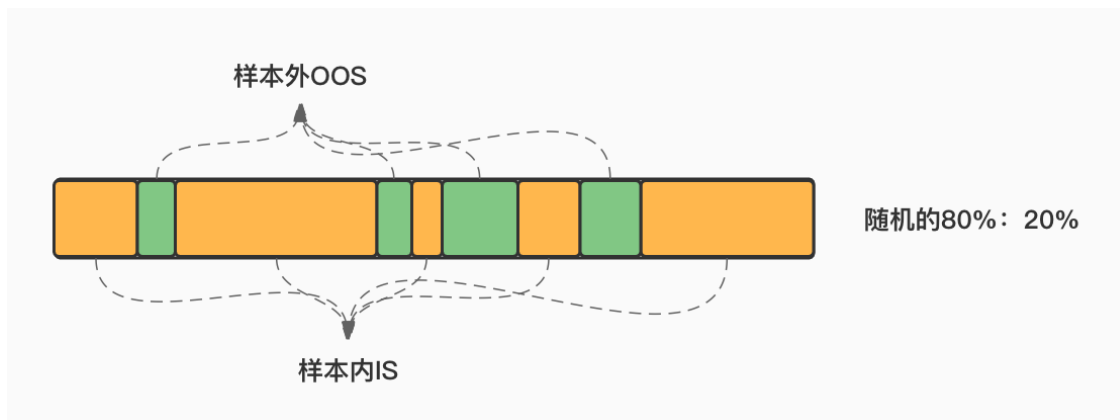
	自变量1	自变量2	自变量3		因变量
样本1				样本1	
样本2		X		样本2	y
样本3				样本3	
...				...	



一般将 80% 的原始数据集的子样本集作为样本内 (In-sample, IS)，剩余 20% 作为样本外 (Out-of-sample, OOS)。样本总数为 500 个，一种简单的切割方法是直接指定前 400 个样本为 IS，后面为 OOS，但是这种采样方法存在弊端。

```
[19]: train_x = X.values[:400]
      train_y = y.values[:400]
```

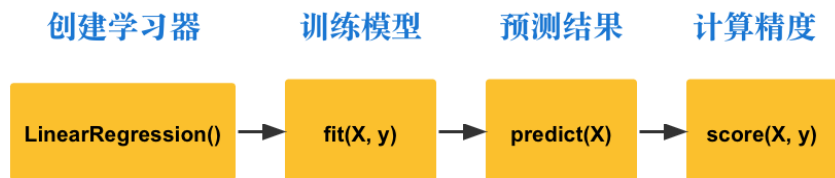
```
test_x = X.values[400:]
test_y = y.values[400:]
```



当我们并不了解原始数据集是不是被刻意排列了，最好的办法是使用随机抽样，即随机抽 80% 为 IS，剩余的 20% 为 OOS。使用 `sklearn.model_selection.train_test_split`

```
[20]: from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X, y, train_size=0.8,
                                                    random_state=42)
```

实现流程



针对于多元回归分析,其一般化的流程如下:

1) 创建学习器，也就是初始化线性回归模型

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

也可以这样表述:

```
from sklearn import linear_model
model = linear_model.LinearRegression()
```

2) 训练模型

```
model.fit(X, y)
```

3) 生成预测结果

```
predicted_y = model.predict(X)
```

4) 计算模型预测精度和拟合优度 R^2

```
precision = model.score(X, y)
```

```
from sklearn.metrics import r2_score
```

```
r2 = r2_score(predicted_y, y)
```

5) 生成汇总信息 (summary)

```
print(" 系数: %s" %model.coef_)
```

```
print(" 截距: %.4f" %model.intercept_)
```

```
print(" 样本内 (IS) 训练集精度: %.2f" %precision)
```

```
print(" 拟合优度 R-squared: %.2f" % r2)
```

参考

1. sklearn 官网: [链接](#)