## 使用 Pandas 进行数据探索

#### 介绍

本次实验通过分析电信运营商的客户离网率数据集来熟悉 Pandas 数据探索的常用方法，并构建一个预测客户离网率的简单模型。

#### 知识点

* 排列
* 索引
* 交叉表
* 透视表
* 数据探索

### Pandas 的主要方法

Pandas 是基于 NumPy 的一种工具，提供了大量数据探索的方法。Pandas 可以使用类似 SQL 的方式对 .csv、.tsv、.xlsx 等格式的数据进行处理分析。

Pandas 主要使用的数据结构是 Series 和 DataFrame 类。下面简要介绍下这两类：

* Series 是一种类似于一维数组的对象，它由一组数据（各种 NumPy 数据类型）及一组与之相关的数据标签（即索引）组成。
  + 通过以为**数组**方式创建
* import numpy as np
* array = np.array([1,2,3,4])
* series1 = pd.Series(array)
* print(series1.values)
* print(series1.index)
* print(series1.dtype)
  + 通过**字典**的方式创建
* series1 = pd.Series({ 'a':1, 'b':2, 'c': 3})
* print(series1.values)
* print(series1.index)
* print(series1.dtype)
* DataFrame 是一个二维数据结构，即一张表格，其中每列数据的类型相同。你可以把它看成由 Series 实例构成的字典。

下面开始此次实验，我们将通过分析电信运营商的客户离网率数据集来展示 Pandas 的主要方法。

首先载入必要的库，即 NumPy 和 Pandas。

**（自行去搜索两个库的安装方法）**

 教学代码：

import numpy as np

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

通过 read\_csv() 方法读取数据，然后使用 head() 方法查看前 5 行数据。

df = pd.read\_csv('telecom\_churn.csv')

df.head()

**注意：下面所有的函数均需要在Python Console下运行才能显示数据处理效果，否则需要借助print（）函数**

（实验报告中给出截图）

我们已经将文件下载，见“telecom\_churn.csv”。

上图中的每行对应一位客户，每列对应客户的一个特征。

接下来让我们查看一下该数据库的维度、特征名称和特征类型。

df.shape

（实验报告中给出截图）

上述结果表明，我们的列表包含 3333 行和 20 列。下面我们尝试打印列名。

df.columns

（实验报告中给出截图给出截图）

我们还可以使用 info() 方法输出 DataFrame 的一些总体信息。

df.info()

（实验报告中给出截图）

bool、int64、float64 和 object 是该数据库特征的数据类型。这一方法同时也会显示是否有缺失值，上述结果表明在该数据集中不存在缺失值，因为每列都包含 3333 个观测，和我们之前使用 shape 方法得到的数字是一致的。

astype() 方法可以更改列的类型，下列公式将 Churn 离网率特征修改为 int64 类型。

df['Churn'] = df['Churn'].astype('int64')

（实验报告中给出截图）

describe() 方法可以显示数值特征（int64 和 float64）的基本统计学特性，如未缺失值的数值、均值、标准差、范围、四分位数等。

df.describe()

（实验报告中给出截图）

通过 include 参数显式指定包含的数据类型，可以查看非数值特征的统计数据。

df.describe(include=['object', 'bool'])

（实验报告中给出截图）

value\_counts() 方法可以查看类别（类型为 object ）和布尔值（类型为 bool ）特征。让我们看下 Churn 离网率 的分布。

df['Churn'].value\_counts()

（实验报告中给出截图）

上述结果表明，在 3333 位客户中， 2850 位是忠实客户，他们的 Churn 值为 0。调用 value\_counts() 函数时，加上 normalize=True 参数可以显示比例。

df['Churn'].value\_counts(normalize=True)

（实验报告中给出截图）

### 排序

DataFrame 可以根据某个变量的值（也就是列）排序。比如，根据每日消费额排序（设置 ascending=False 倒序排列）。

df.sort\_values(by='Total day charge', ascending=False).head()

（实验报告中给出截图）

此外，还可以根据多个列的数值排序。下面函数实现的功能为：先按 Churn 离网率升序排列，再按 Total day charge 每日总话费降序排列，优先级 Churn > Tatal day charge。

df.sort\_values(by=['Churn', 'Total day charge'],

ascending=[True, False]).head()

（实验报告中给出截图）

### 索引和获取数据

DataFrame 可以以不同的方式进行索引。

使用 DataFrame['Name'] 可以得到一个单独的列。比如，离网率有多高？

df['Churn'].mean()

（实验报告中给出截图）

对一家公司而言，14.5% 的离网率是一个很糟糕的数据，这么高的离网率可能导致公司破产。

布尔值索引同样很方便，语法是 df[P(df['Name'])]，P 是在检查 Name 列每个元素时所使用的逻辑条件。这一索引的输出是 DataFrame 的 Name 列中满足 P 条件的行。

让我们使用布尔值索引来回答这样以下问题：离网用户的数值变量的均值是多少？

df[df['Churn'] == 1].mean()

（实验报告中给出截图）

离网用户在白天打电话的总时长的均值是多少？

df[df['Churn'] == 1]['Total day minutes'].mean()

（实验报告中给出截图）

未使用国际套餐（International plan == NO）的忠实用户（Churn == 0）所打的最长的国际长途是多久？

df[(df['Churn'] == 0) & (df['International plan'] == 'No')]['Total intl minutes'].max()

（实验报告中给出截图）

DataFrame 可以通过列名、行名、行号进行索引。loc 方法为通过名称索引，iloc 方法为通过数字索引。

通过 loc 方法输出 0 至 5 行、State 州 至 Area code 区号 的数据。

df.loc[0:5, 'State':'Area code']

（实验报告中给出截图）

通过 iloc 方法输出前 5 行的前 3 列数据（和典型的 Python 切片一样，不含最大值）。

df.iloc[0:5, 0:3]

（实验报告中给出截图）

df[:1] 和 df[-1:] 可以得到 DataFrame 的首行和末行。

df[-1:]

（实验报告中给出截图）

### 应用函数到单元格、列、行

下面通过 apply() 方法应用函数 max 至每一列，即输出每列的最大值。

df.apply(np.max)

（实验报告中给出截图）

apply() 方法也可以应用函数至每一行，指定 axis=1 即可。在这种情况下，使用 lambda 函数十分方便。比如，下面函数选中了所有以W开头的州。

df[df['State'].apply(lambda state: state[0] == 'W')].head()

map() 方法可以通过一个 {old\_value:new\_value} 形式的字典替换某一列中的值。

d = {'No': False, 'Yes': True}

df['International plan'] = df['International plan'].map(d)

df.head()

（实验报告中给出截图）

当然，使用 repalce() 方法一样可以达到替换的目的。

df = df.replace({'Voice mail plan': d})

df.head()

（实验报告中给出截图）

### 分组（Groupby）

Pandas 下分组数据的一般形式为：

df.groupby(by=grouping\_columns)[columns\_to\_show].function()

对上述函数的解释：

* groupby() 方法根据 grouping\_columns 的值进行分组。
* 接着，选中感兴趣的列（columns\_to\_show）。若不包括这一项，那么就会选中所有非 groupby 列（即除 grouping\_colums 外的所有列）。
* 最后，应用一个或多个函数（function）。

在下面的例子中，我们根据 Churn 离网率变量的值对数据进行分组，显示每组的统计数据。

columns\_to\_show = ['Total day minutes', 'Total eve minutes','Total night minutes']

df.groupby(['Churn'])[columns\_to\_show].describe(percentiles=[])

（实验报告中给出截图）

和上面的例子类似，只不过这次将一些函数传给 agg()，通过 agg() 方法对分组后的数据进行聚合。

columns\_to\_show = ['Total day minutes', 'Total eve minutes','Total night minutes']

df.groupby(['Churn'])[columns\_to\_show].agg([np.mean, np.std, np.min, np.max])

（实验报告中给出截图）

### 汇总表

Pandas 中的透视表定义如下：

透视表(Pivot Table)是电子表格程序和其他数据探索软件中一种常见的数据汇总工具。它根据一个或多个键对数据进行聚合，并根据行和列上的分组将数据分配到各个矩形区域中。

通过 pivot\_table() 方法可以建立透视表，其参数如下：

* values 表示需要计算的统计数据的变量列表
* index 表示分组数据的变量列表
* aggfunc 表示需要计算哪些统计数据，例如，总和、均值、最大值、最小值等。

现在，通过 pivot\_table() 方法查看不同区号下白天、夜晚、深夜的电话量的均值。

df.pivot\_table(['Total day calls', 'Total eve calls', 'Total night calls'], ['Area code'], aggfunc='mean')

（实验报告中给出截图）

pivot\_table() 其他的使用方法见 [Pandas 百题大冲关](https://www.lanqiao.cn/courses/1091/labs/6138/document) 的透视表部分。

交叉表（Cross Tabulation）是一种用于计算分组频率的特殊透视表，在 Pandas 中一般使用 crosstab() 方法构建交叉表。

构建一个交叉表查看样本的 Churn 离网率 和 International plan 国际套餐 的分布情况。

pd.crosstab(df['Churn'], df['International plan'])

（实验报告中给出截图）

构建一个交叉表查看 Churn 离网率 和 Voice mail plan 语音邮件套餐 的分布情况。

pd.crosstab(df['Churn'], df['Voice mail plan'], normalize=True)

（实验报告中给出截图）

上述结果表明，大部分用户是忠实用户，同时他们并不使用额外的服务（国际套餐、语音邮件）。

### 增减 DataFrame 的行列

在 DataFrame 中新增列有很多方法，比如，使用 insert()方法添加列，为所有用户计算总的 Total calls 电话量。

total\_calls = df['Total day calls'] + df['Total eve calls'] + df['Total night calls'] + df['Total intl calls']

# loc 参数是插入 Series 对象后选择的列数

# 设置为 len(df.columns)以便将计算后的 Total calls 粘贴到最后一列

df.insert(loc=len(df.columns), column='Total calls', value=total\_calls)

df.head()

（实验报告中给出截图）

上面的代码创建了一个中间 Series 实例，即 tatal\_calls，其实可以在不创造这个实例的情况下直接添加列。

df['Total charge'] = df['Total day charge'] + df['Total eve charge'] + df['Total night charge'] + df['Total intl charge']

df.head()

（实验报告中给出截图）

使用 drop() 方法删除列和行。

# 移除先前创捷的列

df.drop(['Total charge', 'Total calls'], axis=1, inplace=True)

# 删除行

df.drop([1, 2]).head()

（实验报告中给出截图）

对上述代码的部分解释：

* 将相应的索引 ['Total charge', 'Total calls'] 和 axis 参数（1 表示删除列，0 表示删除行，默认值为 0）传给 drop。
* inplace 参数表示是否修改原始 DataFrame （False 表示不修改现有 DataFrame，返回一个新 DataFrame，True 表示修改当前 DataFrame）。

## 预测离网率

首先，通过上面介绍的 crosstab() 方法构建一个交叉表来查看 International plan 国际套餐变量和Churn离网率的相关性，同时使用 countplot() 方法构建计数直方图来可视化结果。

# 加载模块，配置绘图

import matplotlib.pyplot as plt

import seaborn as sns

sns.countplot(x='International plan', hue='Churn', data=df)

（实验报告中给出截图）

结果表明，开通了国际套餐的用户的离网率要高很多，这是一个很有趣的观测结果。也许，国际电话高昂的话费让客户很不满意。

同理，查看 Customer service calls 客服呼叫 变量与 Chunrn离网率的相关性，并可视化结果。

pd.crosstab(df['Churn'], df['Customer service calls'], margins=True)

sns.countplot(x='Customer service calls', hue='Churn', data=df)

上图表明，在客服呼叫 4 次之后，客户的离网率显著下降。

为了更好的突出 Customer service call 客服呼叫和 Churn 离网率的关系，可以给 DataFrame 添加一个二元属性 Many\_service\_calls，即客户呼叫超过 3 次（Customer service calls > 3）。看下它与离网率的相关性，并可视化结果。

df['Many\_service\_calls'] = (df['Customer service calls'] > 3)

pd.crosstab(df['Many\_service\_calls'], df['Churn'], margins=True)

sns.countplot(x='Many\_service\_calls', hue='Churn', data=df)

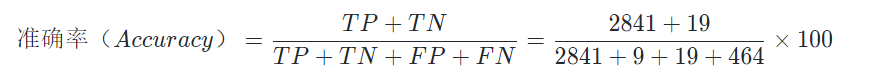
（实验报告中给出截图）

现在我们可以创建另一张交叉表，将 Churn 离网率与 International plan 国际套餐及新创建的 Many\_service\_calls 多次客服呼叫关联起来。

pd.crosstab(df['Many\_service\_calls'] & df['International plan'], df['Churn'])

（实验报告中给出截图）

上表表明，在客服呼叫次数超过 3 次并且已办理 International Plan 国际套餐的情况下，预测一名客户不忠诚的准确率（Accuracy）可以达到 85.8％，计算公式如下：



其中，TP 表示将 True 预测为 True 的数量，TN 表示将 Flase 预测为 Flase 的数量，FP 表示将 Flase 预测为 True 的数量，FN 表示将 True 预测为 Flase 的数量。

复习一下本次实验的内容：

* 样本中忠实客户的份额为 85.5%。这意味着最简单的预测「忠实客户」的模型有 85.5% 的概率猜对。也就是说，后续模型的准确率（Accuracy）不应该比这个数字少，并且很有希望显著高于这个数字。
* 基于一个简单的「（客服呼叫次数 > 3） & （国际套餐 = True） => Churn = 1, else Churn = 0」规则的预测模型，可以得到 85.8% 的准确率。以后我们将讨论决策树，看看如何仅仅基于输入数据自动找出类似的规则，而不需要我们手工设定。我们没有应用机器学习方法就得到了两个准确率（85.5% 和 85.8%），它们可作为后续其他模型的基线。如果经过大量的努力，我们仅将准确率提高了 0.5%，那么我们努力的方向可能出现了偏差，因为仅仅使用一个包含两个限制规则的简单模型就已提升了 0.3% 的准确率。
* 在训练复杂模型之前，建议预处理一下数据，绘制一些图表，做一些简单的假设。此外，在实际任务上应用机器学习时，通常从简单的方案开始，接着尝试更复杂的方案。

### 实验总结

本次实验使用 Pandas 对数据进行了一定程度的分析和探索，交叉表、透视表等方法的运用将使你在数据探索过程中事半功倍。

### 实验报告要求

1. 在《Python简明教程》中学习python基本知识，自行选择在linux或者Winodws系统中配置python环境，并在实验报告中给出配置环境过程（截图）；
2. 按照上述实验步骤，学习和使用数据处理工具pandas，运行上面给出的代码，并在实验报告中给出运行结果的截图；
3. 对test.csv里面的数据进行如下操作：有优先按照attempts排序，当attempts一样，按照name排序，请在实验报告中给出**截图**和**可运行的代码**。
4. 完成实验报告，包含环境配置部分和pandas学习部分，报告中的截图**禁止抄袭**。