# 实验四　Pandas 统计分析

## 1.实验类型

验证型实验

## 2.实验目的和要求

（1） 掌握 csv 数据读取方法。
（2） 掌握 DataFrame 的常用属性与方法。
（3） 掌握 pandas 描述性统计方法。
（4） 掌握时间字符串和标准时间的转换方法。
（5） 掌握时间信息提取的方法。
（6） 掌握时间数据的算术运算。
（7） 掌握分组聚合的原理与步骤。
（8） 掌握 agg／apply 聚合方法。
（9） 掌握 transform 聚合方法。
（10） 掌握透视表的制作方法。

## 3.实验内容

**题目一：读取并查看 P2P 网络贷款数据主表的基本信息**

(1)训练要点

- 掌握 csv 数据读取方法。
- 掌握 DataFrame 的常用属性与方法。
- 掌握 pandas 描述性统计方法。

探索数据的基本信息，洞察数据的整体分布，数据的类属关系，从而发现数据间的关联。

(2)需求说明

P2P 网络贷款主表数据 Training_Master.csv 主要存放了网贷用户的基本信息。探索数据的基本信息，能够洞察数据的整体分布、数据的类属关系，从而发现数据间的关联。

(3)实现步骤

- 使用 ndim、shape、memory_usage 属性分别查看维度、大小、占用内存信息。
- 使用 describe 方法进行描述性统计，并剔除值相同或全为空的列。

**题目二：提取用户信息更新表和登录信息表的时间信息**

(1)训练要点

- 掌握时间字符串和标准时间的转换方法。
- 掌握时间信息提取的方法。
- 掌握时间数据的算术运算。

(2)需求说明

用户信息更新表 Training_Userupdate.csv 和登录信息表 Training_LogInfo.csv 中均存在大量的时间数据，提取时间数据内存在的信息，一方面可以加深对数据的理解，另一方面能够探索这部分信息和目标的关联程度。同时用户登录时间、借款成交时间、用户信息更新时间这些时间的时间差信息也能够反映出 P2P 网络贷款不同用户的行为信息。

(3)实现步骤

- 使用 to_datetime 函数转换用户信息更新表和登录信息表的时间字符串。
- 使用 year，month，week 等方法，提取用户信息更新表和登录信息表中的时间信息。
- 计算用户信息更新表和登录信息表中两时间的差，分别以日、小时、分钟计算。

**题目三：使用分组聚合方法进一步分析用户信息更新表和登录信息表**

(1)训练要点

- 掌握分组聚合的原理与步骤。
- 掌握 agg／apply 聚合方法。
- 掌握 transform 聚合方法。

(2)需求说明

分析用户信息更新表和登录信息表时，除了提取时间本身的信息外，还可以结合用户编号进行分组聚合，然后进行组内分析。通过组内分析可以得出每组组内的最早和最晚信息更新时间、最早和最晚登录时间、信息更新的次数、登录的次数等信息。

(3)实现步骤

- 使用 groupby 方法对用户信息更新表和登录信息表进行分组。
- 使用 agg 方法求取分组后的最早，最晚，更新登录时间。
- 使用 size 方法求取分组后的数据的信息更新次数与登录次数。

**题目四：使对用户信息更新表和登录信息表进行长宽表转换**

(1)训练要点

- 掌握透视表的制作方法。

(2)需求说明

通过对数据的描述性统计，以及时间数据信息提取，分组聚合操作已经获得了相当多的信息，但用户信息更新表 Training_Userupdate.csv 和登录信息表 Training_LogInfo.csv 是长表，而主表是宽表，需要通过长宽表转换将数据合并在一张表内。

(3)实现步骤

- 使用 povit_table 函数进行长宽表转换：

  登录信息表 Training_LogInfo.csv 以'Idx'为索引，列为'LogInfo1'，聚合函数为 count 生成透视表。

  用户信息更新表 Training_Userupdate.csv 以'Idx'为索引，列为'UserupdateInfo1'，聚合函数为 count 生成透视表。

## 4.实验背景知识

pandas 是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas 提供了大量能使我们快速便捷地处理数据的函数和方法。

（1）查看网络贷款数据的大小、维度、占用内存信息

使用函数 ndim 可以查看数据的维度，shape 查询数据的形状大小，memory_usage 查看数据的占用内存信息。如代码 1 所示。

代码 1 网络贷款数据的大小、维度、占用内存信息

```
In[1]:  import pandas as pd
        master = pd.read_csv('./第 4 章/data/Training_Master.csv',encoding='gbk')
        print('P2P 网络贷款主表数据的维度为：',master.ndim)
Out[1]: P2P 网络贷款主表数据的维度为： 2
In[2]:  print('P2P 网络贷款主表数据的形状大小为：',master.shape)
Out[2]: P2P 网络贷款主表数据的形状大小为： (30000, 228)
In[3]:  print('P2P 网络贷款主表数据的占用内存为：',master.memory_usage())
Out[3]:
```

（2）描述统计

describe 函数能够一次性得出数据框所有数值型特征的非空值数目、均值、四分位数、标准差。具体实现代码和结果如代码 2 所示。

**代码 2 使用 describe 方法进行描述性统计**

In[4]:
```
print('P2P 网络贷款主表数据的描述性统计为：\n',master.describe())
```

Out[4]:
```
P2P 网络贷款主表数据的描述性统计为：
                Idx     UserInfo_1     UserInfo_3  WeblogInfo_1  WeblogInfo_2
count  30000.000000  29994.000000  29993.000000    970.000000  28342.000000
mean   46318.673267      3.219911      4.694329      2.201031      0.131466
std    26640.397805      1.827684      1.321458      7.831679      0.358486
min        3.000000      0.000000      0.000000      1.000000      0.000000
25%    22924.250000      1.000000      4.000000      1.000000      0.000000
50%    46849.500000      3.000000      5.000000      1.000000      0.000000
75%    69447.250000      5.000000      5.000000      1.000000      0.000000
max    91703.000000      7.000000      7.000000    133.000000      4.000000
```

（3）剔除值相同或全为空的列

通过 describe 基础的统计可以发现部分数据整列为空，或者取值相同。结合 DataFrame 的查改增删知识，可以将这部分数据去除，如代码 3 所示。

**代码 3 剔除值相同或全为空的列**

In[5]:
```
# 定义一个函数去除全为空值的列和标准差为 0 的列
def dropNullStd(data):
    beforelen = data.shape[1]
    colisNull = data.describe().loc['count'] == 0
    for i in range(len(colisNull)):
        if colisNull[i]:
            data.drop(colisNull.index[i],axis = 1,inplace =True)

    stdisZero = data.describe().loc['std'] == 0
    for i in range(len(stdisZero)):
        if stdisZero[i]:
            data.drop(stdisZero.index[i],axis = 1,inplace =True)
    afterlen = data.shape[1]
    print('去除的列的数目为：',beforelen-afterlen)
    print('去除后数据的形状为：',data.shape)
dropNullStd(master)
```

Out[5]:
```
去除的列的数目为： 2
去除后数据的形状为： (30000, 226)
```

（4）时间字符串转换为标准时间格式

pandas 提供的 to_datetime 函数，能够将时间相关的字符串都会转换成为 Timestamp。如代码 4 所示。

代码 **错误!文档中没有指定样式的文字。**更新用户信息更新表和登录信息表的时间字符串

In[1]:

```
import pandas as pd
LogInfo = pd.read_csv('./第 4 章/data/Training_LogInfo.csv',encoding='gbk')
Userupdate = pd.read_csv('./第 4 章/data/Training_Userupdate.csv',encoding='gbk')
# 转换时间字符串
LogInfo['Listinginfo1']=pd.to_datetime(LogInfo['Listinginfo1'])
LogInfo['LogInfo3']=pd.to_datetime(LogInfo['LogInfo3'])
print('转换登录信息表的时间字符串前 5 行：\n',LogInfo.head())
```

Out[1]:

```
转换登录信息表的时间字符串前 5 行：
   Idx Listinginfo1  LogInfo1  LogInfo2  LogInfo3
0  10001  2014-03-05       107         6 2014-02-20
1  10001  2014-03-05       107         6 2014-02-23
2  10001  2014-03-05       107         6 2014-02-24
3  10001  2014-03-05       107         6 2014-02-25
4  10001  2014-03-05       107         6 2014-02-27
```

In[2]:

```
Userupdate['ListingInfo1']=pd.to_datetime(Userupdate['ListingInfo1'])
Userupdate['UserupdateInfo2']=pd.to_datetime(Userupdate['UserupdateInfo2'])
print('转换用户信息更新表的时间字符串前 5 行：\n',Userupdate.head())
```

Out[3]:

```
转换用户信息更新表的时间字符串前 5 行：
   Idx ListingInfo1  UserupdateInfo1 UserupdateInfo2
0  10001  2014-03-05      _EducationId      2014-02-20
1  10001  2014-03-05        _HasBuyCar      2014-02-20
2  10001  2014-03-05    _LastUpdateDate      2014-02-20
3  10001  2014-03-05  _MarriageStatusId      2014-02-20
4  10001  2014-03-05      _MobilePhone      2014-02-20
```

（5）提取时间序列数据信息

结合 Python 列表推导式，可以实现对 DataFrame 某一列时间信息数据的提取。用户信息更新表和登录信息表时间的年份，月份，日期，周信息提取，如代码 所示。

代码 5 提取用户信息更新表和登录信息表中的时间信息

In[3]:

```
# 定义一个提取用户信息的函数
def extract(file,time):
    global year
    year = [i.year for i in file[time]]
```

```
        month = [i.month for i in file[time]]
        week = [i.week for i in file[time]]
        return year,month,week
ETLog1 = extract(LogInfo,'Listinginfo1')
print('每行的前五个数据：\n',ETLog1[0][0:5],ETLog1[1][0:5],ETLog1[2][0:5])
```

Out[3]:
```
每行的前五个数据：
 [2014, 2014, 2014, 2014, 2014]
 [3, 3, 3, 3, 3]
 [10, 10, 10, 10, 10]
```

In[4]:
```
ETLog2 = extract(LogInfo,'LogInfo3')
print('每行的前五个数据：\n',ETLog2[0][0:5],ETLog2[1][0:5],ETLog2[2][0:5])
```

Out[4]:
```
每行的前五个数据：
 [2014, 2014, 2014, 2014, 2014]
 [2, 2, 2, 2, 2]
 [8, 8, 9, 9, 9]
```

In[5]:
```
ETUser1 = extract(Userupdate,'ListingInfo1')
print('每行的前五个数据：\n',ETUser1[0][0:5],ETUser1[1][0:5],ETUser1[2][0:5])
```

Out[5]:
```
每行的前五个数据：
 [2014, 2014, 2014, 2014, 2014]
 [3, 3, 3, 3, 3]
 [10, 10, 10, 10, 10]
```

In[6]:
```
ETUser2 = extract(Userupdate,'UserupdateInfo2')
print('每行的前五个数据：\n',ETUser2[0][0:5],ETUser2[1][0:5],ETUser2[2][0:5])
```

Out[6]:
```
每行的前五个数据：
 [2014, 2014, 2014, 2014, 2014]
 [2, 2, 2, 2, 2]
 [8, 8, 8, 8, 8]
```

（6）加减时间数据

Timedelta 能够直接对两个时间序列进行相减，从而得出一个 Timedelta。如代码 所示。

**代码 6 计算用户信息更新表和登录信息表中两时间的差**

In[7]:
```
# 分别计算更新表和登录信息表中两时间的差
TDLog = LogInfo['Listinginfo1'] - LogInfo['LogInfo3']
print('计算登录信息表中两时间之差：\n',TDLog.head())
```

Out[7]:
```
计算更新表中两时间之差：
 0    13 days
1    10 days
2     9 days
```

```
3      8 days
4      6 days
dtype: timedelta64[ns]
```

In[8]:

```
TDUser = Userupdate['ListingInfo1'] - Userupdate['UserupdateInfo2']
print('计算更新表中两时间之差：\n',TDUser.head())
```

Out[8]:

```
计算更新表中两时间之差：
 0    13 days
1    13 days
2    13 days
3    13 days
4    13 days
dtype: timedelta64[ns]
```

(7) 使用 groupby 方法对数据分组

使用 groupby 函数对用户信息更新表和登录信息表进行分组，如代码 所示。

**代码 7 对用户信息更新表和登录信息表进行分组**

In[1]:

```
import pandas as pd
import numpy as np
LogInfo = pd.read_csv('./第 4 章/data/Training_LogInfo.csv',encoding='gbk')
Userupdate = pd.read_csv('./第 4 章/data/Training_Userupdate.csv',encoding='gbk')
# 使用 groupby 方法对用户信息更新表和登录信息表进行分组
LogGroup = LogInfo[['Idx','LogInfo3']].groupby(by = 'Idx')
UserGroup = Userupdate[['Idx','UserupdateInfo2']].groupby(by = 'Idx')
```

(8) 使用 agg 方法更新登录时间

使用 agg 方法求取分组后的最早，最晚，更新登录时间，如代码 所示。

**代码 8 使用 agg 方法求取分组后的最早，最晚，更新登录时间**

In[2]:

```
# 使用 agg 方法求取分组后的最早，最晚，更新登录时间
print('分组后的最早登录时间为：\n',LogGroup.agg(np.min))
```

Out[2]:

```
分组后的最早登录时间为：
Idx        LogInfo3
3          2013-08-30
5          2013-10-24
8          2013-10-25
12         2012-12-08
```

In[3]:

```
print('分组后的最晚登录时间为：\n',LogGroup.agg(np.max))
```

Out[3]:

```
分组后的最晚登录时间为：
Idx        LogInfo3
```

```
3       2013-11-01
5       2013-11-06
8       2013-11-06
12      2013-11-01
```

In[4]:

```
print('分组后的最早更新时间为：\n',UserGroup.agg(np.min))
```

Out[4]:

```
分组后的最早更新时间为：
Idx         UserupdateInfo2
3           2013/08/30
5           2013/10/24
8           2013/10/25
12          2012/12/08
```

In[5]:

```
print('分组后的最晚更新时间为：\n',UserGroup.agg(np.max))
```

Out[5]:

```
分组后的最晚更新时间为：
Idx         UserupdateInfo2
3           2013/08/30
5           2013/10/24
8           2013/11/04
12          2013/10/02
```

(9) 使用 size 方法求数据的信息更新次数与登录次数

使用 size 方法求取分组后的数据的信息更新次数与登录次数，如代码所示。

**代码 9 求取分组后的数据的信息更新次数与登录次数**

In[6]:

```
# 使用 size 方法求取分组后的数据的信息更新次数与登录次数
print('分组后的数据的信息更新次数为：\n',LogGroup.size())
```

Out[6]:

```
分组后的数据的信息更新次数为：
Idx
3          26
5          11
8         125
12        199
```

In[7]:

```
print('分组后的数据的登录次数为：\n',UserGroup.size())
```

Out[7]:

```
分组后的数据的登录次数为：
 Idx
3          13
5          13
8          14
12         14
```

(10) 使用 povit_table 函数进行长宽表转换

使用 povit_table 函数进行长宽表转换，如代码 所示。

代码 10 使用 povit_table 函数进行长宽表转换

In[1]:

```
import pandas as pd
LogInfo = pd.read_csv('./第 4 章/data/Training_LogInfo.csv',encoding='gbk')
Userupdate = pd.read_csv('./第 4 章/data/Training_Userupdate.csv',encoding='gbk')
# 用 pivot_table 函数将长表转换成宽表
LogInfo_pivot = pd.pivot_table(LogInfo,index='Idx',columns = ['LogInfo1'],aggfunc='count')
print('用 LogInfo1 作为分组键创建的登录信息表：\n',LogInfo_pivot.head())
```

Out[1]:

用 LogInfo1 作为分组键创建的登录信息表：

| | Listinginfo1 | | | | ... | LogInfo3 | | | |
|---|---|---|---|---|---|---|---|---|---|
| LogInfo1 | -10 | -4 | 0 | 1 | ... | 1000 | 2000 | 3000 | 3001 |
| Idx | | | | | ... | | | | |
| 3 | NaN | 15.0 | NaN | 4.0 | ... | NaN | NaN | NaN | NaN |
| 5 | 1.0 | 4.0 | NaN | 3.0 | ... | NaN | NaN | NaN | NaN |
| 8 | 1.0 | 109.0 | NaN | 4.0 | ... | NaN | NaN | NaN | NaN |
| 12 | 1.0 | 35.0 | NaN | 4.0 | ... | NaN | NaN | NaN | NaN |
| 16 | NaN | 10.0 | NaN | 3.0 | ... | NaN | NaN | NaN | NaN |

In[2]:

```
Userupdate_pivot=pd.pivot_table(Userupdate,index='Idx',columns=['UserupdateInfo1'],aggfunc='count')
print('用 UserupdateInfo1 作为分组键创建的用户信息更新表：\n',Userupdate_pivot.head())
```

Out[2]:

用 UserupdateInfo1 作为分组键创建的登录信息表：

| | ListingInfo1 | | | ... | UserupdateInfo2 | |
|---|---|---|---|---|---|---|
| UserupdateInfo1 | _Age | _BussinessAddress | ... | | _webShopUrl | _workYears |
| Idx | | | ... | | | |
| 3 | NaN | NaN | ... | | NaN | NaN |
| 5 | NaN | NaN | ... | | NaN | NaN |
| 8 | NaN | NaN | ... | | NaN | NaN |
| 12 | 1.0 | NaN | ... | | NaN | NaN |
| 16 | NaN | NaN | ... | | NaN | NaN |

## 5.实验思考

（1） DataFrame 和数组有什么相似之处？

（2） 时间数据中存在哪些信息？

（3） 为什么索引的时候有 loc 和 iloc，设计者的用意何在？