PYTHON数据科学系列课程（五）
PANDAS高级分析与案例

东南大学 学习科学研究中心 儿童发展与教育研究所
夏小俊
http://www.seuct.com

本章内容提要

- 5.1 RFM分析
- 5.2 矩阵分析
- 5.3 相关分析
- 5.4 回归分析

5.1 RFM分析

RFM分析是一个经典的商业数据模型，R代表Recency（近度），F代表Frequency（频度），M代表Monetary（金额）。

根据三个维度的高低不同，可以将用户分为8个不同的类别

R	F	M	类型
高	高	高	高价值客户
低	高	高	重点保持客户
高	低	高	重点发展客户
低	低	高	重点挽留客户
高	高	低	一般价值客户
低	高	低	一般保持客户
高	低	低	一般发展客户
低	低	低	潜在客户

5.1 RFM分析

In[1]:

```
import numpy as np
import pandas as pd
from datetime import datetime
data = pd.read_csv('CSV/RFM.csv')
data['DealDateTime'] = pd.to_datetime(data.DealDateTime)
data['DateDiff'] = datetime.now() - data.DealDateTime
data['DateDiff'] = data.DateDiff.dt.days  # 计算上次购物的时间距离
```

5.1 RFM分析

In[1]:

```
R_Agg = data.groupby('CustomerID',
as_index=False)['DateDiff'].agg('min')
F_Agg = data.groupby('CustomerID',
as_index=False)['OrderID'].agg('count')
M_Agg = data.groupby('CustomerID',
as_index=False)['Sales'].agg('sum')
aggData = R_Agg.merge(F_Agg).merge(M_Agg)
#按客户分组，分别统计每名客户最近购物的天数、购物总次数和购物总金额，
然后再合并数据框
```

5.1 RFM分析

In[1]:

```
aggData.columns = ['CustomerID', 'RecencyAgg', 'FrequencyAgg', 'MonetaryAgg']
bins = aggData.RecencyAgg.quantile(q=[0, 0.2, 0.4, 0.6, 0.8, 1], interpolation= 'nearest')
bins[0] = 0 #确保最小值在区间内
R_S = pd.cut(aggData.RecencyAgg, bins, labels=[5, 4, 3, 2, 1])
bins = aggData.FrequencyAgg.quantile(q=[0, 0.2, 0.4, 0.6, 0.8, 1], interpolation= 'nearest')
bins[0] = 0
F_S = pd.cut(aggData.FrequencyAgg, bins, labels=[1, 2, 3, 4, 5])
bins = aggData.MonetaryAgg.quantile(q=[0, 0.2, 0.4, 0.6, 0.8, 1], interpolation= 'nearest')
bins[0] = 0
M_S = pd.cut(aggData.MonetaryAgg, bins, labels=[1, 2, 3, 4, 5])
#分别对三个维度的数据进行切割和打分，注意R维度是越小越好。
```

5.1 RFM分析

In[1]:

```
aggData['R_S'] = R_S
aggData['F_S'] = F_S
aggData['M_S'] = M_S
aggData['RFM'] = 100 * R_S.astype(int) + 10 * \
   F_S.astype(int) + 1*M_S.astype(int)
bins = aggData.RFM.quantile(
   q=[0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1], interpolation='nearest')
bins[0] = 0
aggData['level'] = pd.cut(aggData.RFM, bins, labels=[1, 2, 3, 4, 5, 6, 7, 8])
aggData.groupby('level')['CustomerID'].agg('count')
#将R、F和M的权重分别设为100、10和1，计算总分后再分为八组，最后统计各组人数。
```

5.2 矩阵分析

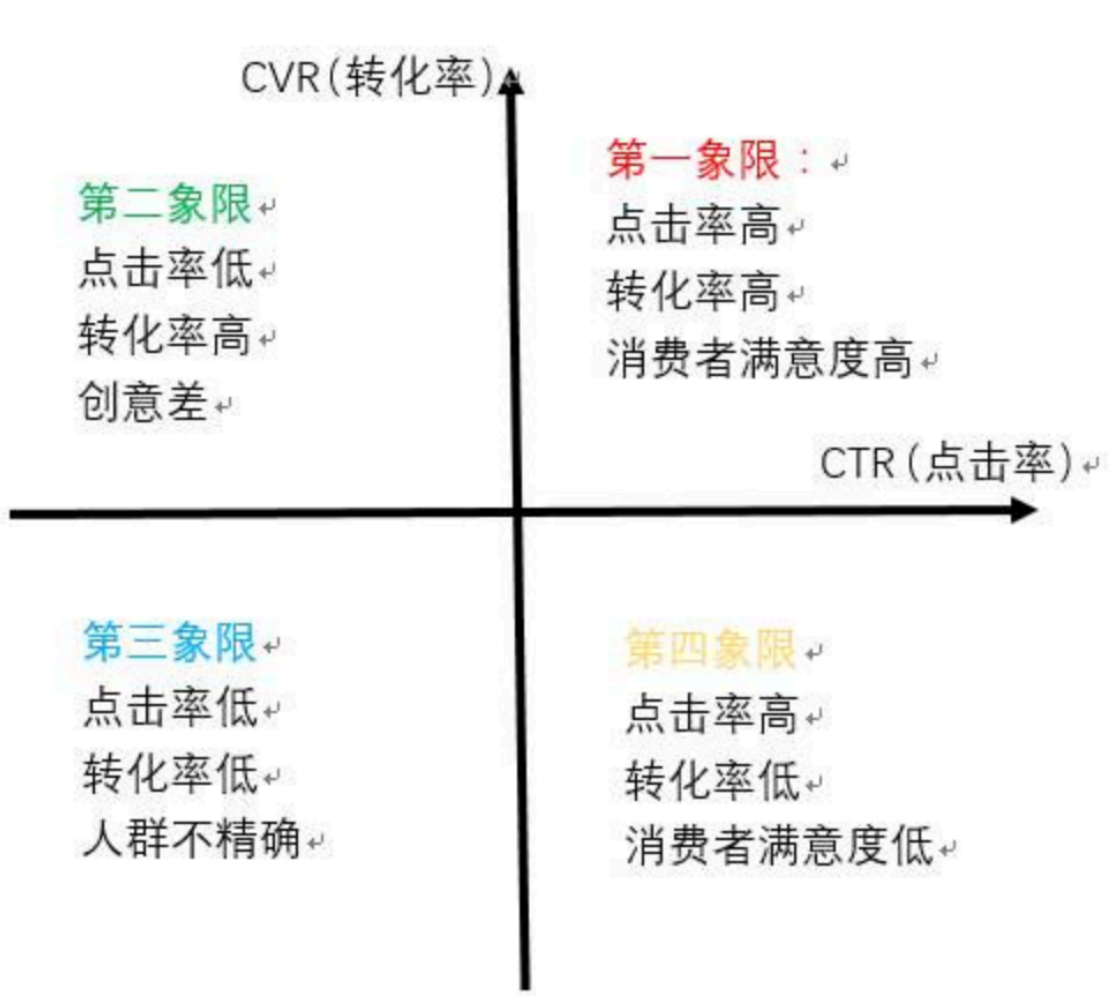
矩阵分析是指根据事物的两个重要属性（或指标）作为分析的依据进行关联分析，找出解决问题的一种分析方法，也称为矩阵关联分析法。

通常选取两个属性组成坐标系，根据不同的水平进行象限划分，将要分析的对象投射到象限内，进行交叉分类分析，将属性的关联性表现出来，从而分析每个对象在这两个属性的表现，也称为象限图分析法。

矩阵分析法在生活当中有着非常广泛的应用，可以为决策者提供重要的参考依据。

5.2 矩阵分析

如图所示，从点击率和转化率两个维度分析网站广告的质量。



5.2 矩阵分析

In[2]:

```
import numpy as np
import pandas as pd
data = pd.read_csv('csv/矩阵分析.csv')
#从消费和流量两个角度 分析各省的用户质量
costAgg = data.groupby(by= '省份' , as_index = False)['月消费（元）
'].agg('mean')
dataAgg = data.groupby(by= '省份' , as_index = False)['月流量
（MB）'].agg('mean')
aggData = costAgg.merge(dataAgg)
#后续通过散点图绘制完成任务
```

5.3 相关分析

世界是普遍联系的，某种现象的变化必然受与之相联系的其他现象发展变化的发展和制约。

相关关系：现象之间存在非严格的、不确定的依存关系。当给定某一现象的某一值时，另一现象会有若干数值与之对应，但遵循一定的规律。

回归函数关系：现象之间存在依存关系，可以用一定的数学表达式表示出来。

相关分析可分为线性相关和非线性相关，通常讨论线性相关。

通常通过pearson相关系数r来研究线性相关的强度，范围在[-1,1]。

5.3 相关分析

In[3]:

```
import numpy as np
import pandas as pd
data = pd.read_csv('csv/相关分析.csv')
data['人口'].corr(data['文盲率'])
corrMatrix = data[[
    '超市购物率','网上购物率','文盲率','人口'
]].corr()  # method='pearson','kendall','spearman'
corrMatrix
```

5.4 回归分析

“回归”一词来源于Galton对人类身高的研究，即高矮人群的后代从不同方向回归到人类的平均身高。

回归分析（Regression Analysis），是研究自变量和因变量之间数量变化关系的一种方法，通过建立因变量和自变量之间的回归模型，来预测因变量发展的趋势。

相关分析和回归分析的联系：先进行相关分析，计算相关系数，再进行拟合回归模型，最后进行预测。

相关分析和回归分析的区别：相关分析不分变量类型，且都是随机变量；回归分析一定要区分自变量和因变量，而且自变量是确定的普通变量。

5.4 回归分析

回归分析模型分为线性回归和非线形回归两种，前者又分为简单线性回归和多重线性回归；非线形回归也通常通过数学转化的方法，转化为线形方式。

回归分析的五个步骤：

1）根据预测目标，确定自变量和因变量

2）绘制散点图，确定回归模型类型

3）估计模型参数，建立回归模型

4）对回归模型进行检验

5）利用回归模型进行预测

5.4.1 简单回归分析

In[4]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from sklearn.linear_model import LinearRegression

mpl.rcParams['font.sans-serif'] = ['KaiTi'] #plot函数的汉字字体
mpl.rcParams['font.serif'] = ['KaiTi']

data = pd.read_csv('csv/简单线性回归.csv')
```

5.4.1 简单回归分析

In[4]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from sklearn.linear_model import LinearRegression

mpl.rcParams['font.sans-serif'] = ['KaiTi'] #plot函数的汉字字体
mpl.rcParams['font.serif'] = ['KaiTi']

data = pd.read_csv('csv/简单线性回归.csv')
```

5.4.1 简单回归分析

In[4]:

```
x = data[['广告费用(万元)']]
y = data[['销售额(万元)']]

data['广告费用(万元)'].corr(data['销售额(万元)'])

data.plot('广告费用(万元)', '销售额(万元)', kind='scatter')

#求出广告和销售额的相关系数，并简单绘制散点图
```

5.4.1 简单回归分析

In[4]:

```
lrModel = LinearRegression()
lrModel.fit(x, y) # 训练模型
print(lrModel.coef_) # 参数
print(lrModel.intercept_) # 截距
lrModel.score(x, y) # 拟合优度，决定系数
pX = pd.DataFrame(
    {'广告费用(万元)': [20]}
)
lrModel.predict(pX)
```

5.4.2 多重回归分析

In[5]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
from sklearn.linear_model import LinearRegression

mpl.rcParams['font.sans-serif'] = ['KaiTi']
mpl.rcParams['font.serif'] = ['KaiTi']

data = pd.read_csv('csv/多重线性回归.csv') #多个自变量
```

5.4.2 多重回归分析

In[5]:

```
x = data[['广告费用(万元)', '客流量(万人次)']]
y = data[['销售额(万元)']]

data['广告费用(万元)'].corr(data['销售额(万元)'])
data['客流量(万人次)'].corr(data['销售额(万元)'])


data.plot('广告费用(万元)', '销售额(万元)', kind='scatter')
data.plot('客流量(万人次)', '销售额(万元)', kind='scatter')
```

5.4.2 多重回归分析

In[5]:

```
lrModel = LinearRegression()
lrModel.fit(x, y)
print(lrModel.coef_) # 参数
print(lrModel.intercept_) # 截距
lrModel.score(x, y) # 拟合优度，决定系数
pX = pd.DataFrame(
    {'广告费用(万元)': [20],
     '客流量(万人次)': [5]
     }
)
lrModel.predict(pX)
```

5.5 PANDAS分析案例

In[6]:

```
import numpy as np
import pandas as pd

df = pd.read_csv('csv/job.csv', encoding='gb2312')
df.info()  #读取数据框的信息

print(len(df.positionId.unique())) # 不重复的职位ID
df_new = df.drop_duplicates(subset='positionId', keep='first')
df_new.info()
```

5.5 PANDAS分析案例

In[6]:

```
def cut_word(word, method):   #处理工资数据
    position = word.find('-')
    length = len(word)
    if position != -1:
        bottom_salary = word[:position - 1]
        top_salary = word[position + 1: length - 1]
    else:
        top_salary = bottom_salary = word[:word.upper().find('K')]
    if method == 'bottom':
        return bottom_salary
    else:
        return top_salary
```

5.5 PANDAS分析案例

In[6]:

```
#清洗工资数据，新增最低、最高和平均工资
df_new['bottomSalary'] = df_new.salary.apply(cut_word,
method='bottom')
df_new.bottomSalary = df_new.bottomSalary.astype('int')
df_new['topSalary'] = df_new.salary.apply(cut_word,
method='top')
df_new.topSalary = df_new.topSalary.astype('int')
df_new['avgSalary'] = (df_new.bottomSalary + df_new.topSalary)
/ 2
```

5.5 PANDAS分析案例

In[7]:

```
df_clean = df_new[['city', 'companyShortName', 'companySize', 'education',
                   'positionName', 'positionLables', 'workYear', 'avgSalary']]
df_clean.city.value_counts() #非零元素的个数
df_clean.describe()
#绘图暂略
df_clean.groupby('city').count()
df_clean.groupby('city').mean()
df_clean.groupby(['city','education']).mean()
df_clean.groupby(['city','education']).mean().unstack() #行列转置
df_clean.groupby(['city','education']).avgSalary.count().unstack() #行列转置
#按公司统计
df_clean.groupby('companyShortName').avgSalary.agg(['count',
'mean']).sort_values(by='count',ascending=False)
df_clean.groupby('companyShortName').avgSalary.agg(lambda x:max(x) -
min(x))
```

5.5 PANDAS分析案例

In[7]:

```
#不同城市，招聘数据分析师、职位需求前5
def topN(df, n=5):
    counts = df.value_counts()
    return counts.sort_values(ascending=False)[:n]

df_clean.groupby('city').companyShortName.apply(topN)
df_clean.groupby('city').positionName.apply(topN)
```

5.5 PANDAS分析案例

In[8]:

```
# 薪资水平
bins = [0, 3, 5, 10, 15, 20, 30, 100]
level = ['0-3', '3-5', '5-10', '10-15', '15-20', '20-30', '30-100']
df_clean['level'] = pd.cut(df_clean['avgSalary'], bins=bins, labels=level)
df_clean[['avgSalary', 'level']]
```

5.5 PANDAS分析案例

In[9]:

```
#处理统计职位描述
word = df_clean.positionLables.str[1:-1].str.replace(' ', '')

df_word = word.dropna().str.split(',').apply(pd.value_counts)
df_word.unstack().dropna().reset_index()
df_word.unstack().dropna().reset_index().groupby('level_0').count()
```

谢谢大家！