Pandas 实战 Kaggle titanic 数据探索性分析

在上一天完成数据清洗任务后，今天我们开始对数据展开探索性分析，简称 EDA 分析。它主要探索各个分类变量，及变量间的关系，以及各个变量与目标值的关系。

Pandas 非常适合做 EDA 分析，它提供了友好的函数，便于我们做数据透视。常用的函数包括：

- melt
- pivot, pivot_table
- crosstab
- groupby

下面先举例论述，以上函数如何使用，然后在 titanic 预测数据集上，实战应用其中某些函数，完成 EDA.

1 melt 透视

melt 函数，将宽 DataFrame 透视为长 DataFrame.

构造一个 DataFrame：

复制
```
d = {\
"district_code": [12345, 56789, 101112, 131415],
"apple": [5.2, 2.4, 4.2, 3.6],
"banana": [3.5, 1.9, 4.0, 2.3],
"orange": [8.0, 7.5, 6.4, 3.9]
}

df = pd.DataFrame(d)
df
```

打印结果：

复制
```
district_code    apple   banana  orange
0    12345   5.2 3.5 8.0
1    56789   2.4 1.9 7.5
2    101112  4.2 4.0 6.4
3    131415  3.6 2.3 3.9
```

5.2 表示 12345 区域的 `apple` 价格，并且 `apple`, `banana`, `orange`，这三列都是一种水果，使用 `melt` 把这三列合并为一列。方法如下：

复制
```
dfm = df.melt(\
id_vars = "district_code",
var_name = "fruit_name",
value_name = "price")
dfm
```

打印结果：

复制
```
district_code    fruit_name  price
0    12345   apple   5.2
1    56789   apple   2.4
2    101112  apple   4.2
3    131415  apple   3.6
4    12345   banana  3.5
5    56789   banana  1.9
6    101112  banana  4.0
7    131415  banana  2.3
8    12345   orange  8.0
9    56789   orange  7.5
10    101112  orange  6.4
11    131415  orange  3.9
```

2 pivot 和 pivot_table

pivot 将长 DataFrame 透视为宽 DataFrame，与 melt 函数透视方向相反。函数的主要参数说明：

1). index 指明哪个列变为新 DataFrame 的 index； 2). columns 指明哪些列变为 columns； 3). values 指明哪些列变为新 DataFrame 的数据域，如果不指明，则默认除了被指明 index 和 columns 的其他列。

对上面使用 melt 透视后的 dfm ，使用 `pivot` 函数，将长 DataFrame 变形为宽 DataFrame：

复制
```
dfp = dfm.pivot(index='district_code',columns='fruit_name',values=['price'
])
dfp
```

结果：

复制
```
price
fruit_name    apple   banana  orange
district_code
12345    5.2 3.5 8.0
56789    2.4 1.9 7.5
101112    4.2 4.0 6.4
131415    3.6 2.3 3.9
```

打印透视后的 dfp 的列：

复制
```
dfp.columns
```

结果如下，为多索引列：

复制
```
MultiIndex([('price',  'apple'),
            ('price', 'banana'),
            ('price', 'orange')],
           names=[None, 'fruit_name'])
```

因此，透视后访问某列，就得使用多索引列：

```
dfp[('price',  'apple')]
```

结果：

```
district_code
12345     5.2
56789     2.4
101112    4.2
131415    3.6
Name: (price, apple), dtype: float64
```

pivot 函数是没有聚合功能的。

但是，pandas 中提供的 `pivot_table()` 提供聚合功能。因此，`pivot_table` 可看是 `pivot` 函数的升级版。

pivot_table

为了演示，`pivot_table` 的聚合透视功能，先生成一个 DataFrame:

```
d = {\
"district_code": [12345, 12345, 56789, 101112, 131415,12345, 12345, 56789
, 101112, 131415],
"fruit_name": ['apple', 'apple', 'orange', 'banana', 'orange','apple', 'a
pple', 'orange', 'banana', 'orange'],
"price":      [3.5, 3.7, 1.9, 4.0, 2.3,4.5, 4.7, 2.9, 5.0, 3.3]
}

df2 = pd.DataFrame(d)
df2
```

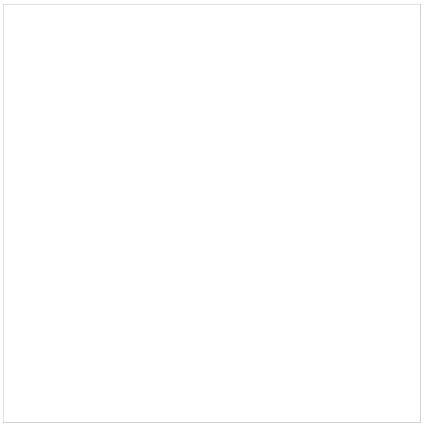
结果：

```
district_code    fruit_name  price
0    12345   apple   3.5
1    12345   apple   3.7
2    56789   orange  1.9
3    101112  banana  4.0
4    131415  orange  2.3
5    12345   apple   4.5
6    12345   apple   4.7
7    56789   orange  2.9
8    101112  banana  5.0
9    131415  orange  3.3
```

`pivot_table` 函数，不仅具有 `pivot` 函数将长 DataFrame 透视为宽 DataFrame, 同时还具有 sum 聚合功能：

```
dfp2 = df2.pivot_table(index='district_code',columns='fruit_name',values=[
'price'],aggfunc=[np.sum])
dfp2
```

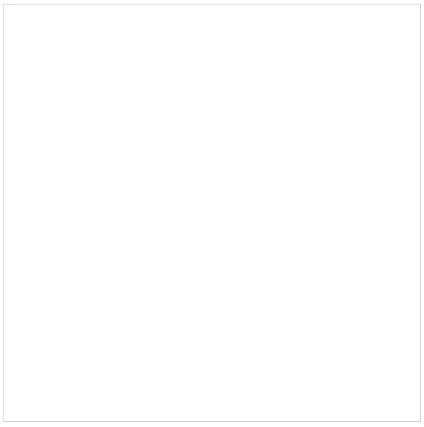
结果如下，district_code 为 `12345` 的区域，apple 价格求和已经聚合为 16.4



一次可以使用多个聚合方法，如下，使用求和、求平均值聚合：

```
dfp3 = df2.pivot_table(index='district_code',columns='fruit_name',values=[
'price'],aggfunc=[np.sum,np.mean])
dfp3
```

结果：



Pandas 还提供一个透视功能更加专一的函数，按照频次透视函数： crosstab

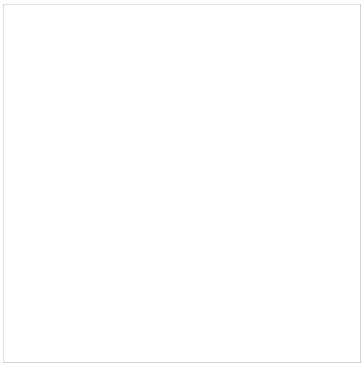
3 crosstab 透视频次

有 2 个必传递的参数 `index`, `colmns` ，分别为透视后的行、列索引。

`crosstab` 与 `pivot_table` 很相似，看一个例子。

复制
```
a = np.array(['apple','apple', 'orange', 'banana', 'orange'], dtype=objec
t)
b = np.array(['china','china', 'ameri', 'ameri', 'korea'], dtype=object)
c = np.array([ 'good','good','good','good','better'],dtype=object)
pd.crosstab(a,[b,c])
```

结果为：



以上，实质等价于：

复制
```
for it in zip(a,b,c):
    print(it)
```

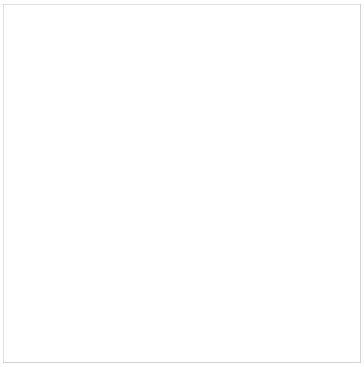
可以看到 apple, china, good 这项出现的频次为 2，其他频次都是 1.

复制
```
('apple', 'china', 'good')
('apple', 'china', 'good')
('orange', 'ameri', 'good')
('banana', 'ameri', 'good')
('orange', 'korea', 'better')
```

同理，

复制
```
pd.crosstab([a,b],[c])
```

结果为：



还是只有一项(apple, china, good)频次为2，和上面的原理一样。

例子

复制
```
df = pd.DataFrame({'类别':['水果','水果','水果','蔬菜','蔬菜','肉类','肉类'],
                   '产地':['美国','中国','中国','中国','新西兰','新西兰','美国'],
                   '水果':['苹果','梨','草莓','番茄','黄瓜','羊肉','牛肉'],
                  '数量':[5,5,9,3,2,10,8],
                  '价格':[5,5,10,3,3,13,20]})
df #显示df
```

结果：

复制
```
类别    产地  水果  数量  价格
0   水果  美国  苹果  5   5
1   水果  中国  梨   5   5
2   水果  中国  草莓  9   10
3   蔬菜  中国  番茄  3   3
4   蔬菜  新西兰 黄瓜  2   3
5   肉类  新西兰 羊肉  10  13
6   肉类  美国  牛肉  8   20
```

`类别` 列设置为 index, `产地` 列设置为 columns，统计词条出现频次：

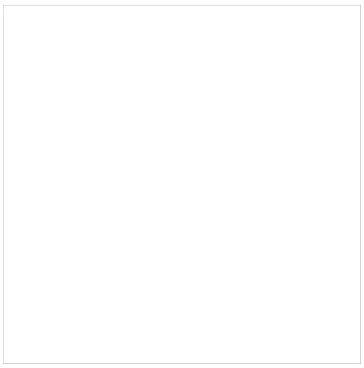
复制
```
pd.crosstab(df['类别'],df['产地'],margins=True)
```

结果：

如果想使用聚合函数，`aggfun` 参数，必须指明 `values` 列，如下：

```
pd.crosstab(df['类别'],df['产地'],values=df['价格'],aggfunc=np.max, margins=True)
```

结果如下：



crosstab本质：按照指定的 index 和 columns 统计 DataFrame 中出现(index, columns)的频次。

值得注意，这些透视函数，melt, pivot, pivot_table, crosstab，都基于 `groupby` 分组基础上，而分组大家是更容易理解的，所以在理解这些透视函数时，可以结合分组思想。

4 4 种表连接方法

给定两个 DataFrame，它们至少存在一个名称相同的列，如何连接两个表？使用merge 函数连接两个 DataFrame，连接方式共有 4 种，分别为：left, right, inner,outer. 如何区分这 4 种连接关系

2 个 DataFrame 分别为 left、 right，名称相同的列为 key，left 的 key 取值为：k0, k1, k2 ; right 表的 key 取值为：k0, k0, k1

1) 如果 left 的 key **指向** right ，此连接方式为: **left** : 关系图表达为：

```
left    right
k0      k0
        k0
k1      k1
k2      NaN
```

2) 如果 right 的 key **指向** left，则称此连接方式为: **right**

```
right  left
k0      k0
k0
k1      k1
```

3) 如果只拿 left 和 right 公有 key（也就是交集）建立关系 ，称此连接方式为: **inner**

```
left  right
k0     k0
k0     k0
k1     k1
```

4) 如果 left 和 right 的 key合并（也就是并集）再建立关系后，称此连接方式为: **outer**

```
left  right
k0     k0
       k0
k1     k1
k2     NaN
```

以上就是 merge 连接 2 个DataFrame 时，根据 key 节点建立关系的 4 种方法。

下面举例说明：

left 和 right 分别为：

left

```
   age1   key
0   10   k0
1   20   k1
2   30   k2
```

right

```
   age2   key
0   40   k0
1   50   k0
2   60   k1
```

如果连接方法参数 `how` 取值为 'left'

```
pd.merge(left,right,how='left',on='key')
```

结果：

```
   age1   key   age2
0   10   k0   40.0
1   10   k0   50.0
2   20   k1   60.0
3   30   k2   NaN
```

如果连接方法参数 `how` 取值为 'right'

```
pd.merge(left,right,how='right',on='key')
```

结果：

```
age1   key   age2
0   10   k0   40
1   10   k0   50
2   20   k1   60
```

如果连接方法参数 `how` 取值为 'inner'

```
pd.merge(left,right,how='inner',on='key')
```

结果：

```
age1   key   age2
0   10   k0   40
1   10   k0   50
2   20   k1   60
```

如果连接方法参数 `how` 取值为 'outer'

```
pd.merge(left,right,how='outer',on='key')
```

结果：

```
age1   key   age2
0   10   k0   40.0
1   10   k0   50.0
2   20   k1   60.0
3   30   k2   NaN
```

5 EDA 实战

读入经过清洗后的泰坦尼克预测数据集，共包括 891 行 、7列。

首先导入使用的包和数据集：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data_eda = pd.read_csv('../input/titanic_eda_data.csv')
data_eda
```

导入后的数据展示如下所示：

img

验证每一列取值是否都为数值型：

```
data_eda.dtypes
```

结果：

```
Survived    int64
Pclass      int64
Sex         int64
AgeBin      int64
IsAlone     int64
Title       int64
Embarked    int64
dtype: object
```

泰坦尼克数据集目标预测船员的生死，因此，**所有探索分析都要聚焦在 `Survived` 这一列上，分析出给定的 6 个特征与 1 个目标值 `Survived` 的关系。**

单变量分析

单变量分析，是指分析单个特征与 `Survived` 的关系。很明显，`Survived` 的取值为 0、 1，船员要么 die, 要么 live.

下面分别透视每个特征与 `Survived` 的关系。

```
eda_cols = ['Survived','Pclass','Sex','AgeBin','IsAlone','Title','Embarked']

for x in eda_cols[1:]:
    print(x)
    pivot_x = data_eda.pivot_table(index=data_eda.index,
    columns=[x],values=['Survived']).agg(['count','mean'])
    print(pivot_x)
    print('*'*20)
```

结果：

```
 Pclass
        Survived
```

```
                1           2           3
count  216.00000  184.000000  491.000000
mean     0.62963    0.472826    0.242363
********************
Sex
         Survived
                0           1
count  314.000000  577.000000
mean     0.742038    0.188908
********************
AgeBin
      Survived
             0           1           2          3          4
count   100.00  525.000000  186.000000  69.000000  11.000000
mean      0.55    0.344762    0.403226   0.434783   0.090909
********************
IsAlone
        Survived
               0           1
count  354.00000  537.000000
mean     0.50565    0.303538
********************
Title
      Survived
             0          1           2           3        4
count   40.000  27.000000  182.000000  517.000000  125.000
mean     0.575   0.444444    0.697802    0.156673    0.792
********************
Embarked
         Survived
                0         1           2
count  168.000000  77.00000  646.000000
mean     0.553571   0.38961    0.339009
********************
```

对 `Pclass` 特征，取值有 1,2,3，其中 216人 `Pclass` 为 1， 获救比率大约 0.629.

再看 `Sex` 特征，为 0 表示女性，获救比率 74.2%，也就是说，如果一刀切，只看男女特征，预测船员是否获救，我们就可以说，如果是女性有 74.2% 的可能被获救，而男性只有 18.%多的概率被获救。

下面绘制每个单变量对 `Survived` 的影响的柱状图：

复制

```
plt.figure(figsize=[16,12])

for i,x in enumerate(eda_cols[1:]):
    gx = data_eda[[x, 'Survived']].groupby(x, as_index=False).mean()
    plt.subplot(int('23'+str(i+1)))
    sns.barplot(x=gx[x], y=gx['Survived'])
    plt.title(x+' Survived')
    plt.xlabel(x)
    plt.ylabel('Survived')
```

柱状图如下所示：

img

从图中看出，`Embarked` 特征共有 3 种可能取值，并且获救的平均值，波动不大，也就是方差较小。

因此，`Embarked` 特征对于预测船员是否获救的价值就没有 `Sex` 特征意义大。下面，分别计算出获救平均值的方差。

复制

```
for x in eda_cols[1:]:
    print(x)
    pivot_x = data_eda.pivot_table(index=data_eda.index,columns=[x],value
s=['Survived']).agg(['mean'])
    print(pivot_x.iloc[0,:].std())
```

结果：

复制

```
Pclass
0.19479759330588634
Sex
0.391122024068098
AgeBin
0.1703605609503242
IsAlone
0.14291444223489114
Title
0.24772394245713808
Embarked
0.11216104937942797
```

绘制柱状图：

复制

```
stds = []
for x in eda_cols[1:]:
    print(x)
    pivot_x = data_eda.pivot_table(index=data_eda.index,columns=[x],value
s=['Survived']).agg(['mean'])
    stds.append((x,pivot_x.iloc[0,:].std()))
sns.barplot(x=[e[0] for e in stds], y=[e[1] for e in stds])
plt.title('std of mean survived')
```

img

价值最大的三个特征：Sex, Title, Pclass，分别是性别，头衔，社会等级，某种角度讲，这也非常符合我们的尝试，更有可能逃命的往往是身居要职、出人头地、有头有脸的人。

以上就是单变量的分析过程。

如果船员既是女性，并且 Title 又等于 4，我们预期她会有大于 62.9 % 的概率被获救，这就是双变量分析或多变量分析。

多变量分析

透视完单个变量与船员是否获救后，透视双变量预测船员是否获救的分析，能帮助我们获取更大信息量。

首先组合特征对：

```
feature_pair = [ (e1,e2) for i,e1 in enumerate(eda_cols[1:]) for e2 in ed
a_cols[i+1:] if e1!=e2]

feature_pair ### 组合特征对
```

结果：

```
[('Pclass', 'Sex'),
 ('Pclass', 'AgeBin'),
 ('Pclass', 'IsAlone'),
 ('Pclass', 'Title'),
 ('Pclass', 'Embarked'),
 ('Sex', 'AgeBin'),
 ('Sex', 'IsAlone'),
 ('Sex', 'Title'),
 ('Sex', 'Embarked'),
 ('AgeBin', 'IsAlone'),
 ('AgeBin', 'Title'),
 ('AgeBin', 'Embarked'),
 ('IsAlone', 'Title'),
 ('IsAlone', 'Embarked'),
 ('Title', 'Embarked')]
```

绘制特征组合对与 `Survived` 的关系柱状图：

```
plt.figure(figsize=[16,12])

i=0
for e1,e2 in feature_pair[:9]:
   pair = data_eda[[e1,e2,'Survived']].groupby([e1,e2],as_index=False).me
an()
   plt.subplot(int('33'+str(i+1)))
   sns.barplot(x='('+ pair[e1].astype('str') +','+ pair[e2].astype('str')
 + ')', y=pair['Survived'],palette="rocket")
   plt.title('(%s,%s) mean of Survived '%(e1,e2))
   plt.ylabel('Survived')
   plt.xticks([])
   i += 1
```

打印结果，特征对与 `Survived` 值：

```
   Pclass  Sex  Survived
0       1    0  0.968085
1       1    1  0.368852
2       2    0  0.921053
3       2    1  0.157407
4       3    0  0.500000
5       3    1  0.135447
    Pclass  AgeBin  Survived
0        1       0  0.888889
1        1       1  0.640449
2        1       2  0.681159
3        1       3  0.534884
4        1       4  0.166667
5        2       0  0.904762
6        2       1  0.422680
7        2       2  0.446809
8        2       3  0.352941
9        2       4  0.000000
10       3       0  0.400000
11       3       1  0.244838
12       3       2  0.100000
13       3       3  0.111111
14       3       4  0.000000
   Pclass  IsAlone  Survived
0       1        0  0.728972
1       1        1  0.532110
2       2        0  0.637500
3       2        1  0.346154
4       3        0  0.299401
5       3        1  0.212963
    Pclass  Title  Survived
0        1      0  1.000000
1        1      1  0.611111
2        1      2  0.956522
3        1      3  0.345794
4        1      4  0.976190
5        2      0  1.000000
6        2      1  0.111111
7        2      2  0.941176
8        2      3  0.087912
9        2      4  0.902439
10       3      0  0.392857
11       3      2  0.500000
12       3      3  0.112853
13       3      4  0.500000
   Pclass  Embarked  Survived
0       1         0  0.694118
1       1         1  0.500000
2       1         2  0.589147
3       2         0  0.529412
4       2         1  0.666667
5       2         2  0.463415
6       3         0  0.378788
7       3         1  0.375000
8       3         2  0.189802
   Sex  AgeBin  Survived
0    0       0  0.673469
1    0       1  0.718391
2    0       2  0.791045
3    0       3  0.916667
4    1       0  0.431373
5    1       1  0.159544
6    1       2  0.184874
7    1       3  0.177778
8    1       4  0.090909
   Sex  IsAlone  Survived
0    0        0  0.712766
1    0        1  0.785714
2    1        0  0.271084
3    1        1  0.155718
   Sex  Title  Survived
0    0      1  1.000000
1    0      2  0.697802
2    0      4  0.792000
3    1      0  0.575000
4    1      1  0.250000
5    1      3  0.156673
   Sex  Embarked  Survived
0    0         0  0.876712
1    0         1  0.750000
2    0         2  0.692683
```

```
3    1         0  0.305263
4    1         1  0.073171
5    1         2  0.174603
```

当 Sex 为 0 ， Title 为 1 时， `Survived` 列的平均值为 1.，即全部获救。我们打印下这部分人群数：

复制

```
sex_title_count = data_eda[['Sex','Title','Survived']].groupby(
['Sex','Title'],as_index=False).agg(['count'])
sex_title_count
```

img

发现，一共只有 7 人，样本数太少，只有一定的参考价值。

打印 Pclass, Title 的样本个数，在 Pclass 等于 1 ， Title 等于 2 时，有42人，获救比率高达 95.% 多，因此，接下来的预测人群中，只要满足这两个特征值，我们有信心预测他们获救。

复制

```
sex_title_count = data_eda[['Pclass','Title','Survived']].groupby(
['Pclass','Title'],as_index=False).agg(['count'])
sex_title_count
```

img

换一种绘制图，seaborn 的 catplot 图，上面双变量与 `Survived` 关系图：

复制

```
plt.figure(figsize=[16,12])

for e1,e2 in feature_pair:
    pair = data_eda[[e1,e2,'Survived']].groupby([e1,e2],as_index=False).m
ean()
    sns.catplot(x=e1, y="Survived", hue=e2, data=pair,
                kind="bar", palette="muted")
    plt.ylabel('Survived')
```

结果图：

img

img

img

img

img

img

img

img

img

img

img

img

img

img

最后，绘制特征间的 pairplot 图：

复制

```
g = sns.PairGrid(data_eda, hue="Survived")
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter)
g.add_legend();
```

结论：

`Pclass`, `Sex`, `Title`, `AgeBin` 是影响 `Survived` 的最重要 4 个特征。

如在 `Pclass` 等于 1， `Title` 等于 2 或 4 时，预测船员会获救。

6 小结

今天，与大家一起学习数据探索分析的常用技术和思考方式，包括：

- melt
- pivot, pivot_table
- crosstab
- 4 种 DataFrame 连接方法
- 数据分析实战：单变量，多变量分析的思考方法

下一章

互动评论

说点什么

存

评论

The Scrapper 1 个月前

习题也不错

鼓掌