数据分析实战——共享单车

共享单车在近两年来火爆的不要不要的，而我在学习了数据分析师（入门）的几堂课之后，十分想要尝试一下数据分析的过程。

此次打算从 kaggle 上的共享单车项目进行入手，选择了 bike-sharing-demand 项目，COME ON！

说到数据分析就要想到一个东西，数据分析的流程。数据分析主要分为 5 大流程：

1 需求分析

明确自己的需求是什么，想要用这个数据集来做什么？

这个应该是很多小伙伴们都很纠结的一个问题，不知道从何入手，那我们可以去找一下资料，看看别人做这个分析的时候都会从哪些角度入手？怎么样研究问题呢？

2 数据获取

我们有了明确的目标之后，就要考虑数据是怎么获取的，本文最初采用 kaggle 数据集中的数据。

获取数据的方法除了从 kaggle 下载之外，还有很多方法，之前我们的公众号 DC

黑板报就有一篇专门介绍数据获取方式的文章（学会数据获取方式，搞定数据分析第一步）

。

3 数据处理

数据处理是整个数据分析过程中最麻烦的步骤，有句话说“数据科学家的 70%时间都是用在数据处理上”。

4 数据分析

当我们拿到了已经进行清洗完的数据之后，那我们就要考虑具体的分析内容了。

分析方法有很多，常见的有描述性统计分析、探索性数据分析、验证性数据分析。可以根据自己的数据和分析目标去选择。

5 数据可视化

数据可视化是数据分析的最后一步，也叫做结果展示，通过图表的方式有效并且清晰的来展示与传达信息。

在本文中我们采用的是 python 中的 seaborn 库进行可视化。我们可以认为它是 matplotlib 库的高级版，对复杂的可视化图表支持的比较好并且也很美观，可以媲美 R 语言的 ggplot2 库。好了，简单介绍完了数据分析的流程之后，我们就要正式开工了。

step1 导入包

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

这里面的包就不过多解释啦，都是我们常用的数据分析及可视化包。

你们可能会比较好奇%matplotlibinline 是什么东西。这个其实是因为我用的是 jupyternotebook，如果不用这一行代码的话，就只会显示出一串稀奇古怪的东西。

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a61d59bd68>
```

除了这行代码可以之外呢，还有另外一个也可以，就是%matplotlib note，它的功能更为强大，还能放大缩小图表，但是缺点就是更耗内存。

step2 导入数据

```
#导入数据集
train_data = pd.read_csv("F:\\共享单车\\train.csv",engine = 'python',encoding = 'utf-8')
test_data = pd.read_csv("F:\\共享单车\\test.csv",engine = 'python')
```

在此处选择的是试用 pandas 库中 read.csv()方法，后面看到很多的参数，原因是我之前没有加后面的参数之后，一直在报错无法初始化。

考虑两个情况，第一个就是因为有中文的问题，所以在第一个里面放入了 encoding 参数，找到是 engine 的过程很曲折，此处不细说。

step3 查看数据字段含义

```
train_data.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
test_data.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

用 head 方法查看前五个数据，可以看出 train 数据集里面的字段比 test 的字段多三个，然后查看数据量。

```
print('训练集：',train_data.shape,'测试集:',test_data.shape)
```

```
训练集： (10886, 12) 测试集: (6493, 9)
```

训练集的数据量是 10886 条 12 个字段，测试集是 6493 条数据 9 个字段，训练集比测试集的字段量多的为：casual、registered、count。

这些会不会给我们的分析带来灵感呢？下面我们就来看下这个数据集里面给的条件。

- Datatime:日期时间，可以拆分出来年月日、时分秒这些数值
- Season：季节 1=春天 2=夏天 3=秋天 4=冬天
- Holiday:是否是节假日 0=是，1=不是
- Workingday:1=工作日 0=周末
- Weather:1=晴天，多云 2=雾天，阴天 3=小雪，小雨 4=大雨，大雪，大雾
- Temp:气温摄氏度
- Atemp:体感温度
- humidity：湿度
- windspeed：风速
- casual：非注册用户个数
- registered :注册用户个数
- count:总租车人数（在特定时间内）

step4 数据清洗

字段含义已经很清晰明了了，下面就开始进行数据清洗工作。

01

我们采用的是查看是否有空值，那就需要将两个数据集合并。

```
all_data = pd.concat([train_data,test_data],ignore_index=True)
all_data.isnull().sum()
```

得到的结果：

```
atemp            0
casual        6493
count         6493
datetime         0
holiday          0
humidity         0
registered    6493
season           0
temp             0
weather          0
windspeed        0
workingday       0
dtype: int64
```

我们可以看到除了 test 里面的三个不含有的字段之外，其它都是 0，这说明这个数据非常的完整，良心数据。

02

为了后续分析方便，datatime 字段我们会处理成单个的日期时间，分割为日期和时间，因为给出的都是小时数，所以我们直接切分为小时。

```
all_date["date"] = all_date.datetime.apply(lambda x : x.split()[0])
all_date["time"] = all_date.datetime.apply(lambda x : x.split()[1].split(":")[0])
```

```
all_date["date"].head()
```

```
0    2011-01-01
1    2011-01-01
2    2011-01-01
3    2011-01-01
4    2011-01-01
Name: date, dtype: object
```

```
all_date["time"].head()
```

```
0    00
1    01
2    02
3    03
4    04
Name: time, dtype: object
```

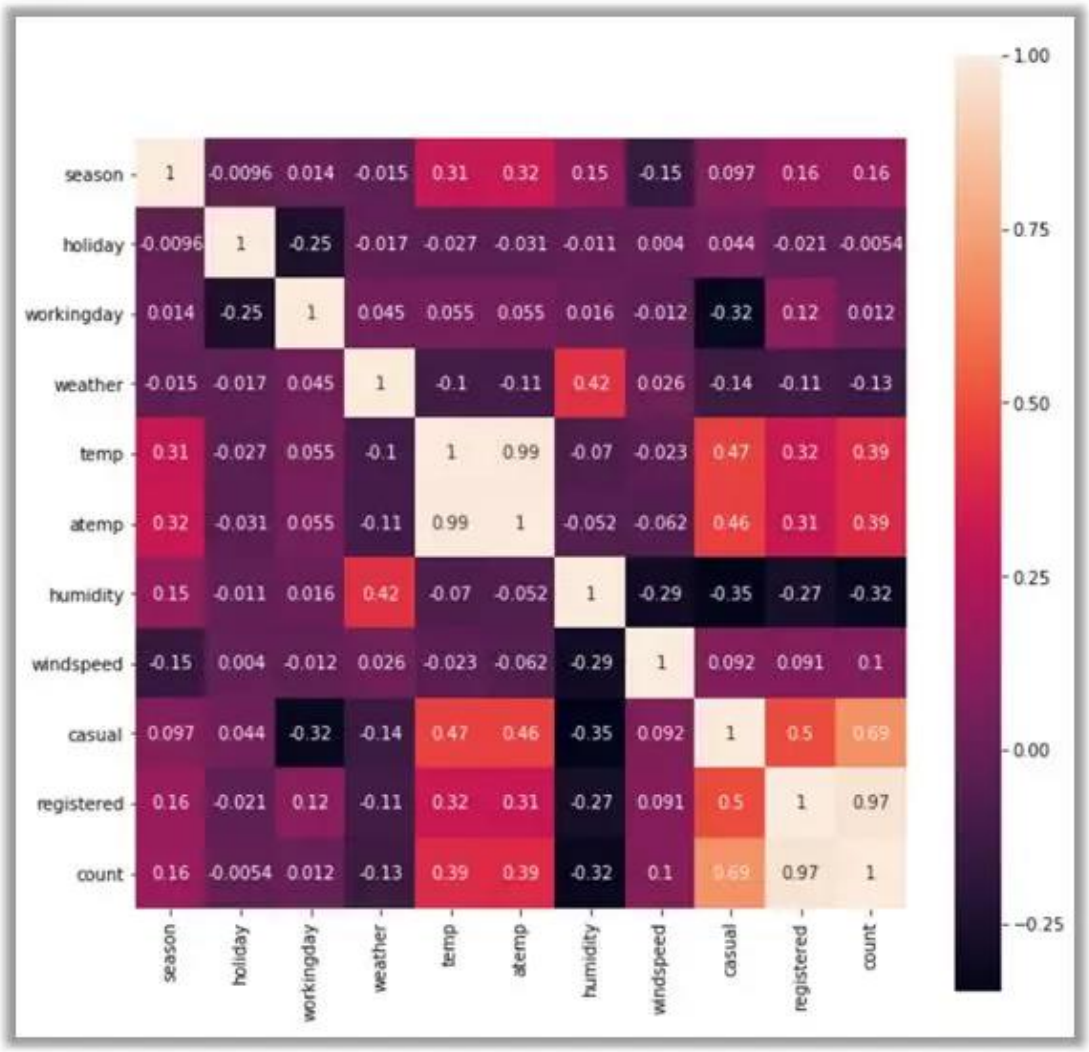
为了方便后面的处理，我们将原始数据转化为 DataFrame 格式。

```
all_dateDf = pd.DataFrame(all_date)
```

step5 数据分析及可视化

完成上面的处理之后我们就要进行数据的分析了，首先我们选择使用相关性分析。

01 热图



根据上图我们可以看出

- ✧ 季节和体感温度、外界温度有强的正相关
- ✧ 天气和湿度呈现较强的正相关
- ✧ 气温温度和体感温度及租车人数呈现强正相关性
- ✧ 体感温度和气温温度及租车人数呈现强正相关性
- ✧ 湿度和租车人数呈现较强的负相关性

总结上面的两点就是，温度越高租车人数越多；湿度越大租车人数越少。

湿度和环境有很大关系，猜测湿度非常大可能是雨雪天气，那此时人们对于单车的需求将会大大降低。这个说法并不严谨，肯定有个温度的上限，超过即人数下降。

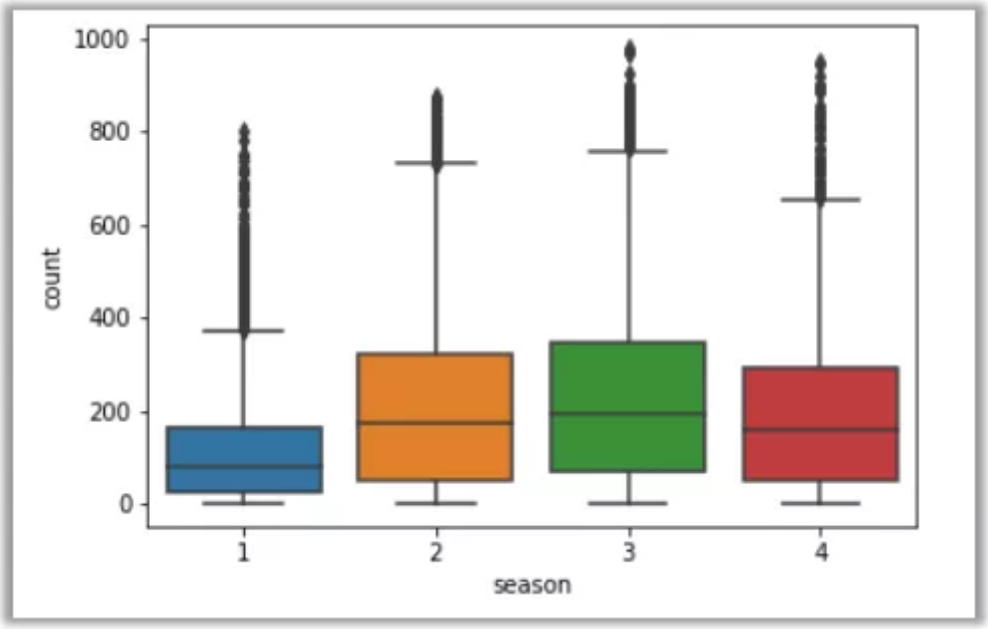
虽然气温和体感温度呈现强相关性，天气和湿度，季节和外界温度体感温度也是强相关性，但是只是本身的性质，对于目前分析来说没有什么帮助。

02 季节和租车人数的关系

既然季节和体感温度、气温温度有强相关，而后两者又和租车人数有关，那我们来研究一下季节和租车人数有什么样的关系。

```
ax = fig.add_subplot(4,2,5)
sns.boxplot(data = all_dateDf,x = "season",y = "count")
ax.set(xlabel = "season",ylabel = "count",)
```

获得下图：



我们可以看到相对于夏季和秋季来说，春季租单车人数明显较少，而冬季相对来说变化不是非常大，略小于夏秋季节。

原因可能是春季还没有回暖，部分地区可能会风雨较多，并且所谓春寒料峭嘛，人们大多数不太喜欢骑单车。

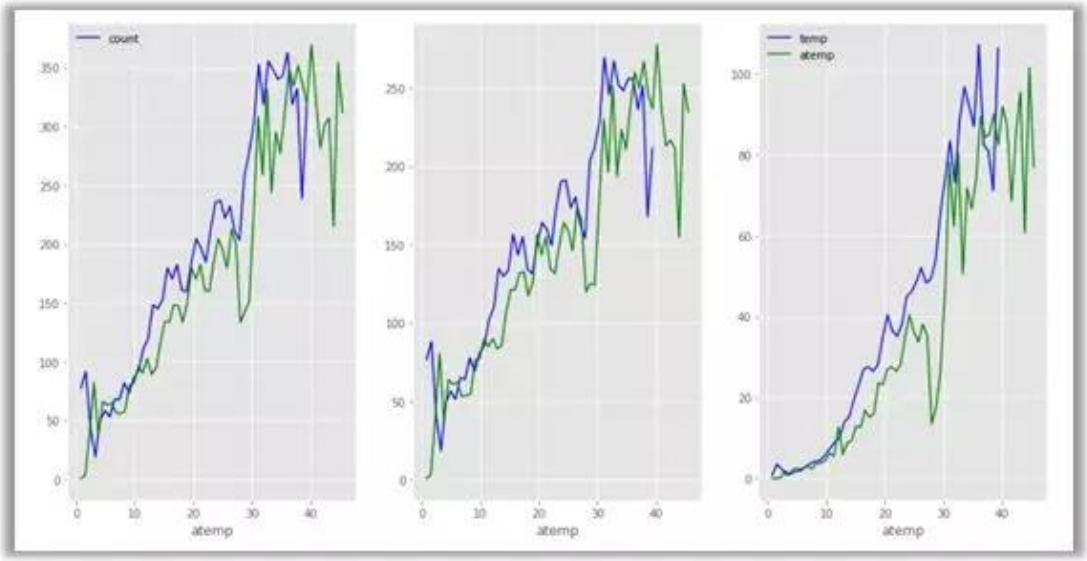
那冬季为什么会比春天多呢？推测可能是因为人们从秋天过渡到冬天的，最开始的时候还是会比较喜欢骑车子，后面越来越冷才会减少骑车。

03 温度和租车人数的关系

到底是不是因为温度的原因呢？那我们下面就研究一下温度。

刚才我们说到了温度和租车人数呈现强相关性，我们能推测出来，温度越高租车的人越多，但是不准确，我们验证一下，Let's GO。

```
fig, axes = plt.subplots(nrows=1,ncols=3)
fig.set_size_inches(16, 8)
all_date.groupby('temp').mean()["count"].plot(linestyle="-",color="b",ax=axes[0],legend=["temperature"])
all_date.groupby('atemp').mean()["count"].plot(linestyle="-",color='g',ax=axes[0])
plt.legend(["temp","atemp"])
all_date.groupby('temp').mean()["registered"].plot(linestyle="-",color="b",ax=axes[1])
all_date.groupby('atemp').mean()["registered"].plot(linestyle="-",color='g',ax=axes[1])
plt.legend(["temp","atemp"])
all_date.groupby('temp').mean()["casual"].plot(linestyle="-",color="b",ax=axes[2])
all_date.groupby('atemp').mean()["casual"].plot(linestyle="-",color='g',ax=axes[2])
plt.legend(["temp","atemp"])
plt.show()
```



根据图像可以看出来，在当温度及体感温度高于大约 35 摄氏度的时候，用户的使用频率就开始降低。

也就是说，虽然确实是随着温度的增加，骑车人的数目也会增加，但是达到一定的温度还是会下降的，说明之前的推测是对的，确实有个上限。

好了，这次简单的小实战练习就结束了，这次的代码参考与了 KAGGLE、知乎上面的大佬的代码，让我在琢磨的过程中受益匪浅~感谢大佬们的分享与帮助。