**Kaggle （Bike Sharing Demand）20%**

**题目：<https://www.kaggle.com/c/bike-sharing-demand>**

**Github地址：<https://github.com/cqychen/mykaggle/tree/master/Bike%20Sharing%20Demand>**

**强调，特征决定结果的高度，模型决定如何逼近这个高度**

# 数据探探

这是一个关于自行车租赁预测的题目，相当于国内的ofo，摩拜单车啦。

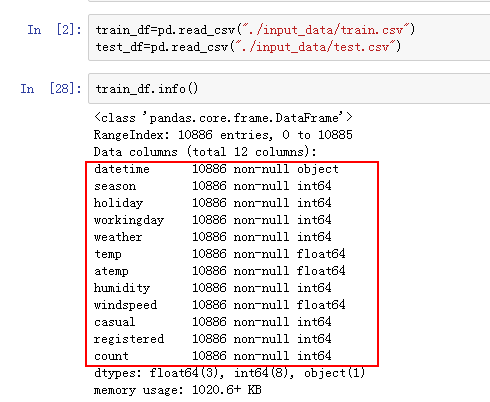
You are provided hourly rental data spanning two years. For this competition, the training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. You must predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.

训练集提供了一个月的前19天的数据和使用情况，测试集提供后面20号以后的数据，我们主要的任务就是预测20号以后的使用量。

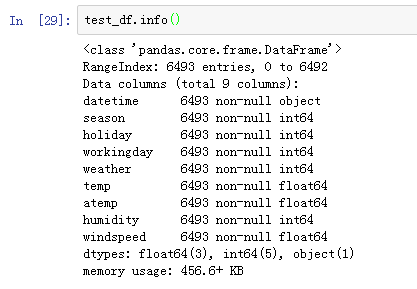
|  |  |  |
| --- | --- | --- |
| 列名 | desc | 中文描述 |
| datetime | hourly date + timestamp | 小时日期 和时间戳 |
| season | 1 = spring, 2 = summer, 3 = fall, 4 = winter | 1：春天 2：夏天 3：秋天 4：冬天 |
| holiday | whether the day is considered a holiday | 当天是否是节假日 |
| workingday | whether the day is neither a weekend nor holiday | 当天是否是工作日 |
| weather | 1: Clear, Few clouds, Partly cloudy, Partly cloudy  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog | 1：晴，少云，部分多云，部分多云。 2：薄雾+多云，薄雾+破碎的云，薄雾+少量的云，雾 3：小雪，小雨+雷雨+散云，小雨+散云 4：大雨+冰盘+雷雨+雾，雪+雾 |
| temp | temperature in Celsius | 温度 |
| atemp | "feels like" temperature in Celsius | 感受到的温度 |
| humidity | relative humidity | 湿度 |
| windspeed | wind speed | 风速 |
| casual | number of non-registered user rentals initiated | 未注册用户的租赁数量 |
| registered | number of registered user rentals initiated | 注册用户的租赁数量 |
| count | number of total rentals | 总的租赁数量 |

## 数据总览

读入数据，看看大致信息：



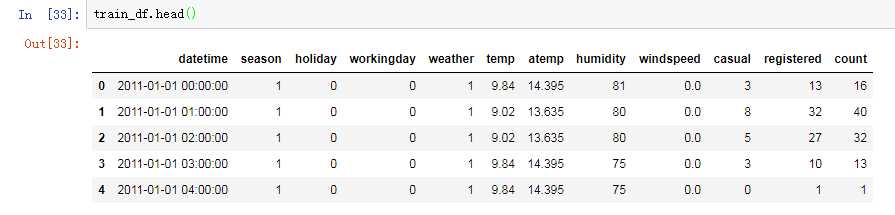
训练集数据共12列，没有数据缺失。哇咔咔



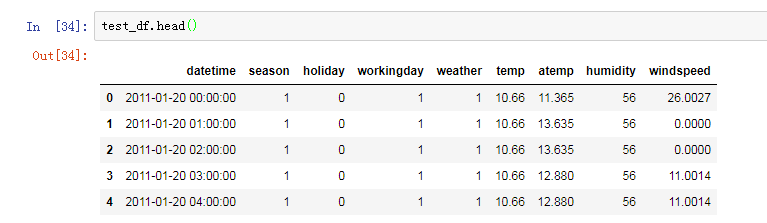
测试集数据共9列，没有数据缺失。

数据明细看看

训练集数据：

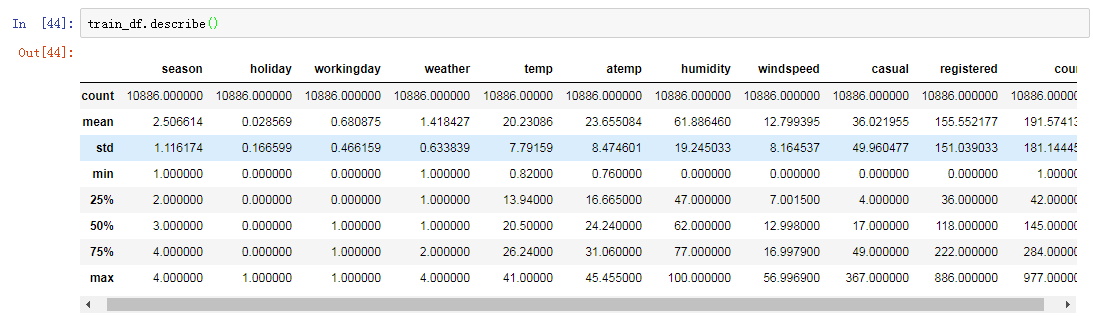


测试集数据：

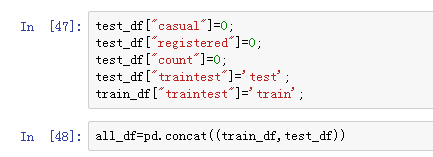


我们可以看到 在测试集中

Casual + register==count



将测试集和训练集合进行拼接，方便做特征工程：



### 日期

我们知道日期的格式是如下：yyyy-MM-dd hh:mm:ss

日期这个东东，基本是要构造出如下的特征：

年

周几

季度

月

小时

一年的第多少周

一年的第多少天

同时可以看到

代码如下：

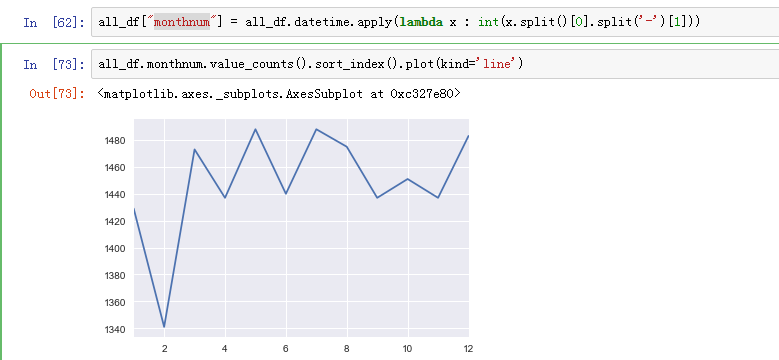
all\_df["date"] = all\_df.datetime.apply(lambda x : x.split()[0])

all\_df["monthnum"] = all\_df.datetime.apply(lambda x : int(x.split()[0].split('-')[1]))

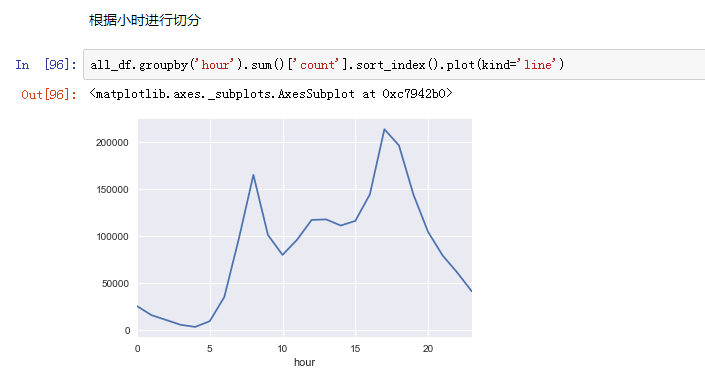
all\_df["daynum"]=all\_df.datetime.apply(lambda x : int(x.split()[0].split('-')[2]))

dailyData["hour"] = dailyData.datetime.apply(lambda x : int(x.split()[1].split(":")[0]))

dailyData["weekday"] = dailyData.date.apply(lambda dateString : calendar.day\_name[datetime.strptime(dateString,"%Y-%m-**%d**").weekday()])



根据小时用量排序查看：



可以看到根据小时用量大致可以分成5个时段：

0~6

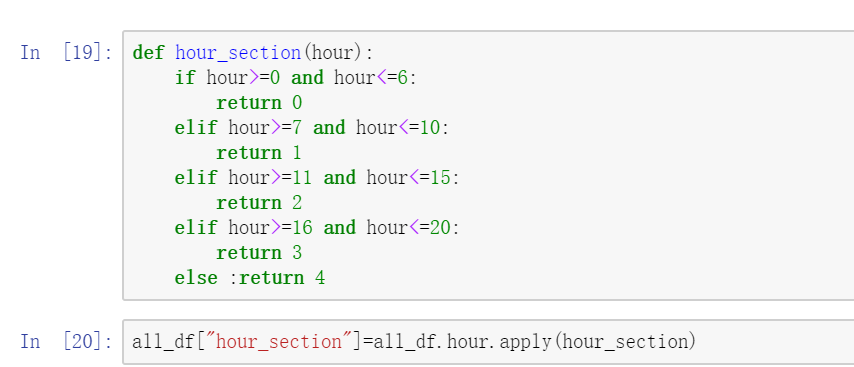
6~10

10~15

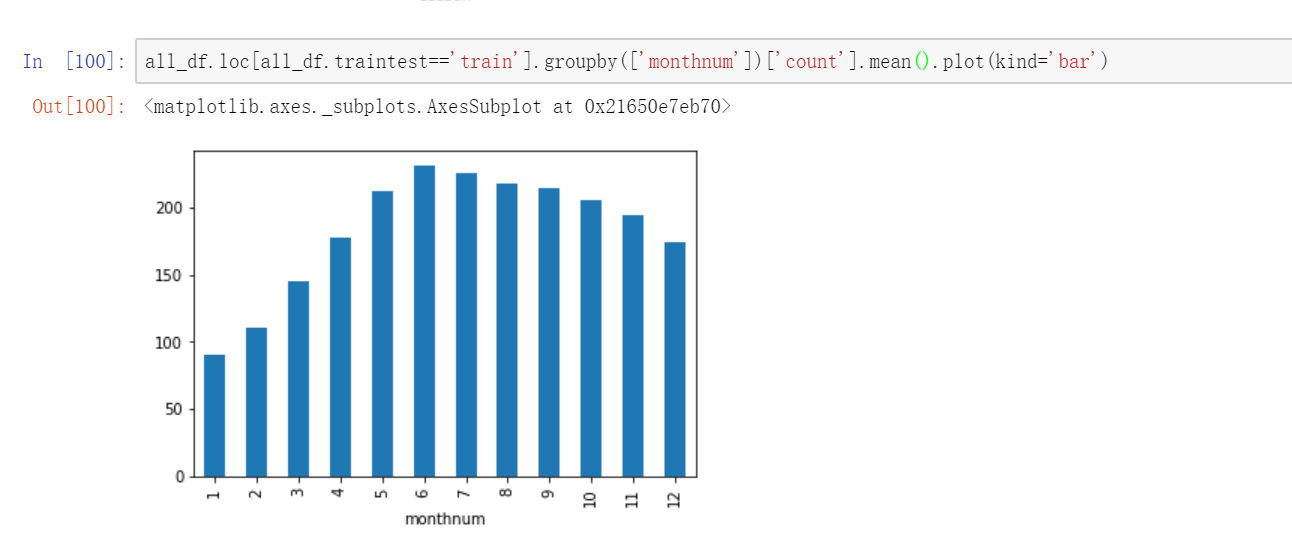
15~20

20~24

代码处理下



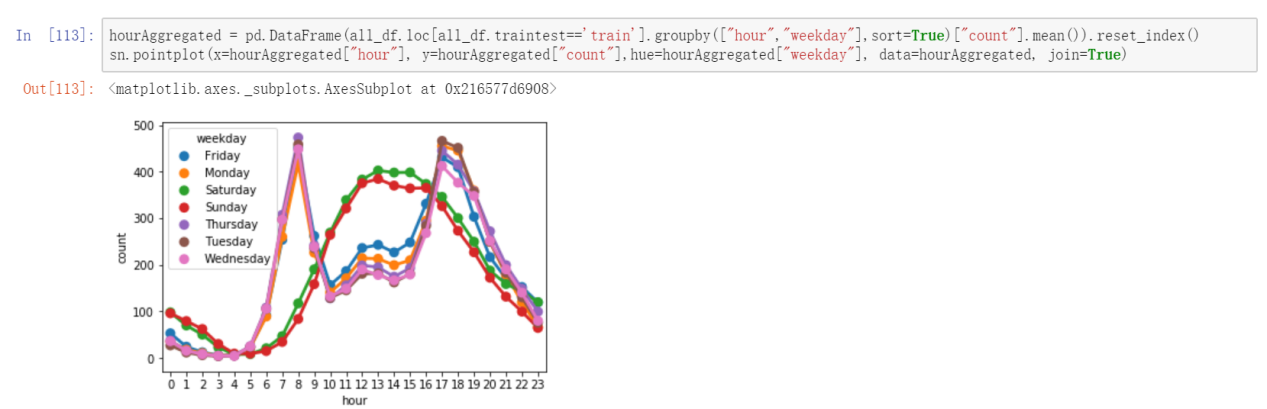
按月份查看使用量；



我们查看周几和小时中骑行情况：

可以看到周六周日 与 周一到周五的情况是完全不同的。也符合我们的逻辑：

1. 上班日一般有早高峰和晚高峰。明显有两个波峰
2. 周六周日有点像正太分布的情况。一天只有一个波峰。所以我们这个时候时间拆分可以更细一步。将周六周日纳入考虑。



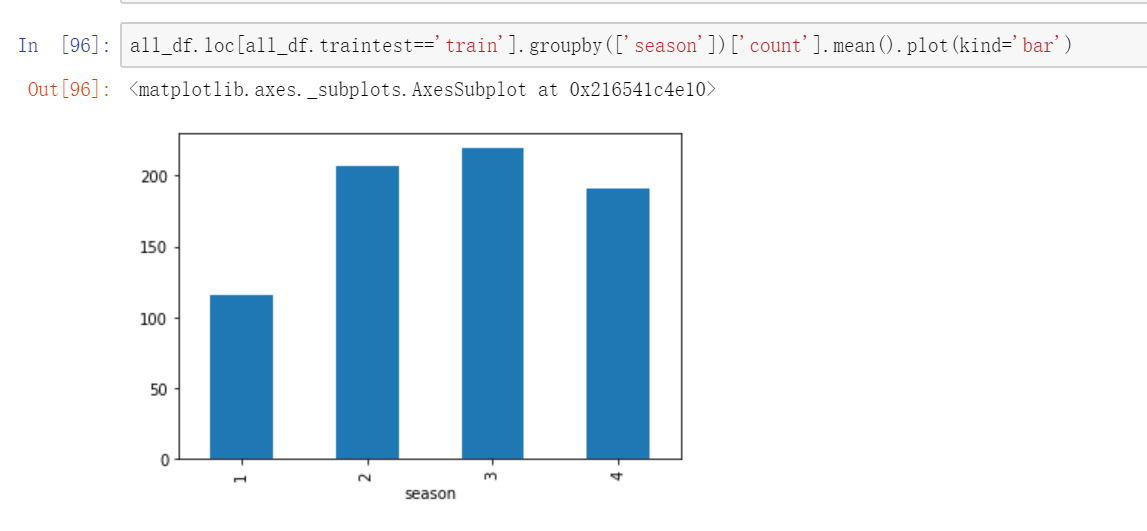
代码如下：



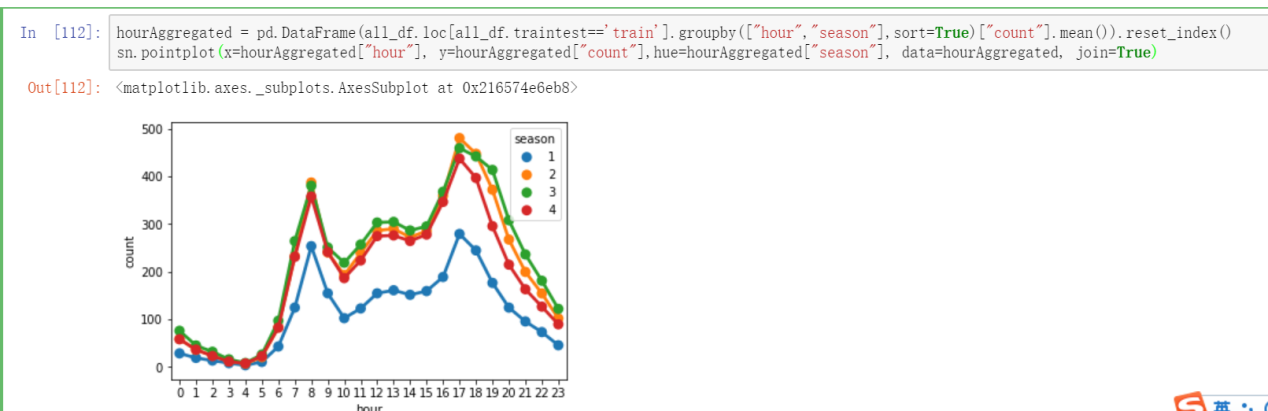
### 季节

按季节查看和count的关系

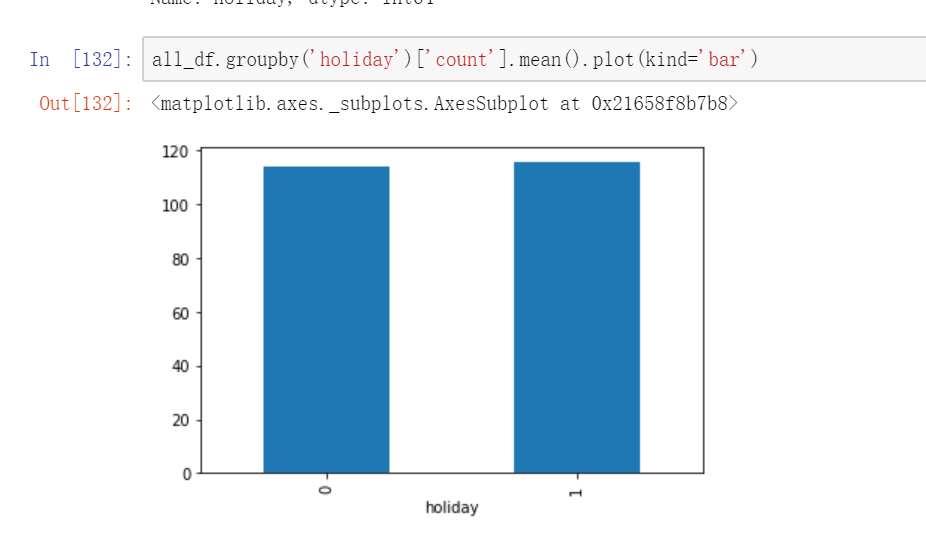
可以看出符合常理，一般春季和冬季比较冷，骑行少，夏季和秋季比较多多



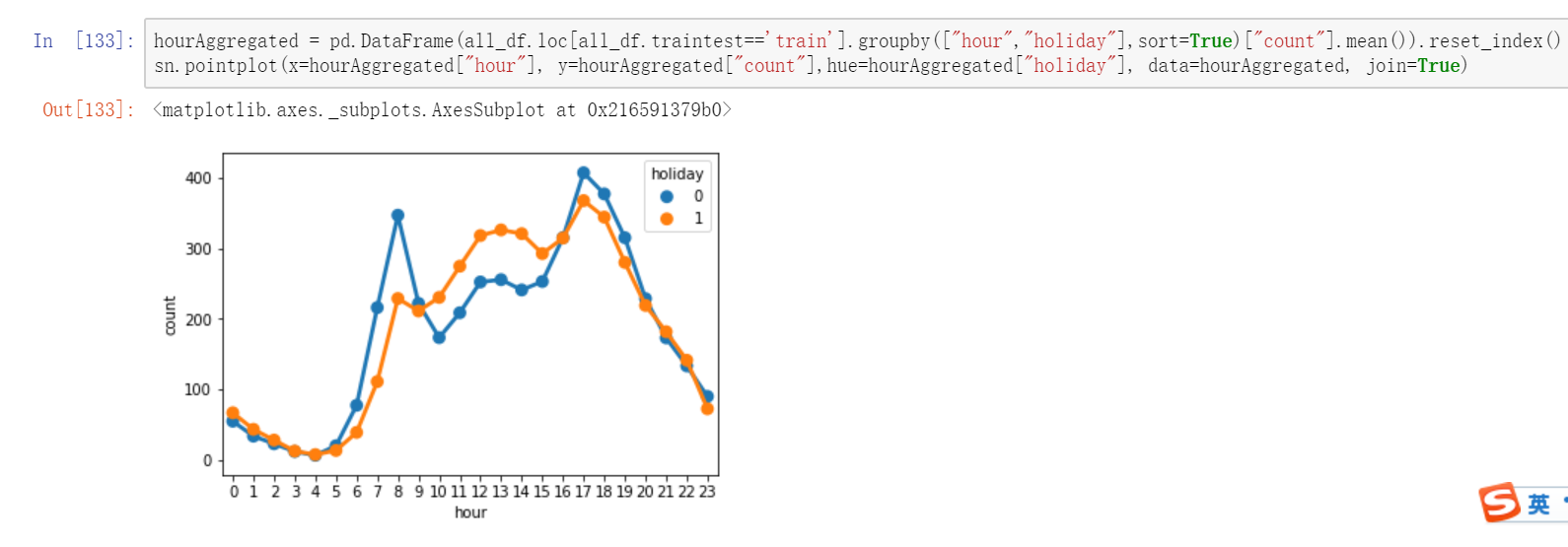
每个季节的每天的分布情况：



### 是否节假日

是否假日的count如下；  


是否是假日貌似没啥影响，从量上看。



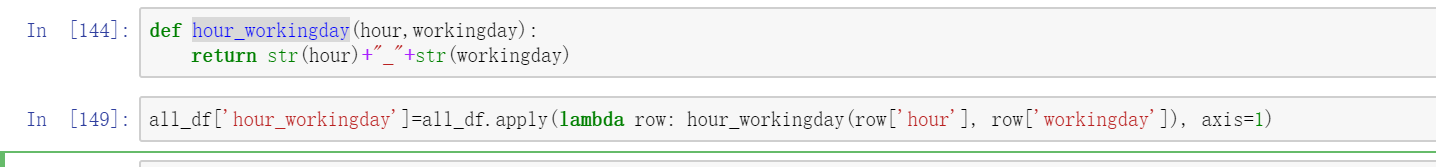
同时貌似和平时没啥关系呢。

### 是否工作日

是否工作日，明显是被我们弄出来的周六周日趋势一样。这个明显需要和小时进行拼接，得到新的特征。

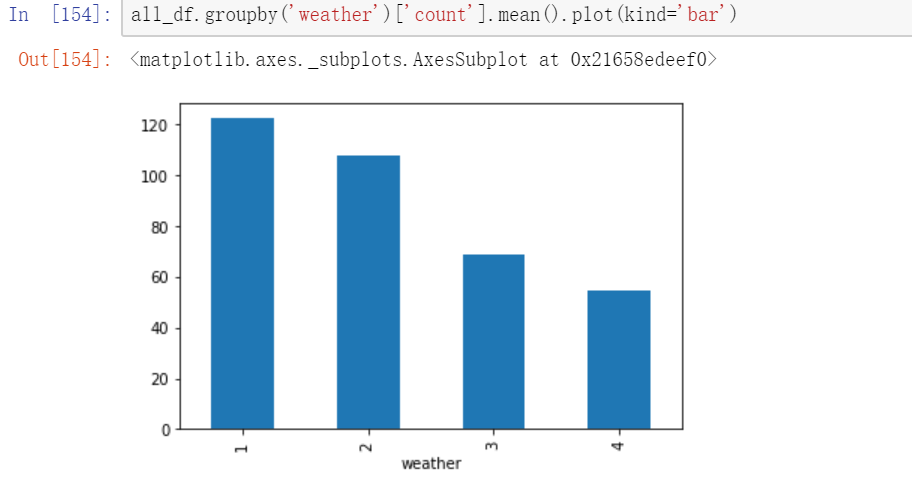


代码如下：

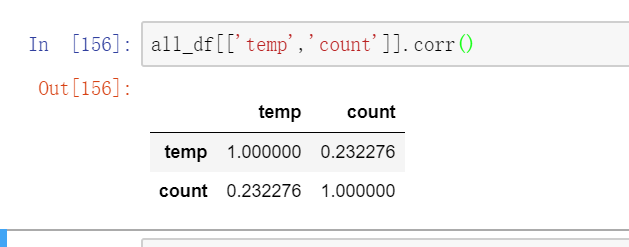


### 天气

天气很明显是一个很好的feature



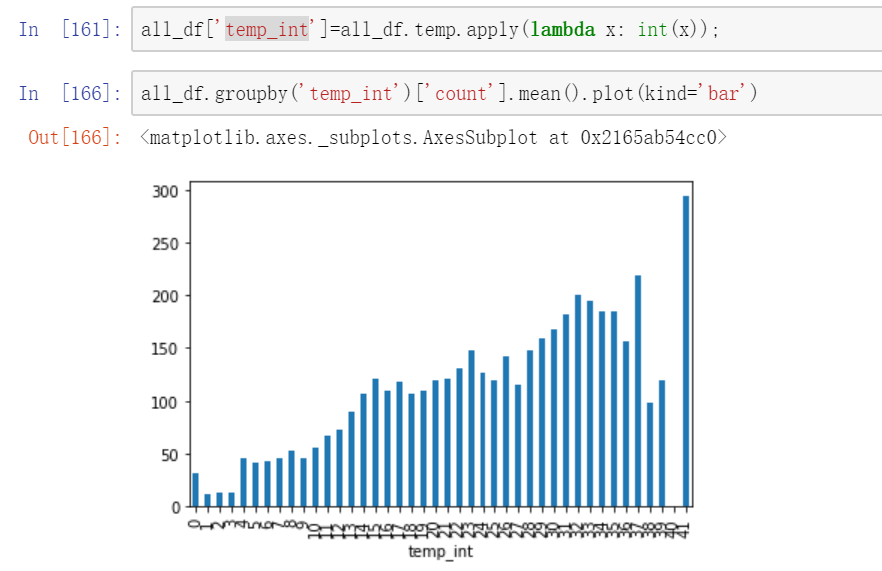
### 温度



相关系数并不高，可以看出来，但是至少是正相关的。

这里就有一个问题，我们猜测温度太高或者太低都有问题是吧？

太高人也不想骑车，太低人也不想骑车对吧？



结果看到的情况如下上图。

貌似数据缺了40度情况下的数据。同时发现41度的温度下，数据量特别大。。。。。

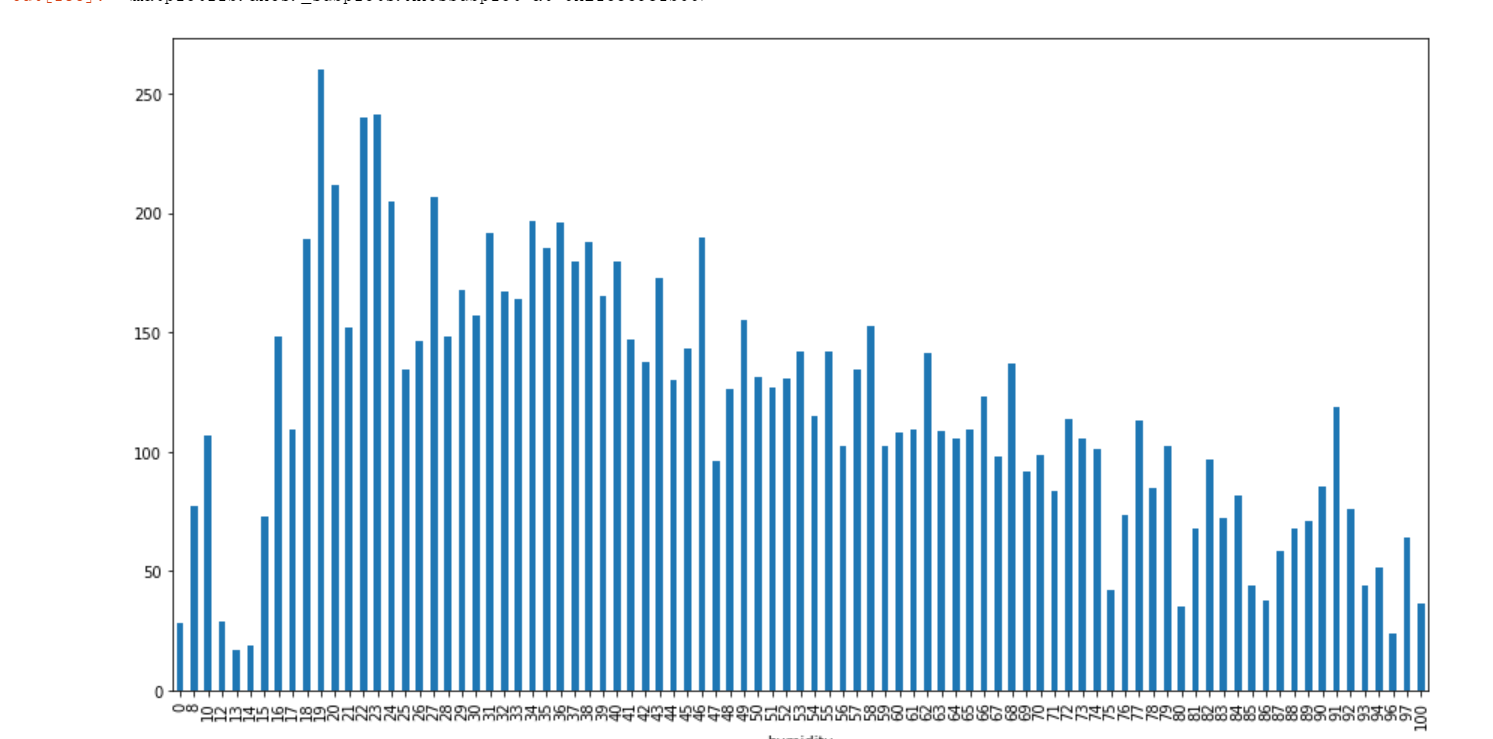
感觉不太符合常理。

### 感觉的温度

这个和温度貌似没啥差别，因为我们可以看到，这个值和温度的相关系数是0.98

### 湿度

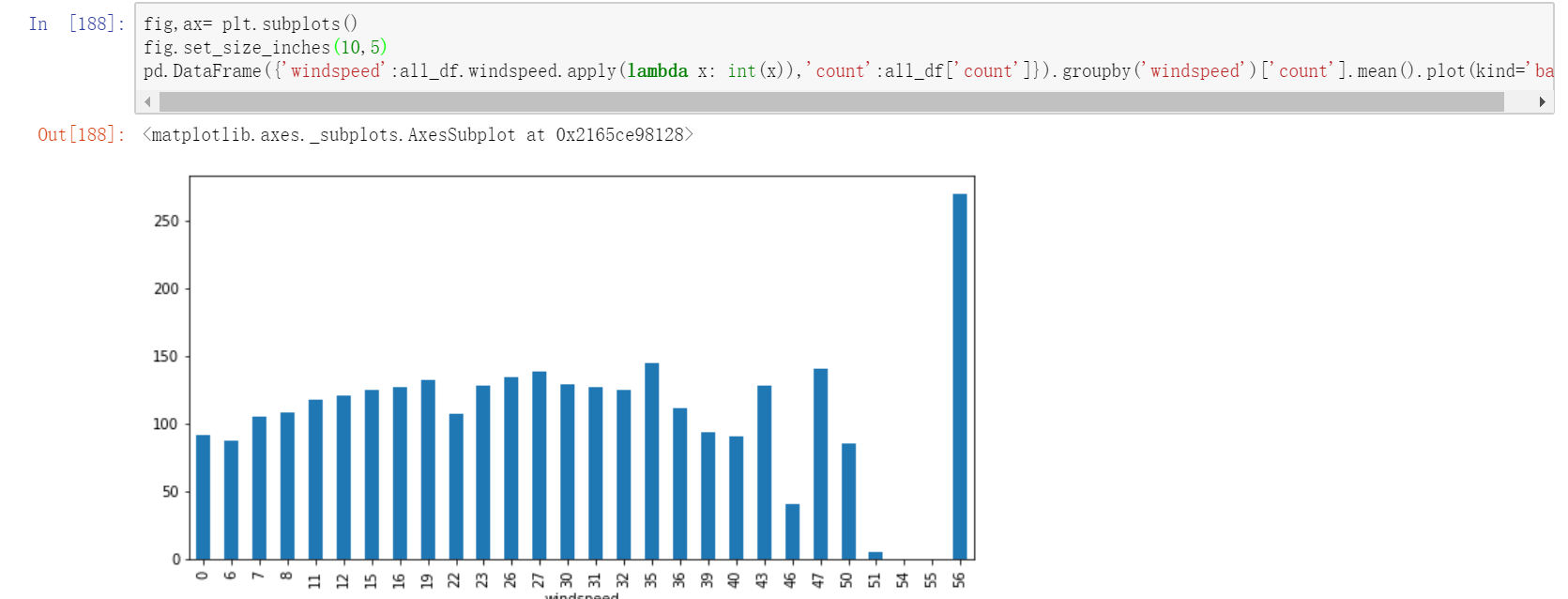
这个值是负相关的。



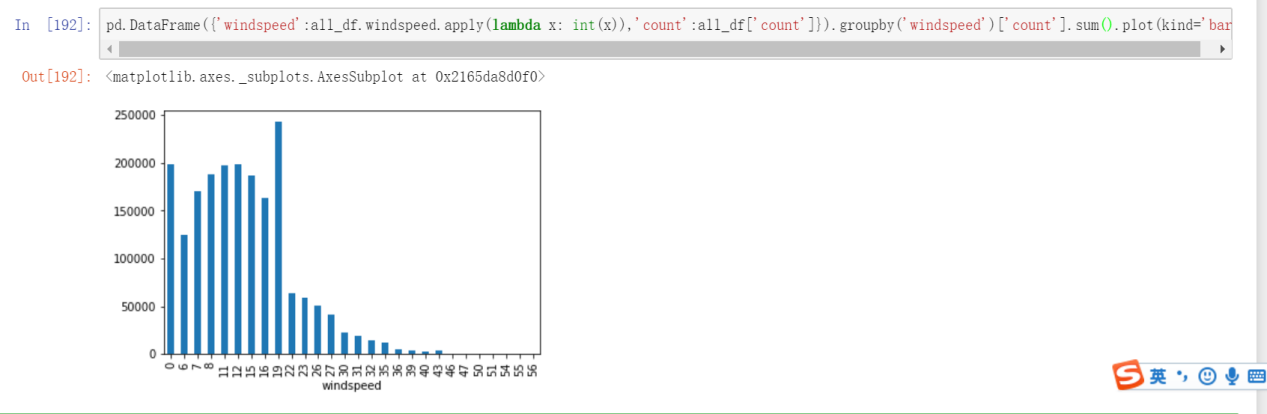
画图一看，果然是哇

### 风速

风速貌似和count没啥关系，相关度是0.07.



确实感觉没啥关系。



不过我们看风速在0的时候明显有些问题。

风速为0可能是由于null填充导致的。我们可能需要预测下。

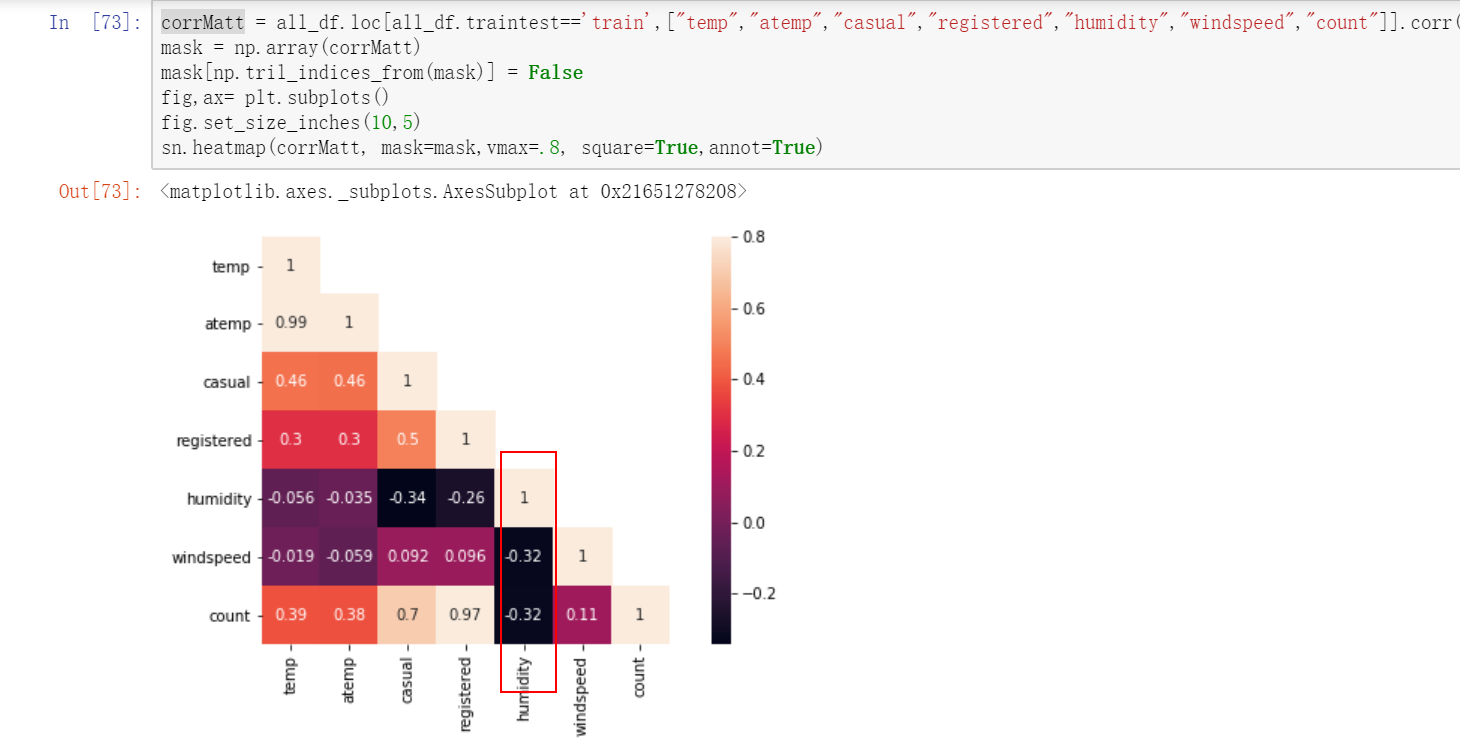
关于这个问题的讨论可以参考：

<https://www.kaggle.com/c/bike-sharing-demand/discussion/10431>

不过风速这这个问题由于和count关系不大，可以放在后面进行优化。

### 数据详情

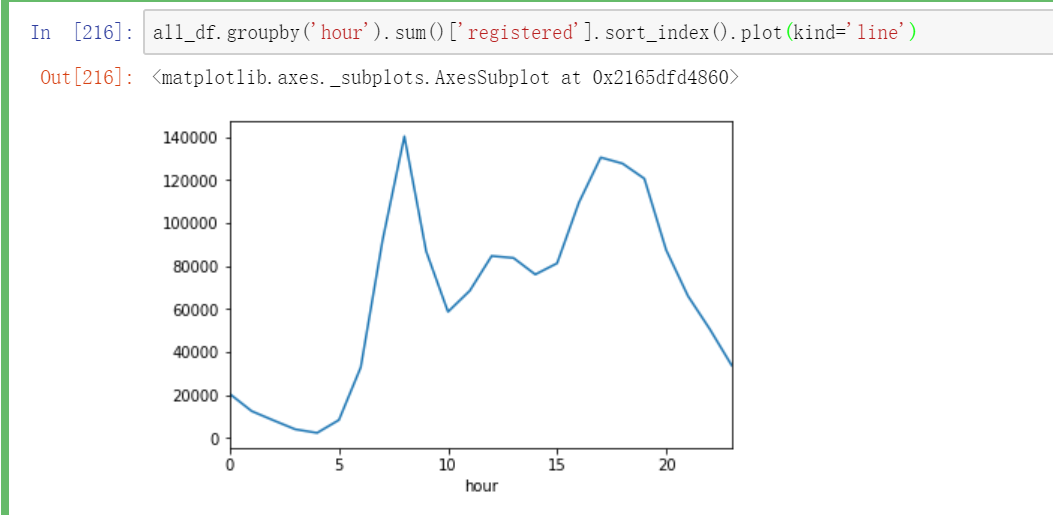
## 各个特征相关系数



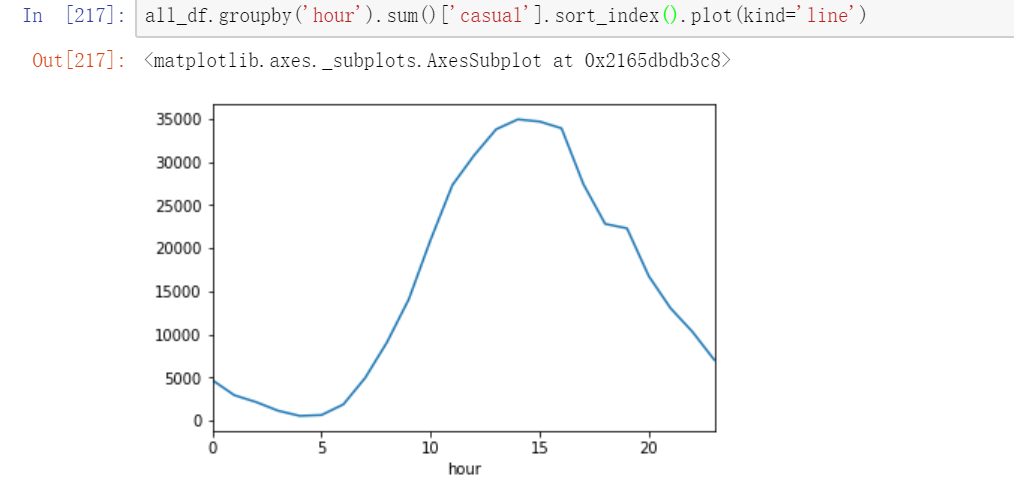
可以看到湿度和count成负相关，temp和count正相关，同时貌似和风速是独立的，相关系数很小，只有0.1

对于注册用户和非注册用户可以说是差别蛮大的：

注册用户如下：



非注册用户



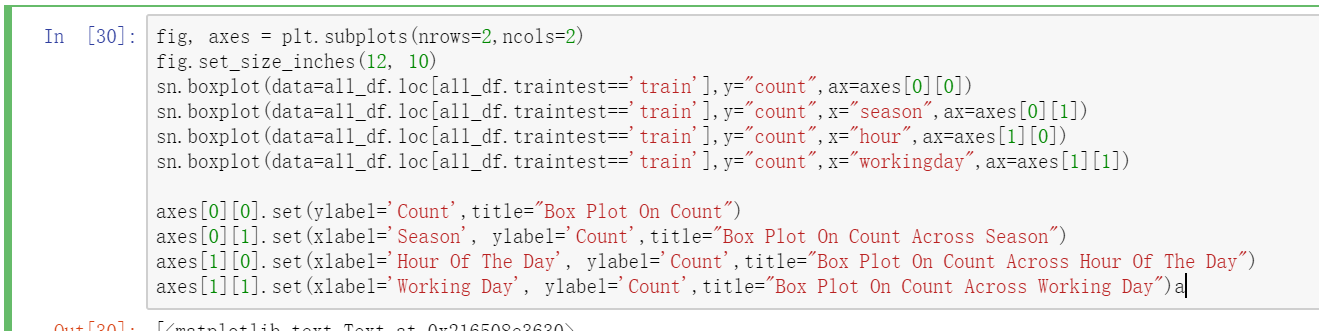
我们可以分别预测注册用户和非注册用户，然后相加进行计算。

# 特征工程

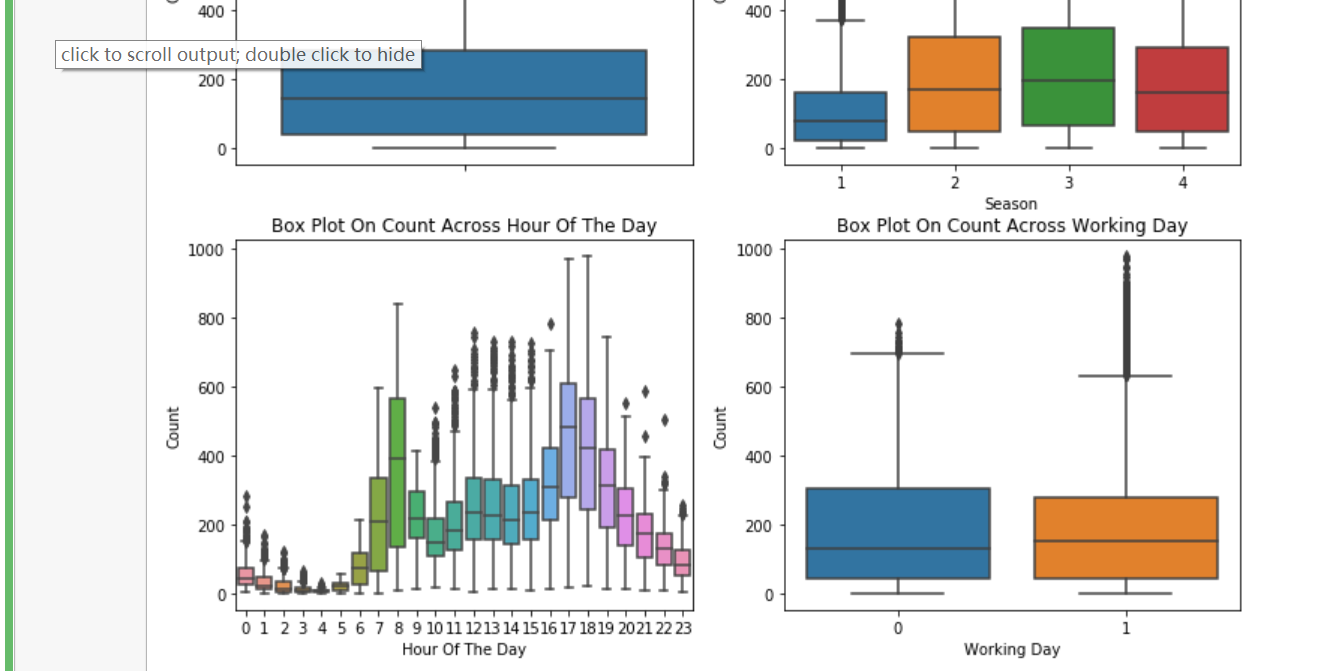
通过上述的数据探探，得到以下结论：

1. 根据日期这个时间戳，我们拆分得到年、月、日、星期、小时
2. 根据日期星期、小时的使用量，构建新的特征
3. 根据工作日和小时，构建新的特征
4. 天气特征ok
5. 温度正相关，但是相关的不够
6. 湿度负相关
7. 风速没啥关系。
8. 我们的label 是需要进行log1p变换的。尽量使得目标预测值保持一正太分布
9. 风速为0可能是缺失值造成的。

## 离群点剔除

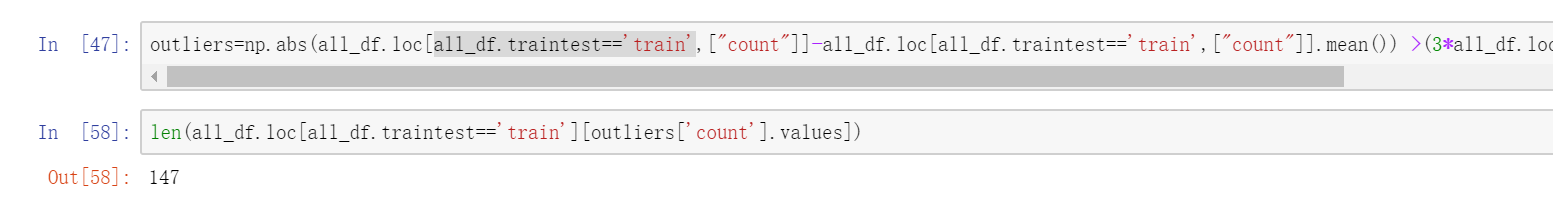


显示



剔除离群点

离群点个数：



147个，还好。

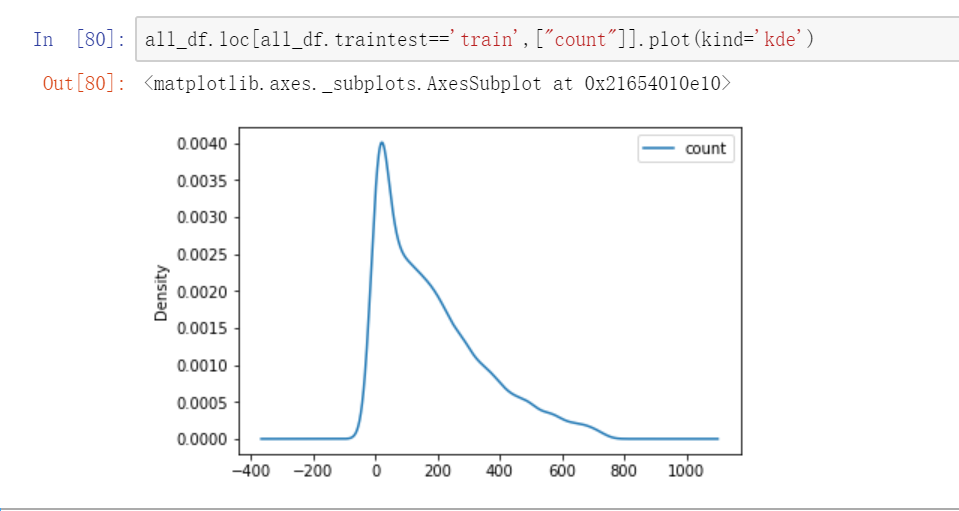
剔除的话不影响。

代码如下：  
len(all\_df.loc[all\_df.traintest=='train'][goodpoints['count'].values])

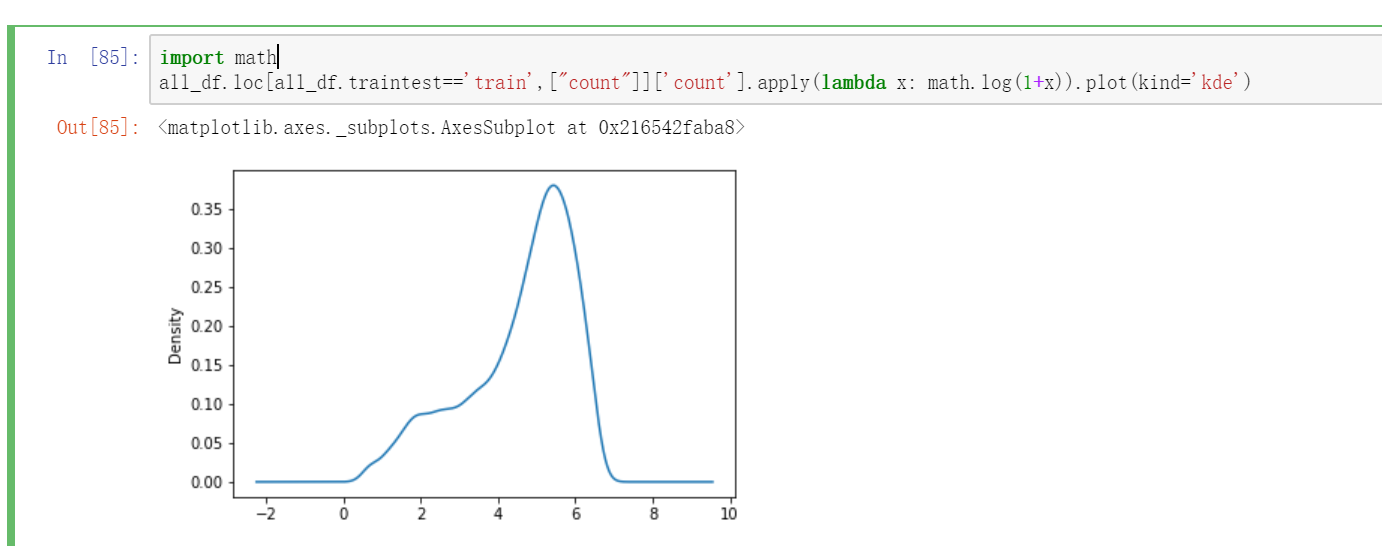
all\_df=pd.concat((all\_df.loc[all\_df.traintest=='train'][goodpoints['count'].values],all\_df.loc[all\_df.traintest=='test']))

## 目标的正态化

这是一个回归问题，所以如果目标值是服从正太分布的话，对于很多模型的效果是好的。



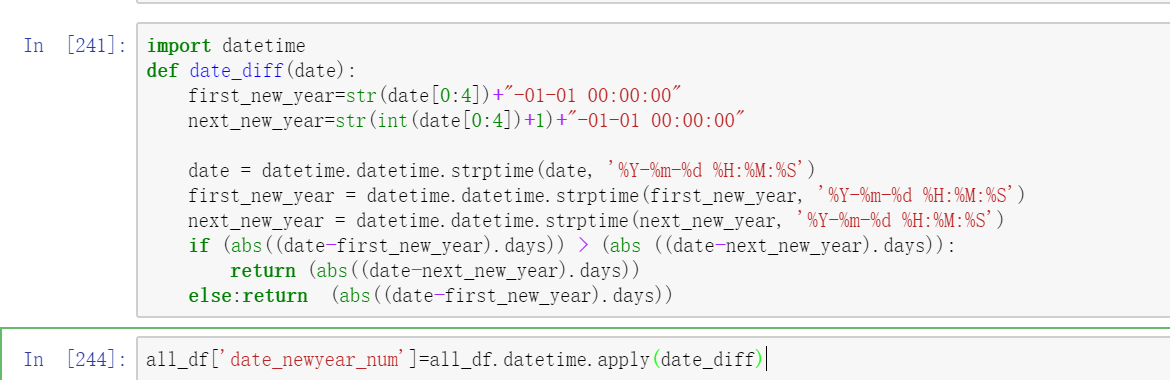
目标进行log1p变换



好看些，但是还是不是一个比较好的正太分布。

## 缺失值填充(风速为0)

## 日期计算



计算当天到最近一次新年的时间。

## 归一化

### 类比型

1）all\_df=pd.get\_dummies(all\_df,columns=['season'])

2）all\_df=pd.get\_dummies(all\_df,columns=['weather'])

3）Holiday 和workingday 不需要，因为本来就是两个值，0和1.

4)month 和hour

all\_df=pd.get\_dummies(all\_df,columns=['hour\_week\_section'])

all\_df=pd.get\_dummies(all\_df,columns=['hour\_workingday'])

### 数值型

对温度做归一化

scaler = preprocessing.StandardScaler()

temp\_scale\_param = scaler.fit(all\_df[['temp']])

all\_df['temp\_scaled'] = scaler.fit\_transform(all\_df[['temp']], temp\_scale\_param)

对感觉的温度做归一化

scaler = preprocessing.StandardScaler()

atemp\_scale\_param = scaler.fit(all\_df[['atemp']])

all\_df['atemp\_scaled'] = scaler.fit\_transform(all\_df[['atemp']], atemp\_scale\_param)

日期到最近一次新年的归一化

scaler = preprocessing.StandardScaler()

date\_newyear\_num\_scale\_param = scaler.fit(all\_df[['date\_newyear\_num']])

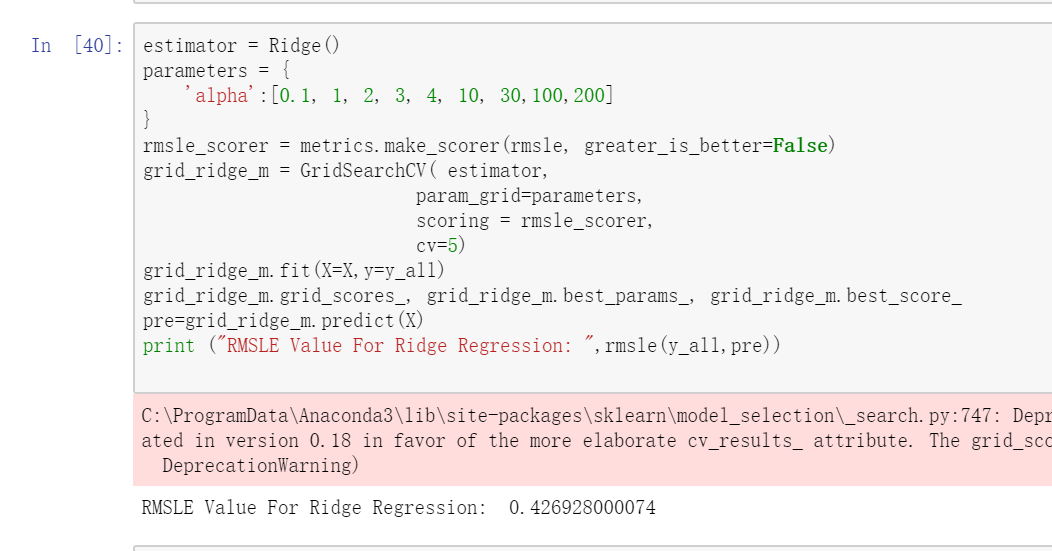
all\_df['date\_newyear\_num\_scaled'] = scaler.fit\_transform(all\_df[['date\_newyear\_num']], date\_newyear\_num\_scale\_param)

同理对风速和湿度都做归一化处理

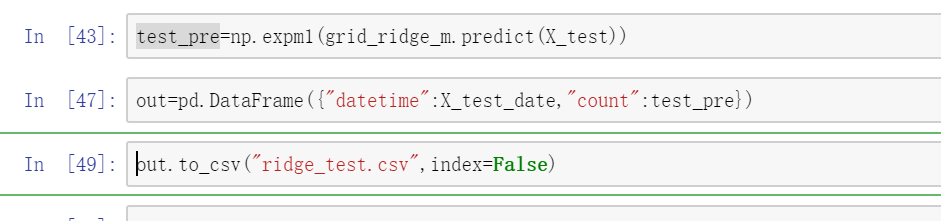
# 模型调优

## Ridge 和lasso

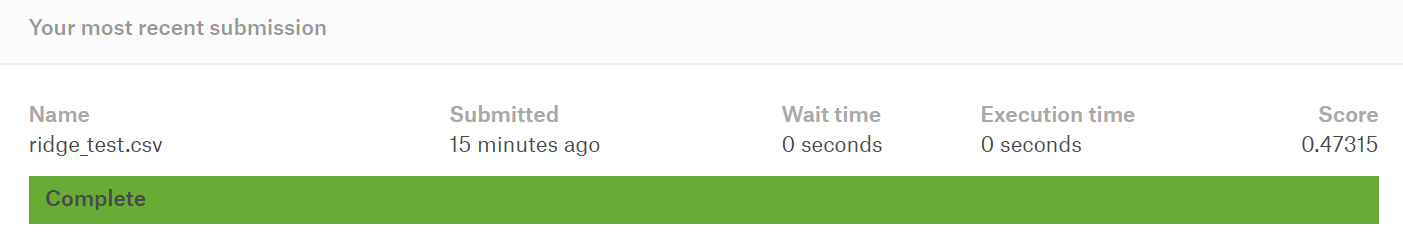
采用ridge 搞一把试试



0.42看起来还不错。当然是越小越好喽。



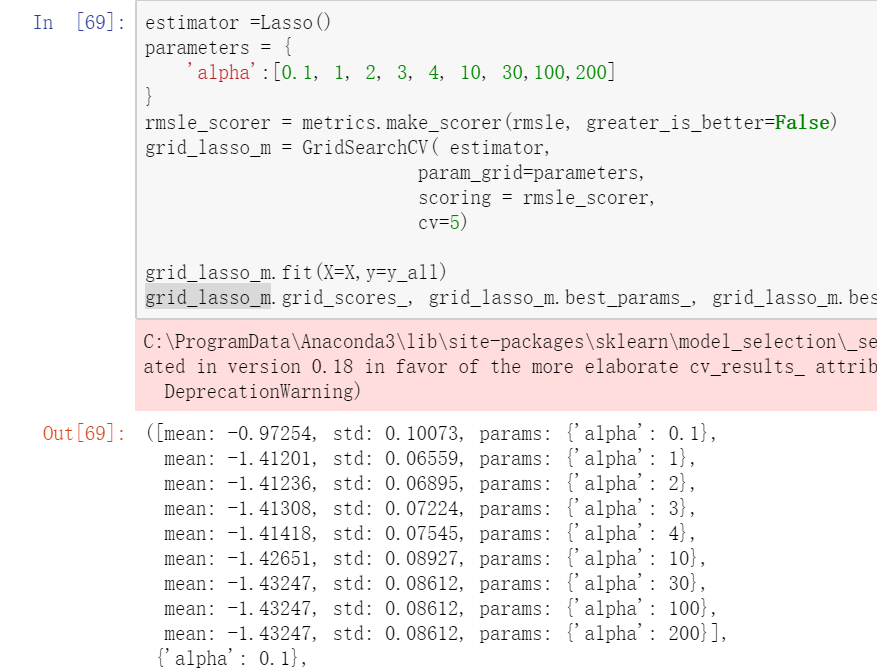
提交试试

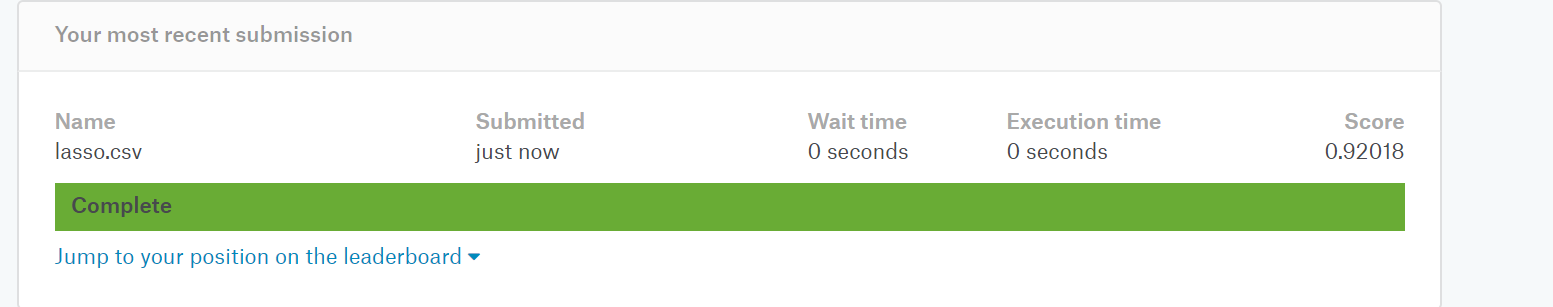


还需要提高，可以看到最新的第一是：0.337。可以把整个leader board 下载下来，因为该比赛已经停止，所以看不到排名。我们大致排名在30%左右。降到0.4就很不错了。

对于lasso

代码如下：





明显lasso放弃。

## 随机森林

太难调整了。。。。主要是俺的笔记本差呀，哎，运行半天出不来的呢。

这里我就使用随便一个参数啦，实际工作中不要这样，需要不断调整参数的。

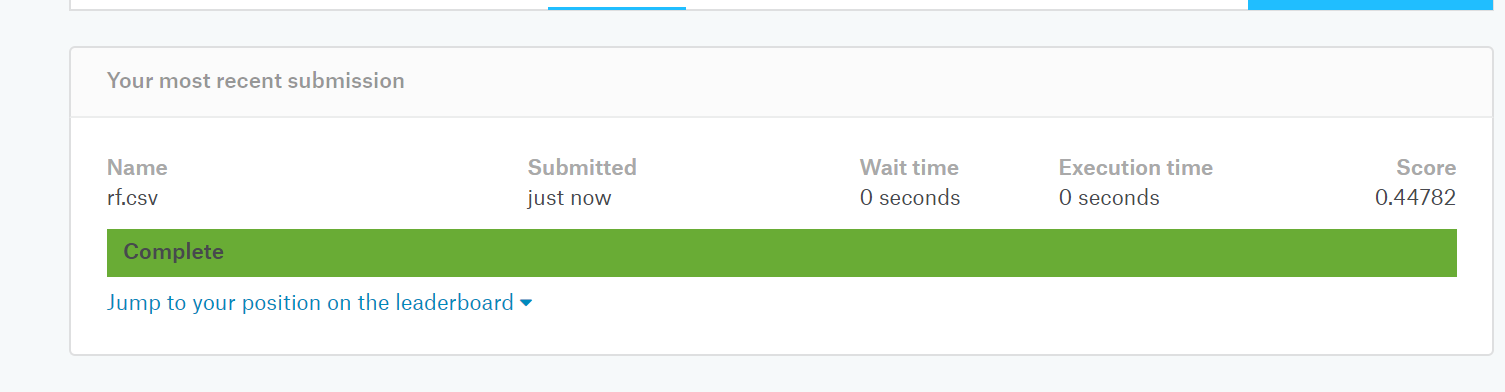
rf=RandomForestRegressor(random\_state=0, n\_estimators=3000, n\_jobs=-1,oob\_score=True)

rf.fit(X=X,y=y\_all)

pre=rf.predict(X)

print ("RMSLE Value For Ridge Regression: ",rmsle(y\_all,pre))

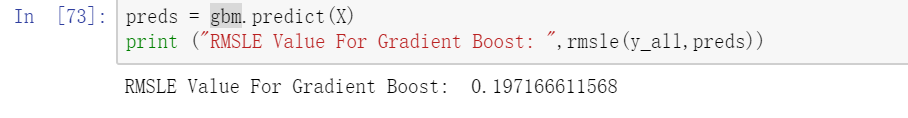
得到结果如下：



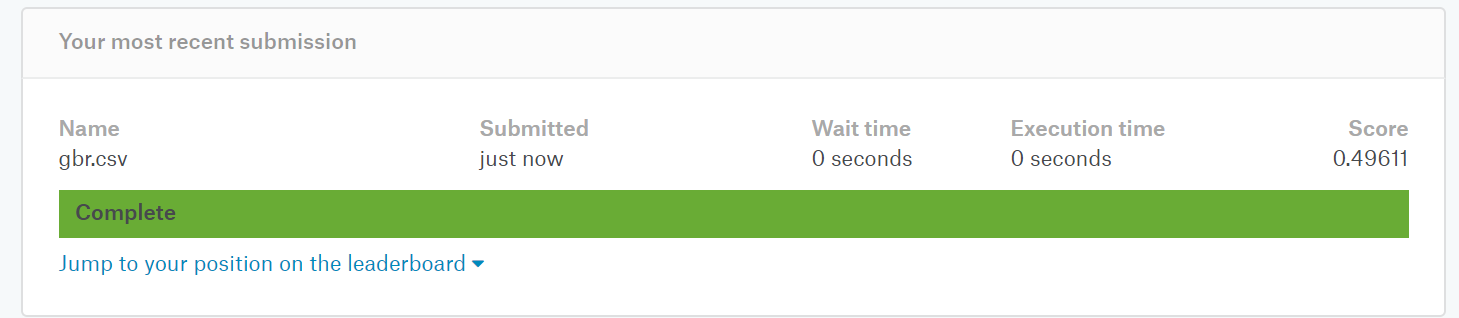
这个结果大约在top20%左右。这个可以作为一个baseline，后面进行模型调优以这个为基准参照。

## GBR（GradientBoostingRegressor）

采用gbr测试下。



估计效果应该不怎么好。



果然，gbr效果不咋地。

这里没有调参，电脑烧不起啊。

# 提交

最后以随机森林为结果进行提交。排名约top20%

# 待改进

1. 风速为0可能需要重新进行填充
2. 特征的太稀疏，采用了小时和工作日进行拼接，这样产生了48个长度特征，导致训练过慢。
3. 还需要进行特征选择，但是对于得分的提升不一定有用。
4. 对于特征组合可能需要采用GBDT进行产生，而这里只是觉得那些是应该组合在一起的，所以手撸的和产生的，相对来说，模型生产的特征组合性能更好点。

**特征工程比模型重要的多，数据的认识非常重要，非常重要非常重要！！！**

# 参考连接

<https://www.kaggle.com/viveksrinivasan/eda-ensemble-model-top-10-percentile/notebook>

<https://www.kaggle.com/c/bike-sharing-demand/discussion/11525>