计算社会科学导论
—— 典型数据类型与分析方法

金耀辉、许岩岩

2023年3月23日

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

不同的数据形式

- **表格数据**
- **时间序列数据**
- **图网络数据**
- **时空数据**
- 半结构化数据
- 文本数据
- 图像数据

调查问卷

最常见的数据形式：

编号	开始答题时间	结束答题时间	1．你的学号是：	2．对Python的掌握...	3．对统计学的了解...
8	2023-02-16 16:44:03	2023-02-16 17:36:39	52	2	3
7	2023-02-16 16:44:10	2023-02-16 16:47:10	52	3	3
6	2023-02-16 16:44:01	2023-02-16 16:45:09	52	3	3
5	2023-02-16 16:44:06	2023-02-16 16:45:05	52	3	4
4	2023-02-16 16:44:05	2023-02-16 16:45:04	52	2	2
3	2023-02-16 16:44:03	2023-02-16 16:45:02	51	2	4
2	2023-02-16 16:44:00	2023-02-16 16:44:56	52	2	3
1	2023-02-16 16:44:08	2023-02-16 16:44:48	52	2	1

表格数据Tabular data

- 最常见的数据形式，通常以一行作为一个样本，一列表示一组特征，行列不需要包含顺序关系；
- 每一列一般需要**具备相同的数据类型**，不同特征的数据类型可能不同；
- 列的类型通常较为固定，一般先有结构再有数据，称为**结构化数据**；
- 在python中通常使用pandas库处理表格数据；

用户ID	姓名	性别	年龄	存款
34a3c68c-3592-44e7-b9ff-311e4467c9e4	张三	男	18	16356
72b1670c-e944-4b55-aecb-fbd4918e494b	李四	男	23	459134
48e0bd1b-fbf4-45b8-bad6-d0caed3fd785	王五	女	16	615613
b8d95db6-3403-4deb-a1b0-85f01365256b	甄路人	女	72	1269833
92341f82-a4ca-4b2c-857d-b5917492c624	贾老板	男	29	156863

```
import pandas as pd
```

数据读取

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
import pandas as pd
pd.set_option('display.max_rows', 500) # 最大显示行数
pd.set_option('display.max_columns', 500) # 最大显示列数
```

```
df = pd.read_csv("../data/alt_fuel_stations_Mar9-2023.csv", low_memory=False)#
df.head(2)#展示前两行
```

	Fuel Type Code	Station Name	Street Address	Intersection Directions	City	State	ZIP	Plus4	Station Phone	Status Code	Expected Date	Groups With Access Code	Access Days Time	Cards Accepted	BD Blends	NG Fill Type Code	NG PSI	EV Level1 EVSE Num
0	CNG	Arkansas Oklahoma Gas Corp	2100 S Waldron Rd	NaN	Fort Smith	AR	72903	NaN	479-783-3181	E	NaN	Public - Credit card at all times	24 hours daily	FuelMan M V Wright_Exp	NaN	Q	3600	NaN
1	CNG	Clean Energy - Logan International Airport	1000 Cottage St Ext	From Route 1, take the first exit after Callah...	East Boston	MA	02128	NaN	866-809-4869	E	NaN	Public - Credit card at all times	24 hours daily; call 866-809-4869 for Clean En...	A CleanEnergy Comdata D FuelMan M V Voyager Wr...	NaN	Q	3000 3600	NaN

表格基本信息

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

列名

```
df.columns#所有列名
```

```
Index(['Fuel Type Code', 'Station Name', 'Street Address',
       'Intersection Directions', 'City', 'State', 'ZIP', 'Plus4',
       'Station Phone', 'Status Code', 'Expected Date',
       'Groups With Access Code', 'Access Days Time', 'Cards Accepted',
       'BD Blends', 'NG Fill Type Code', 'NG PSI', 'EV Level1 EVSE Num',
       'EV Level2 EVSE Num', 'EV DC Fast Count', 'EV Other Info', 'EV Net
work',
```

表格基本信息

每列数据类型

```
df.dtypes#数据类型
```

```
Fuel Type Code                  object
Station Name                    object
Street Address                  object
Intersection Directions         object
City                            object
State                           object
ZIP                             object
Plus4                          float64
Station Phone                   object
Status Code                     object
Expected Date                  float64
Groups With Access Code         object
Access Days Time                object
Cards Accepted                  object
BD Blends                       object
NG Fill Type Code               object
NG PSI                          object
EV Level1 EVSE Num             float64
EV Level2 EVSE Num             float64
```

Pandas 类型	Python 类型	NumPy类型	使用场景
object	str or mixed	string_, unicode_, mixed types	文本或者混合数字
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	整型数字
float64	float	float_, float16, float32, float64	浮点数字
bool	bool	bool_	True/False 布尔型
datetime64[ns]	datetime.datetime	object	日期时间（将从pandas库移除，直接使用datetime库）
timedelta[ns]	datetime.timedelta	object	两个时间之间的距离，时间差

数据描述

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
df.describe()
```

	Plus4	Expected Date	EV Level1 EVSE Num	EV Level2 EVSE Num	EV DC Fast Count	Latitude	Longitude	ID	Federal Agency ID	CNG Dispenser Num	CNG Total Compression Capacity	CNG Storag Capacit
count	0.0	0.0	108.000000	52559.000000	8496.000000	68136.000000	68136.000000	68136.000000	38.000000	812.000000	510.000000	277.00000
mean	NaN	NaN	3.046296	2.235393	3.886417	39.087490	-95.765221	165252.689885	15.421053	2.166256	858.911765	50743.75451
std	NaN	NaN	4.518672	2.187715	4.650965	5.749795	19.634688	58717.695318	4.705555	7.119948	798.696501	59213.52237
min	NaN	NaN	1.000000	1.000000	1.000000	-90.000000	-164.848855	73.000000	5.000000	1.000000	2.000000	77.00000
25%	NaN	NaN	1.000000	2.000000	1.000000	34.241337	-117.838147	122362.750000	16.000000	1.000000	300.000000	33000.00000
50%	NaN	NaN	1.000000	2.000000	2.000000	39.319311	-91.311308	175721.500000	16.000000	2.000000	740.500000	36000.00000
75%	NaN	NaN	2.000000	2.000000	6.000000	42.875098	-78.753091	211735.250000	16.000000	2.000000	1200.000000	60000.00000
max	NaN	NaN	22.000000	80.000000	80.000000	64.852466	62.452833	255741.000000	33.000000	202.000000	8000.000000	593136.00000

数据描述

偏度与峰度(仅数值类型)

```
df.skew(numeric_only=True, skipna=True)
```

Plus4	NaN
Expected Date	NaN
EV Level1 EVSE Num	2.762551
EV Level2 EVSE Num	11.577769
EV DC Fast Count	3.407455
Latitude	-0.434449
Longitude	-0.292289
ID	-0.623570
Federal Agency ID	0.482534
CNG Dispenser Num	27.342476
CNG Total Compression Capacity	2.912387
CNG Storage Capacity	6.170495
RD Blends (French)	NaN
RD Maximum Biodiesel Level	9.345071

dtype: float64

```
df.kurtosis(numeric_only=True, skipna=True)
```

Plus4	NaN
Expected Date	NaN
EV Level1 EVSE Num	6.973327
EV Level2 EVSE Num	231.695556
EV DC Fast Count	24.941583
Latitude	4.562168
Longitude	-0.858999
ID	-0.416493
Federal Agency ID	5.568959
CNG Dispenser Num	767.842702
CNG Total Compression Capacity	16.672870
CNG Storage Capacity	48.764069
RD Blends (French)	NaN
RD Maximum Biodiesel Level	85.648150

dtype: float64

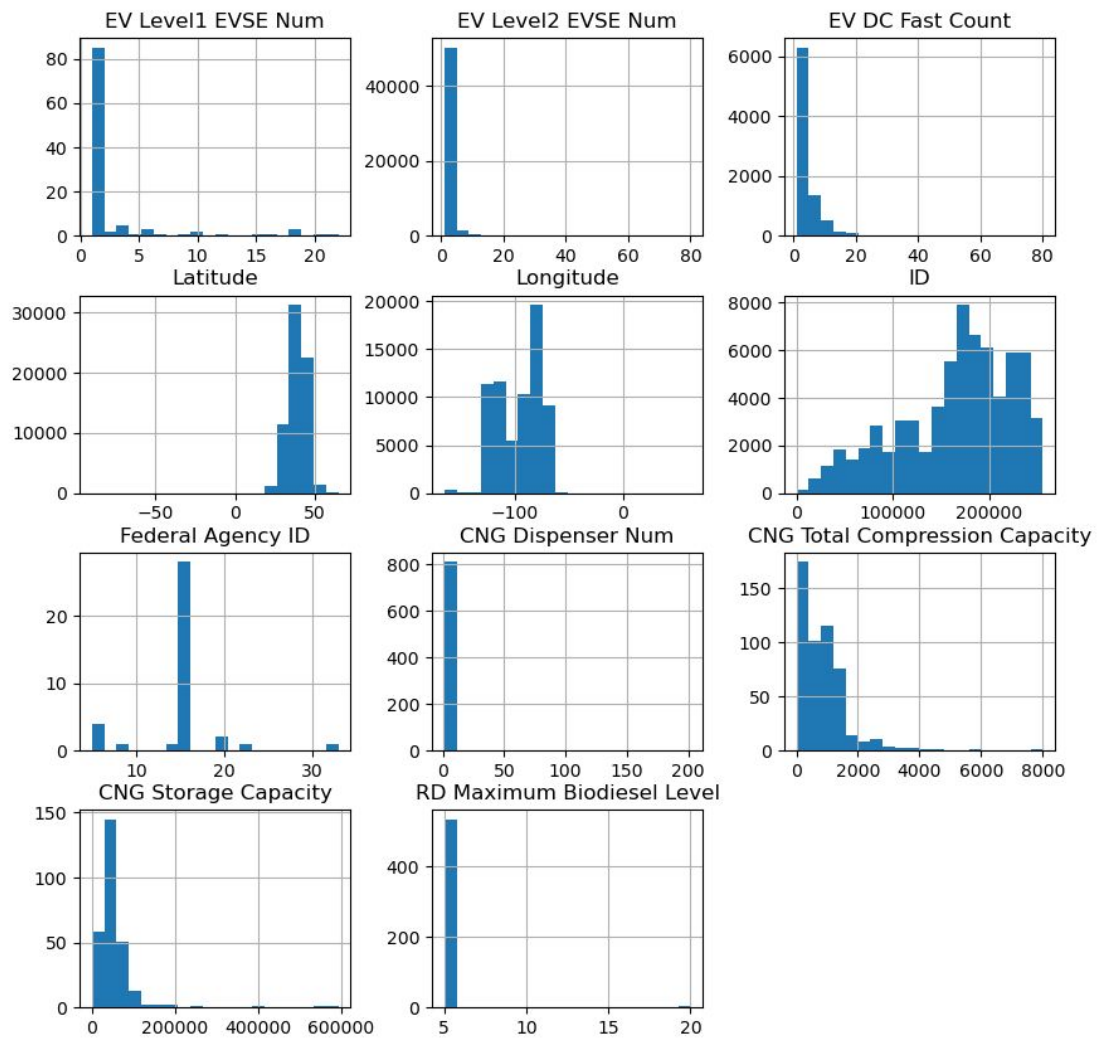
对表格数据的探索

- 数据类型转换
  - 从文本数据中提炼量化数据类型
- 数据完整性
  - 是否存在缺失值？
- 数据分布
  - 是否存在异常值？
  - 离散取值或是连续取值？
  - 数据分布的模式，正态分布、log-norm、均匀分布？
- 什么是有意义的特征
  - 手动筛选、组合特征
  - 机器学习降维：PCA\LDA

利用histogram观察数据分布

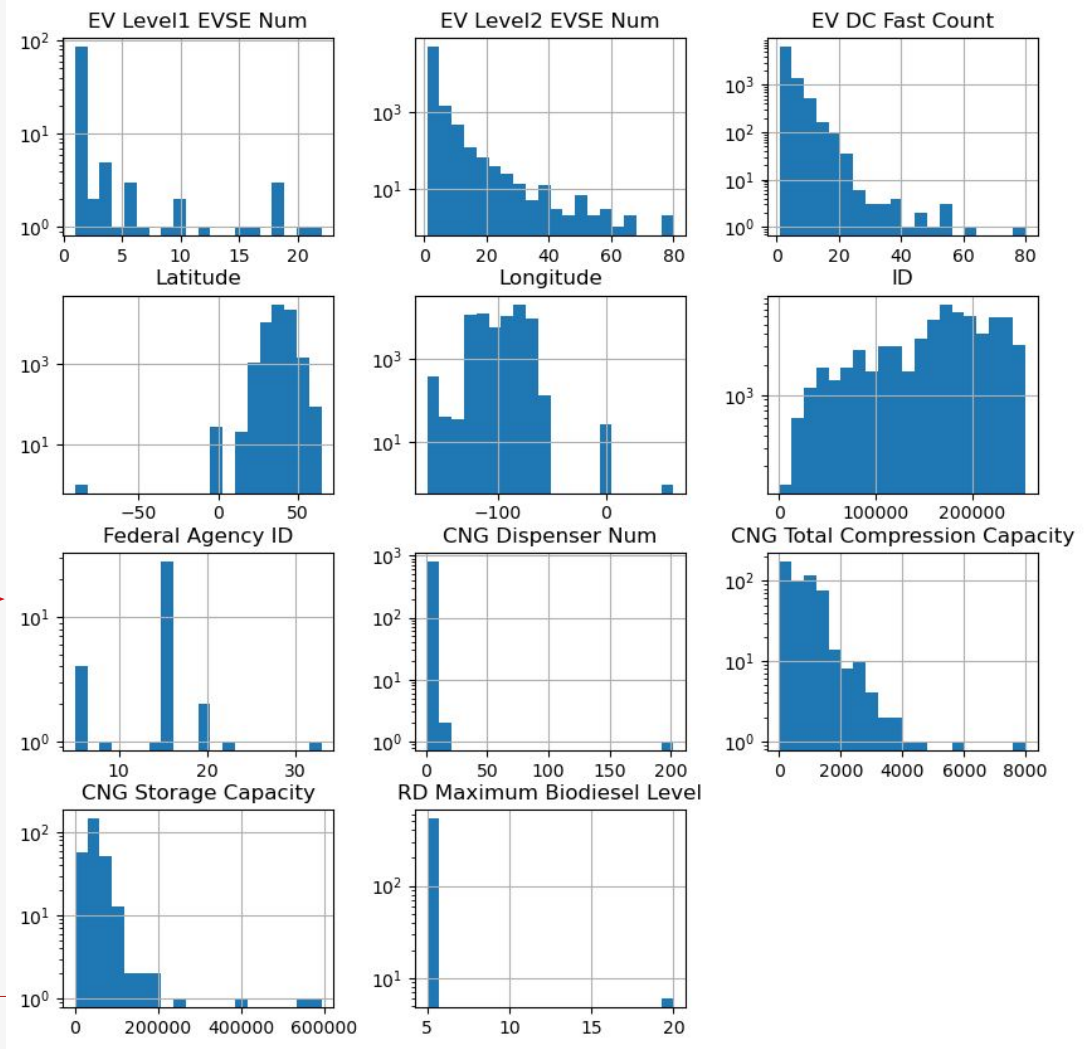
上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
df.hist(figsize=[10, 10], log=False, bins = 20)
```



Log Scale

```
df.hist(figsize=[10, 10], log=True, bins = 20)
```



缺失值的观察与处理

- 狭义的缺失值特指数据为空(NaN)，在pandas中可以用df.isna()来判断每个单元格是否为空，为空则为True；

```
df.isna()
```

	Fuel Type Code	Station Name	Street Address	Intersection Directions	City	State	ZIP	Station Phone	Status Code	Groups With Access Code	Access Days Time	Cards Accepted	BD Blends	NG Fill Type Code	NG PSI	EV Level1 EVSE Num	EV Level2 EVSE Num	EV DC Fast Count	E Othe Inf
0	False	False	False	True	False	False	False	False	False	False	False	False	True	False	False	True	True	True	Tru
1	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	True	True	True	Tru
2	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	True	True	True	Tru
3	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	True	True	True	Tru
4	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	True	True	True	Tru
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
68131	False	False	False	True	False	False	False	False	False	False	False	True	True	True	True	True	False	True	Tru
68132	False	False	False	True	False	False	False	False	False	False	False	False	True	True	True	True	False	True	Tru
68133	False	False	False	True	False	False	False	False	False	False	False	False	True	True	True	True	False	True	Tru
68134	False	False	False	True	False	False	False	False	False	False	True	True	True	True	True	True	True	False	Tru
68135	False	False	False	True	False	False	False	False	False	False	True	True	True	True	True	True	False	True	Tru

68136 rows × 66 columns

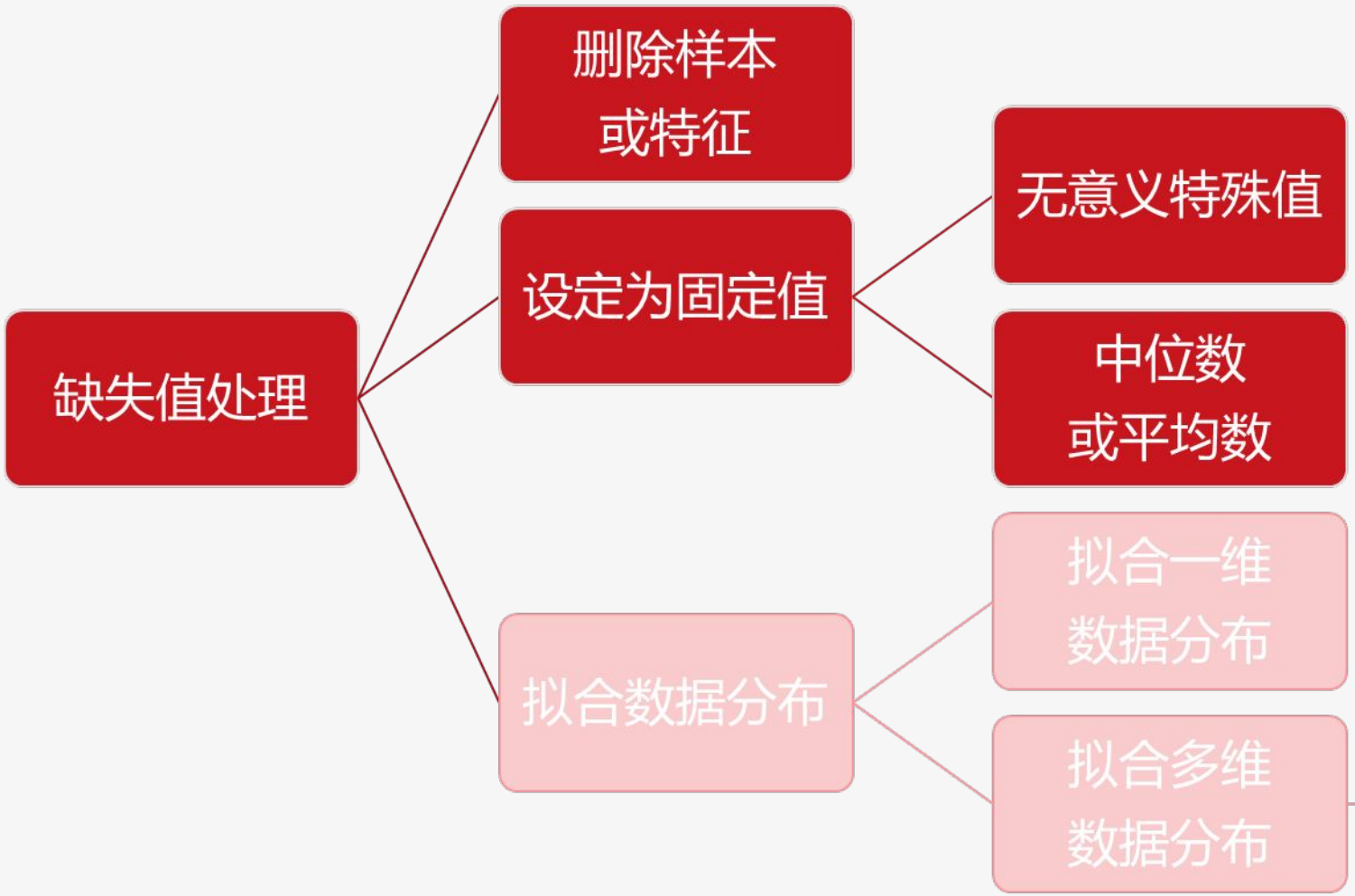
缺失值的观察与处理

- any和all函数表示**或/与**操作，axis参数表示行或列方向
- 右图中df.isna().any(axis = 0)用来**判断列是否存在缺失值**（在列方向上只要存在一个True，则这一列的计算结果为True，也就是本列存在NaN；）

```
temp = df.isna().any(axis = 0)#存在NaN的columns
temp[temp]#这里只取temp中为True的columns显示
```

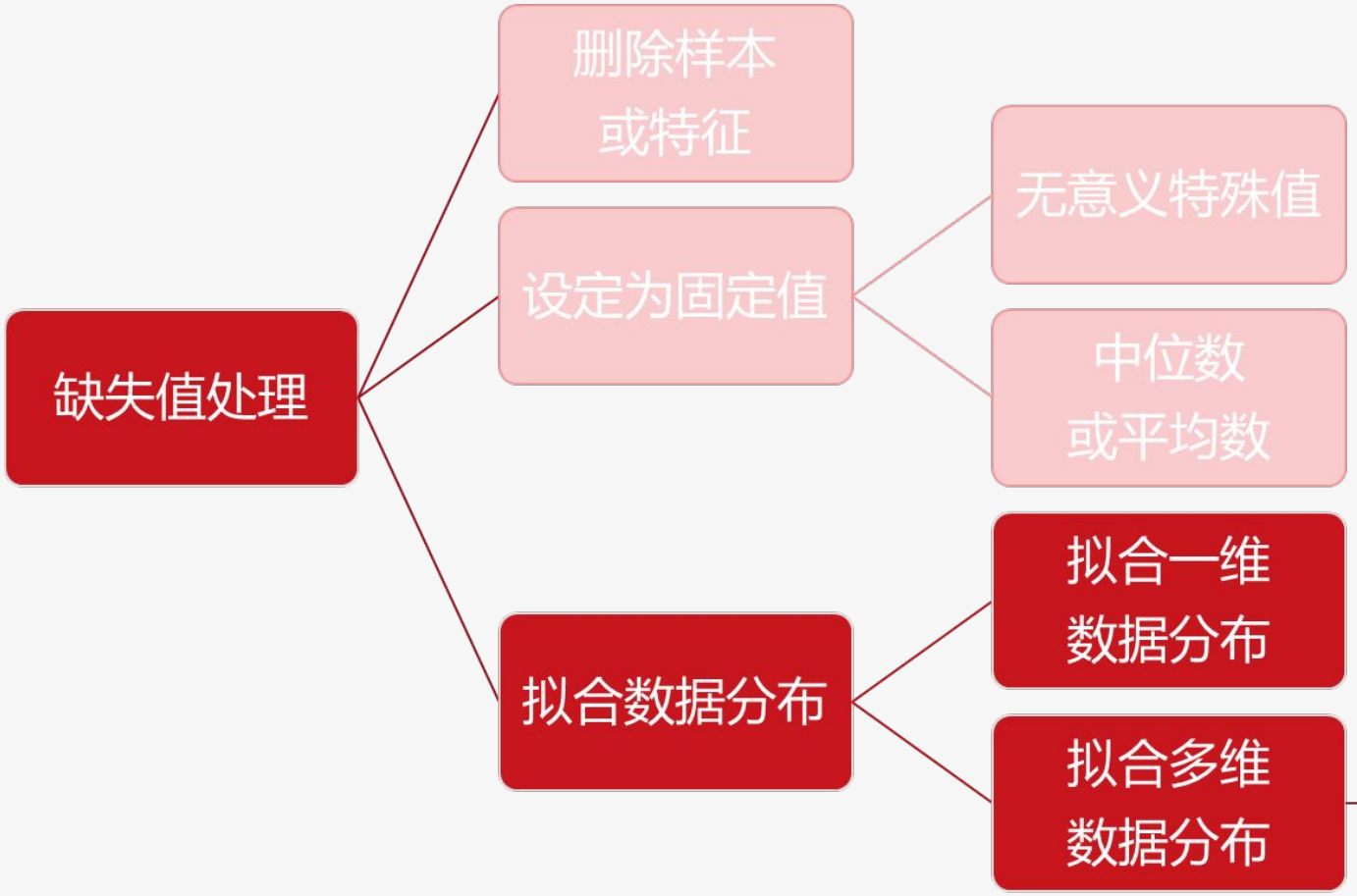
Station Name	True
Intersection Directions	True
State	True
Plus4	True
Station Phone	True
Expected Date	True
Access Days Time	True
Cards Accepted	True
BD Blends	True
NG Fill Type Code	True
NG PSI	True
EV Level1 EVSE Num	True
EV Level2 EVSE Num	True
EV DC Fast Count	True
EV Other Info	True
EV Network	True
EV Network Web	True
Geocode Status	True
Date Last Confirmed	True

缺失值处理



- 缺失值比例过高时选择删除样本或整列特征；
- 根据模型特点，决策树和基于决策树的模型可以将缺失值设定为一个特殊值；
- 在缺失比例较少的情况下，可以采用其他样本的中位数或平均数填充；

缺失值处理



- 在缺失比例较高的情况下，采取中位数或平均数会改变样本的整体分布，此时可以先拟合其他数据的概率分布，再根据分布填补缺失值；
- 仅拟合一维数据忽视了其他维度的信息，可以拟合多维数据，根据其他维度的信息来获得缺失值的概率分布；

删除

设定阈值=0.2，如果一列特征中**非空值占比少于阈值**，则整列删除；

右图展示了每列特征的非空值占比；

```
#缺失值处理：
threshold=0.2#需要至少有threshold*len(df)的有效数值
df.dropna(axis = 1,thresh = int(threshold*len(df)),inplace = True)
1-df.isna().sum(axis = 0)/len(df)#每列非na占比
```

```
Fuel Type Code                    1.000000
Station Name                      0.999985
Street Address                    1.000000
City                              1.000000
State                             0.999897
ZIP                               1.000000
Station Phone                     0.989257
Status Code                       1.000000
Groups With Access Code           1.000000
Access Days Time                  0.873444
EV Level2 EVSE Num                0.771384
EV Network                        0.874706
EV Network Web                    0.772646
Geocode Status                    0.999971
Latitude                          1.000000
Longitude                         1.000000
Date Last Confirmed               0.999721
ID                                1.000000
Updated At                        1.000000
Owner Type Code                   0.351151
Open Date                         0.999677
EV Connector Types                0.874413
Country                           1.000000
Groups With Access Code (French)  1.000000
Access Code                       1.000000
Facility Type                     0.319420
EV Pricing                        0.217418
Restricted Access                 0.279412
dtype: float64
```

固定值

```
df['Date Last Confirmed'].unique()
```

```
array(['2023-02-13', '2022-12-13', '2022-06-14', '2022-10-11',
       '2023-03-08', '2022-07-12', '2022-04-06', '2023-01-10',
       '2022-08-10', '2022-05-05', '2021-12-09', '2022-09-14',
       '2022-03-07', '2021-06-07', '2020-02-06', '2022-01-10',
       '2023-03-09', '2021-10-12', '2021-11-04', '2022-11-10',
       '2021-04-08', '2021-05-06', '2021-07-14', '2021-01-14',
       '2021-08-04', '2017-06-02', '2021-09-10', '2021-03-15',
       '2020-06-09', '2022-02-10', '2020-07-13', '2020-08-05',
       '2020-12-03', '2020-10-09', '2020-11-09', '2022-10-22',
       '2021-02-15', '2021-02-22', '2020-03-03', '2023-01-26',
       '2020-05-06', '2020-09-03', '2020-04-08', '2021-11-02',
       '2022-04-13', '2020-10-07', '2021-12-13', '2022-12-12',
       '2018-05-07', '2022-07-08', '2023-01-12', '2022-01-11',
       '2022-03-04', '2019-04-09', '2018-06-11', '2023-02-09',
       '2020-04-10', '2022-11-09', '2022-06-10', '2022-08-09',
       '2021-08-03', '2020-01-06', '2019-01-14', '2017-09-06',
       '2018-03-29', '2019-10-10', '2019-09-11', '2019-11-08',
       '2021-04-12', '2022-10-10', '2021-09-09', '2020-01-30',
       '2019-12-09', '2020-03-05', '2020-08-10', '2019-06-06',
       '2018-11-12', '2018-02-06', '2022-10-03', '2017-10-03',
       '2018-03-08', '2018-01-03', '2019-07-10', '2021-07-15',
       '2017-05-11', '2021-02-12', '2019-08-09', '2017-12-05',
       '2022-10-06', '2019-03-11', '2020-06-08', '2021-01-07',
       '2018-08-07', '2020-05-15', '2023-03-05', '2020-11-03', nan,
       '2019-05-07'], dtype=object)
```

```
df.loc[:,'Date Last Confirmed'].fillna('1970-01-01',inplace=True)
```

```
df['Date Last Confirmed'].unique()
```

```
array(['2023-02-13', '2022-12-13', '2022-06-14', '2022-10-11',
       '2023-03-08', '2022-07-12', '2022-04-06', '2023-01-10',
       '2022-08-10', '2022-05-05', '2021-12-09', '2022-09-14',
       '2022-03-07', '2021-06-07', '2020-02-06', '2022-01-10',
       '2023-03-09', '2021-10-12', '2021-11-04', '2022-11-10',
       '2021-04-08', '2021-05-06', '2021-07-14', '2021-01-14',
       '2021-08-04', '2017-06-02', '2021-09-10', '2021-03-15',
       '2020-06-09', '2022-02-10', '2020-07-13', '2020-08-05',
       '2020-12-03', '2020-10-09', '2020-11-09', '2022-10-22',
       '2021-02-15', '2021-02-22', '2020-03-03', '2023-01-26',
       '2020-05-06', '2020-09-03', '2020-04-08', '2021-11-02',
       '2022-04-13', '2020-10-07', '2021-12-13', '2022-12-12',
       '2018-05-07', '2022-07-08', '2023-01-12', '2022-01-11',
       '2022-03-04', '2019-04-09', '2018-06-11', '2023-02-09',
       '2020-04-10', '2022-11-09', '2022-06-10', '2022-08-09',
       '2021-08-03', '2020-01-06', '2019-01-14', '2017-09-06',
       '2018-03-29', '2019-10-10', '2019-09-11', '2019-11-08',
       '2021-04-12', '2022-10-10', '2021-09-09', '2020-01-30',
       '2019-12-09', '2020-03-05', '2020-08-10', '2019-06-06',
       '2018-11-12', '2018-02-06', '2022-10-03', '2017-10-03',
       '2018-03-08', '2018-01-03', '2019-07-10', '2021-07-15',
       '2017-05-11', '2021-02-12', '2019-08-09', '2017-12-05',
       '2022-10-06', '2019-03-11', '2020-06-08', '2021-01-07',
       '2018-08-07', '2020-05-15', '2023-03-05', '2020-11-03',
       '1970-01-01', '2019-05-07'], dtype=object)
```

概率分布

计算Owner Type Code这一特征各种取值的分布

```
df['Owner Type Code'].value_counts()
```

```
P      22119
LG      1095
T        409
SG       256
FG        39
J          8
Name: Owner Type Code, dtype: int64
```

概率分布


```
df['Owner Type Code'].value_counts()
```

```
P     22119
LG     1095
T       409
SG      256
FG       39
J         8
Name: Owner Type Co
```

计算Owner Type Code这一特征各种取值的分布，归一化后作为概率作为np.random.choice的参数

```
count = df.loc[:,'Owner Type Code'].isna().sum()
prob_distribution = df['Owner Type Code'].value_counts()
prob_distribution /=prob_distribution.sum()
random_fill = np.random.choice(prob_distribution.index,count,p=prob_distribution.values)
```

```
pd.value_counts(random_fill)
```

```
P     40843
LG     2134
T       707
SG      439
FG       75
J        12
dtype: int64
```

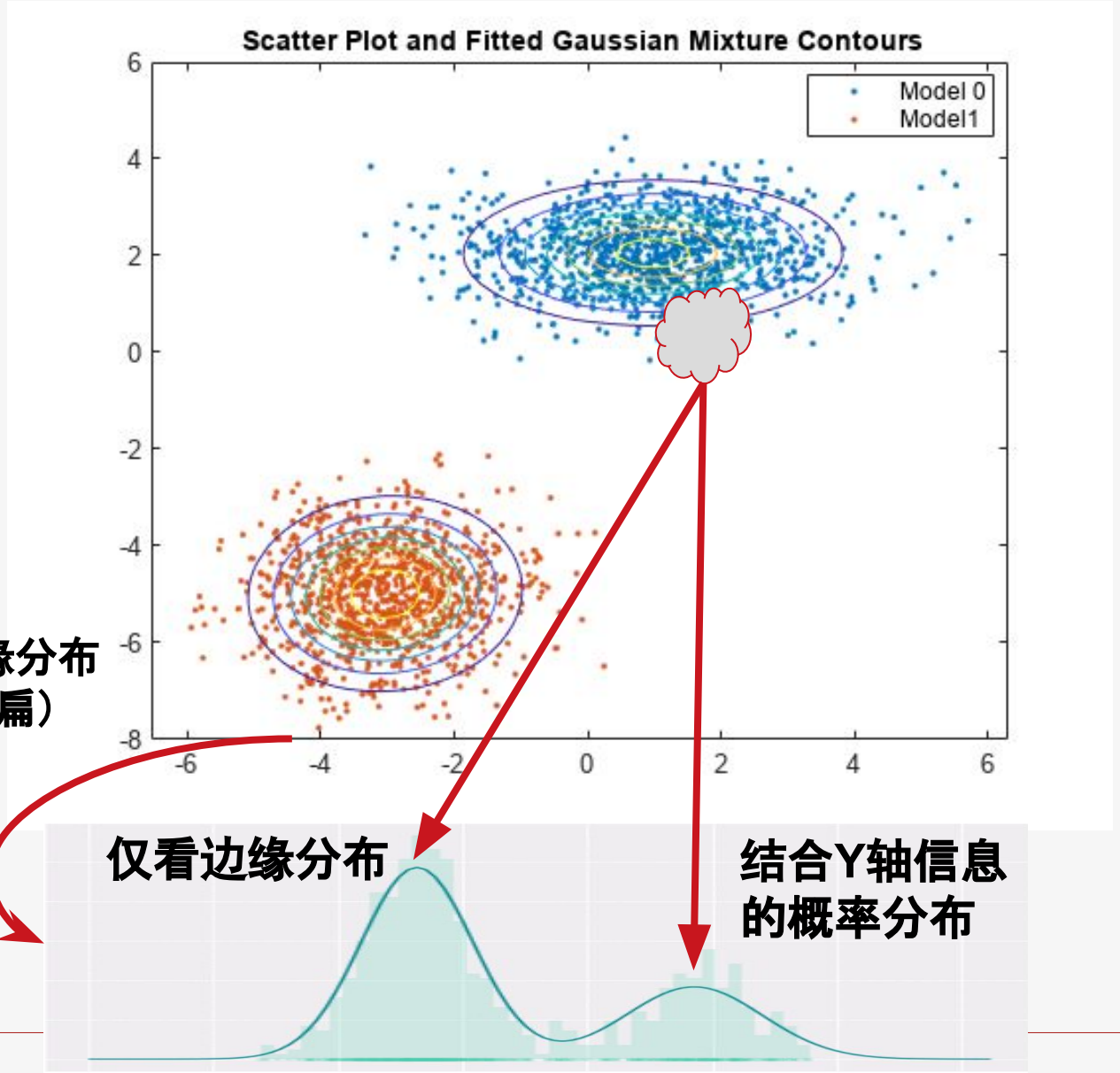
概率分布

将随机生成的结果填补到缺失值中

```
df.loc[df['Owner Type Code'].isna(),'Owner Type Code']=random_fill
df['Owner Type Code'].value_counts()
```

```
P     62962
LG     3229
T      1116
SG      695
FG      114
J        20
Name: Owner Type Code, dtype: int64
```

缺失值处理



- 在缺失比例较高的情况下，采取中位数或平均数会改变样本的整体分布，此时可以先拟合其他数据的概率分布，再根据分布填补缺失值；
- 仅拟合一维数据忽视了其他维度的信息，可以拟合多维数据，根据其他维度的信息来获得缺失值的概率分布；
- $P(x_{i,miss}|X_{i,exist},\theta)$

联合概率分布

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

使用KDE拟合经纬度分布

注意在处理经纬度坐标时，计算距离的方式与平面坐标轴不同，需要设置metric='haversine'，使用半正矢公式计算距离，且坐标需要先转换为弧度(radians)

```
from sklearn.neighbors import KernelDensity
X = np.radians(df.loc[:,["Longitude","Latitude"]].values)
kde = KernelDensity(kernel='gaussian',bandwidth = 0.005,metric='haversine').fit(X)
np.degrees(kde.sample(5))
```

```
array([[ -86.47078138,   35.97360782],
       [-117.33062154,   34.1433392 ],
       [ -79.25918291,   42.97154281],
       [ -89.59255141,   34.72624275],
       [ -80.10977834,   26.22530983]])
```

条件概率分布

给定一个经度坐标(-82.14)，计算该经度下，KDE计算出的纬度的概率密度函数(条件概率)；

注意kde.score_samples计算得到概率的对数；

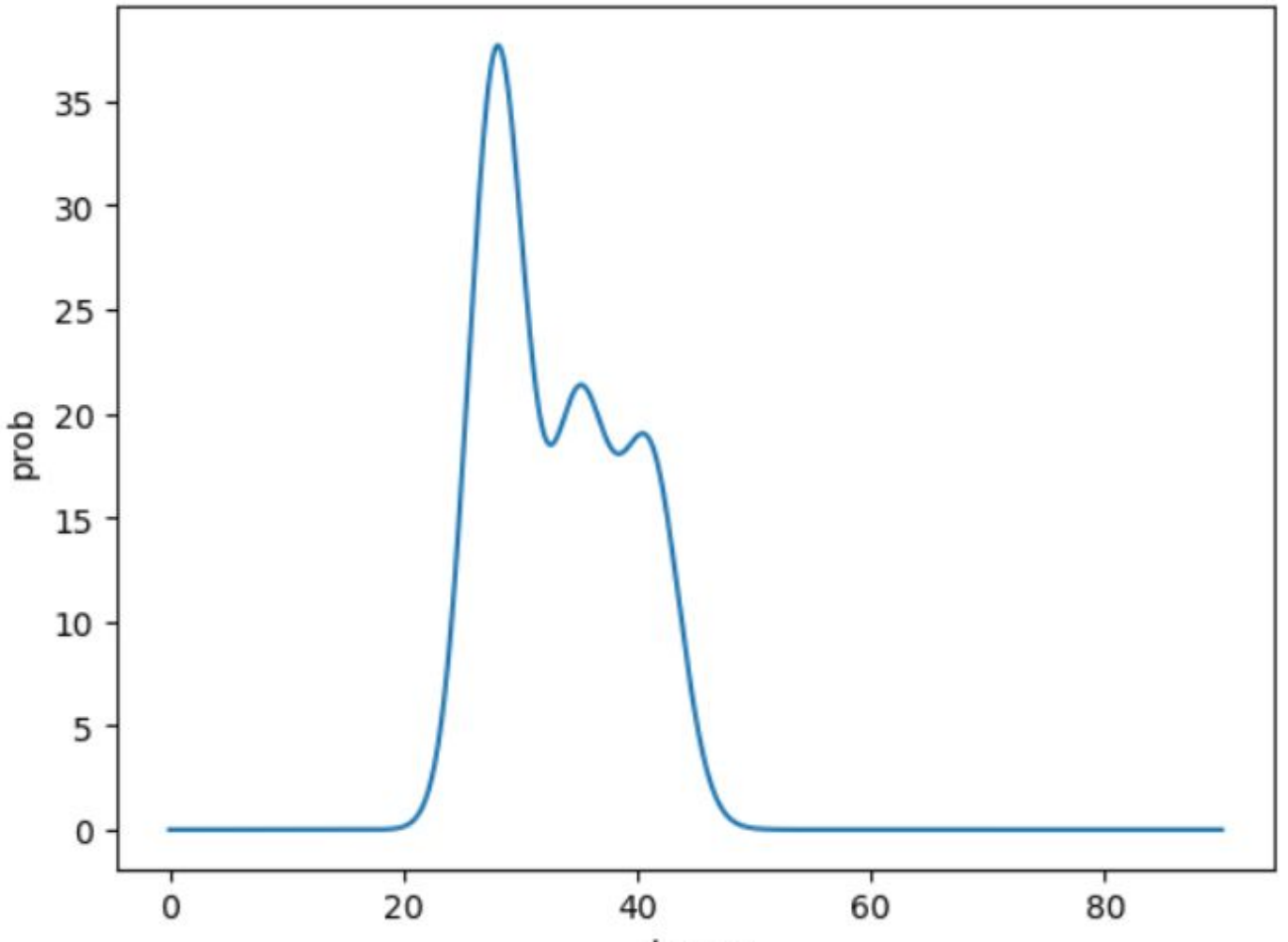
将概率归一化后采样纬度：

```
p/=sum(p)
np.degrees(np.random.choice(y,10,p=p))
```

```
array([25.22522523, 35.85585586, 30.36036036, 28.91891
       25.76576577, 28.28828829, 37.47747748, 26.21621
```

```
y = np.radians(np.linspace(0,90,1000))#从角度转为弧度0-pi/2
xy = np.vstack([np.radians([-82.14546425]*1000),y]).T#构建坐标
p=np.exp(kde.score_samples(xy))#计算log probability，计算指数
plt.plot(np.degrees(y),p)
plt.xlabel("degree")
plt.ylabel("prob")
```

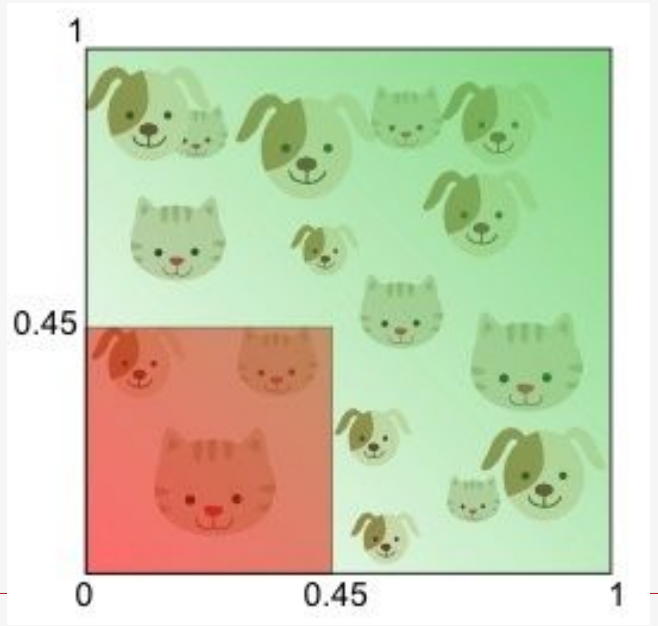
```
Text(0, 0.5, 'prob')
```

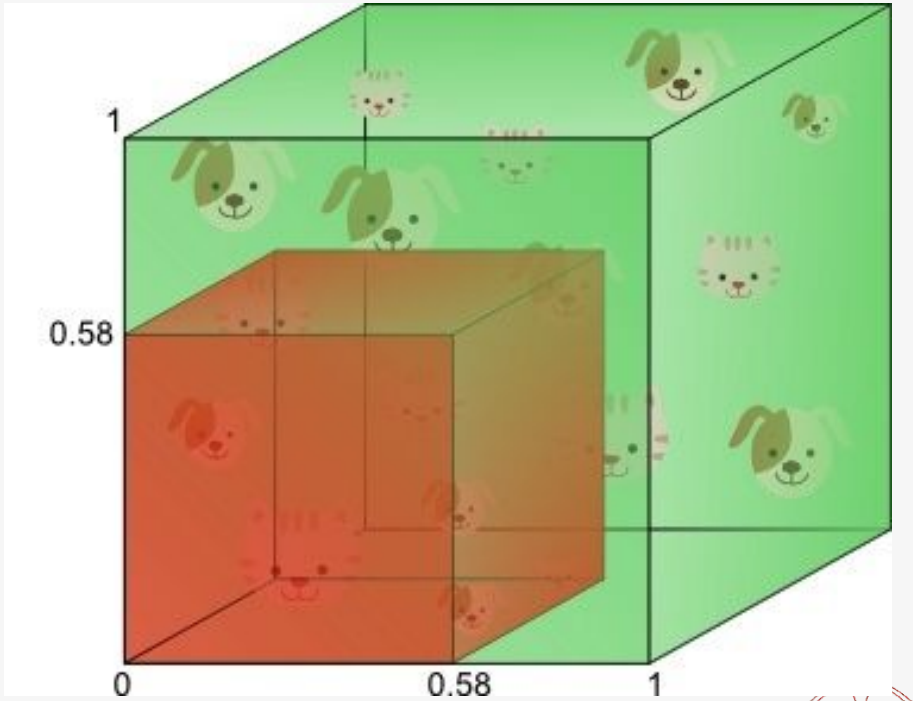


数据降维

★ 为什么要降维

- 高维空间中的数据分布和低维空间的直觉不符
  - 随着特征维度增高，样本在空间中的密度**指数式下降——维度灾难**
  - **维度高导致样本分布稀疏，从而导致模型容易过拟合**





数据降维

★ 为什么要降维

- 数据的不同维度间可能存在共线性
- 模型参数的方差增大，可靠性降低

$$y = x_1 + x_2 \qquad (A)$$
$$x_1 = 2x_2 \qquad (B)$$

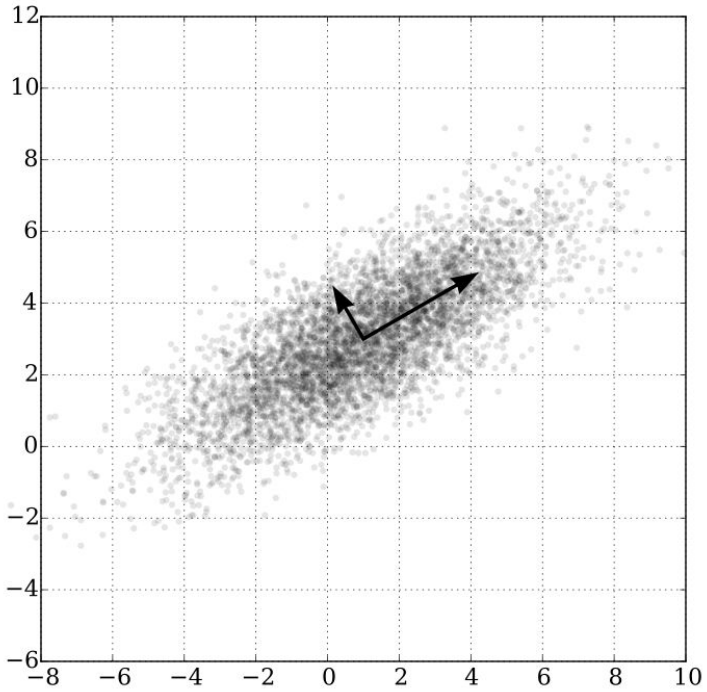
$$y = 1.0x_1 + 1.0x_2 \qquad y = 0.0x_1 + 3.0x_2$$
$$y = 1.5x_1 + 0.0x_2 \qquad y = 0.5x_1 + 2.0x_2$$
$$y = 0.1x_1 + 2.8x_2 \qquad \ldots\ldots$$

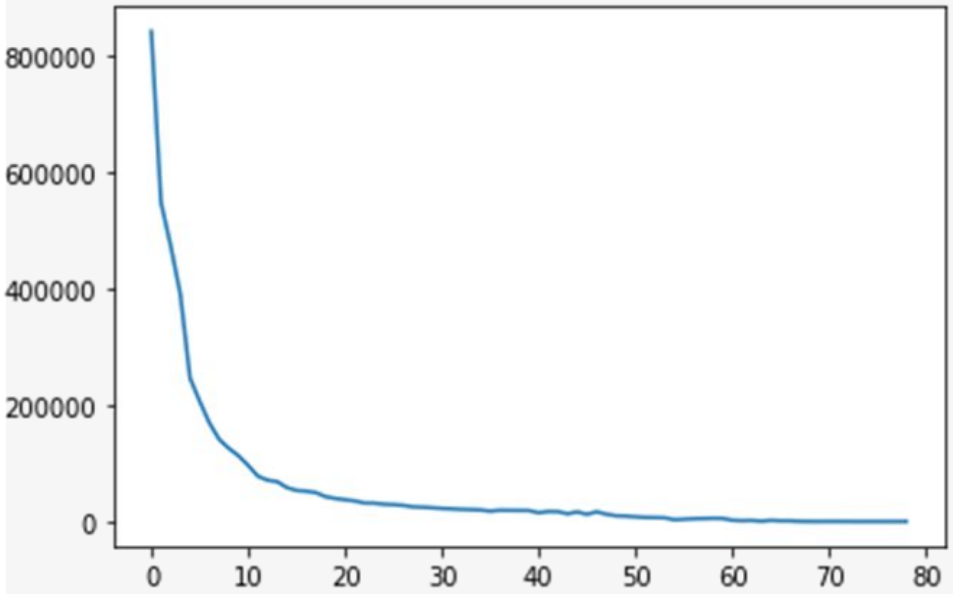
共线性的极端示例

数据降维

★ PCA, Principal component analysis, 主成分分析

- 将数据投影到**方差最大**的方向，即主成分
- 认为N维数据可以近似由m个主成分**线性表示**，m<<N
- 对数据去中心化后，计算协方差矩阵的特征分解，或计算数据矩阵的奇异值分解(等价)





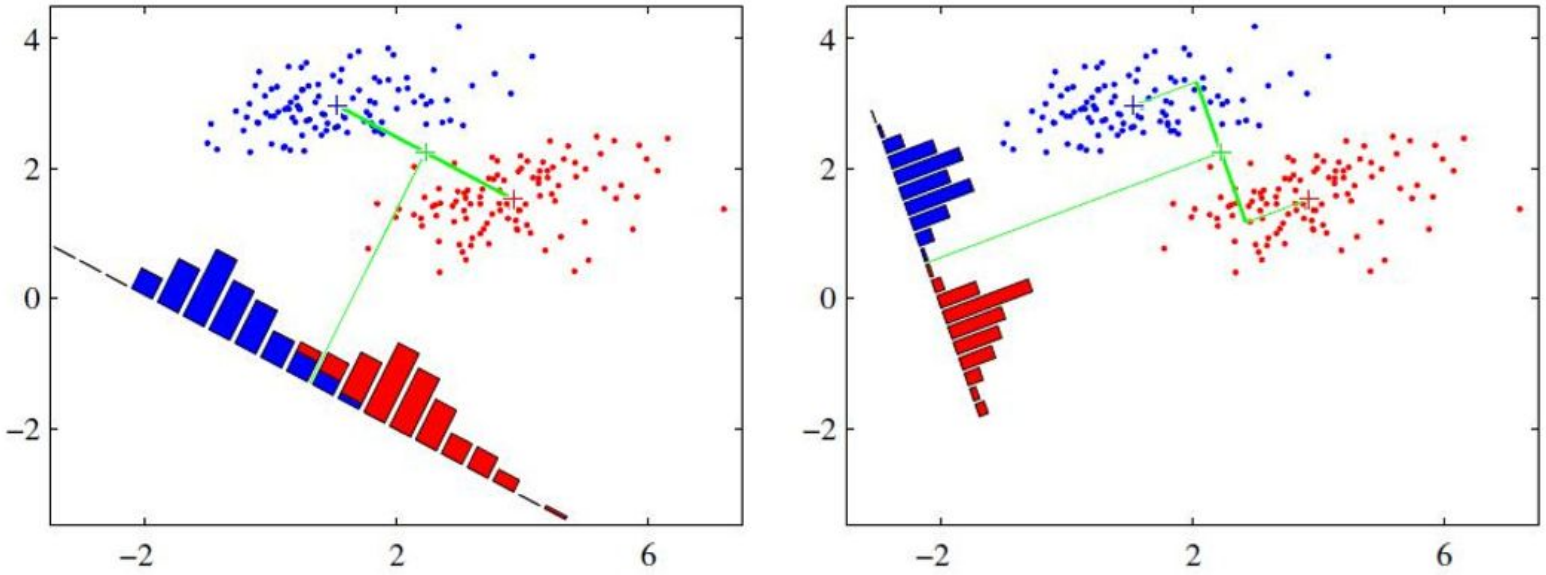
协方差矩阵特征值从大到小排序

左图表示，矩阵的信息量由少数特征值和特征向量提供，因此保留前m个最大的特征值和对应的特征向量，可以实现降维，且不同分量不相关。

数据降维

★ LDA, Linear Discriminant Analysis, 线性判别分析

- 是一种有监督的降维方法
- 根据类别标签，选择**最能区分数据**的平面，使得投影后**类内方差最小，类间方差最大**
- 下图中红蓝表示两类数据标签，右图选择的投影方向使得两类数据**更好地被区分**



Ability to Classify

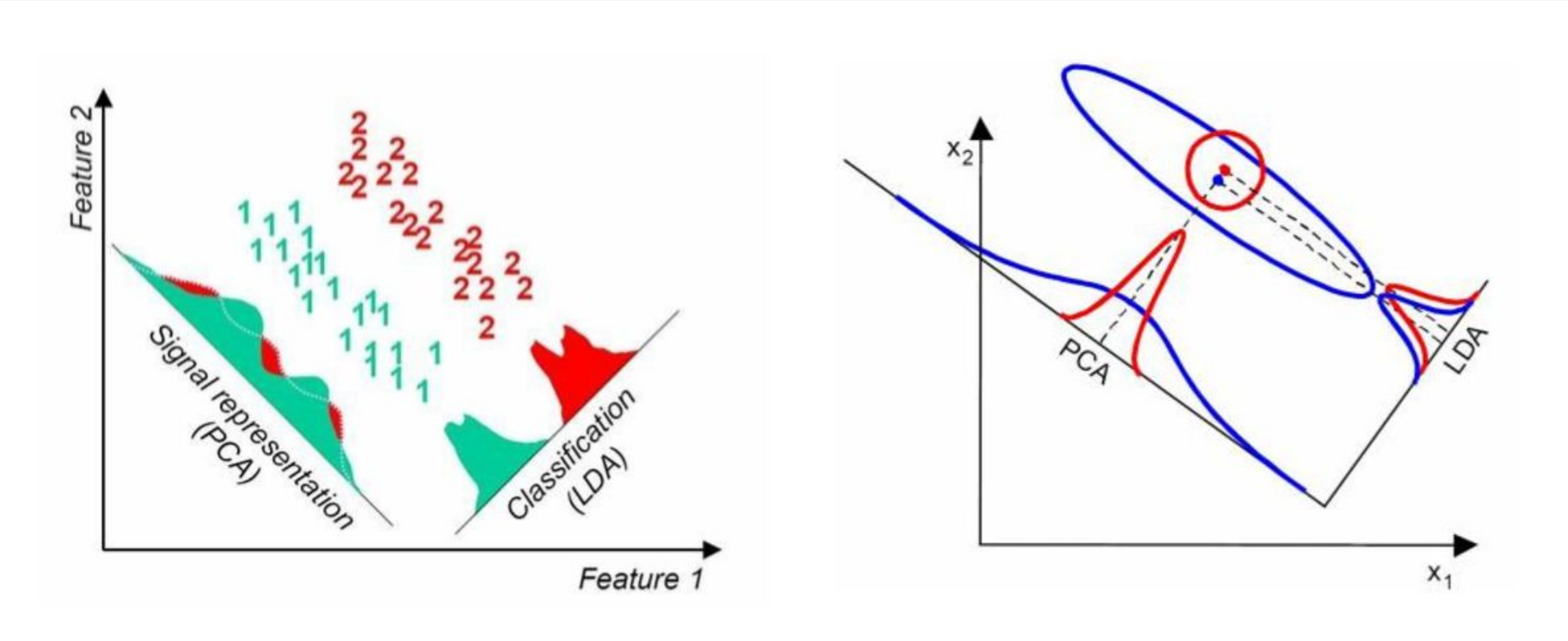
$$\Pr(G=k|X=x)=\frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K}f_\ell(x)\pi_\ell}.$$

Bayes Theorem

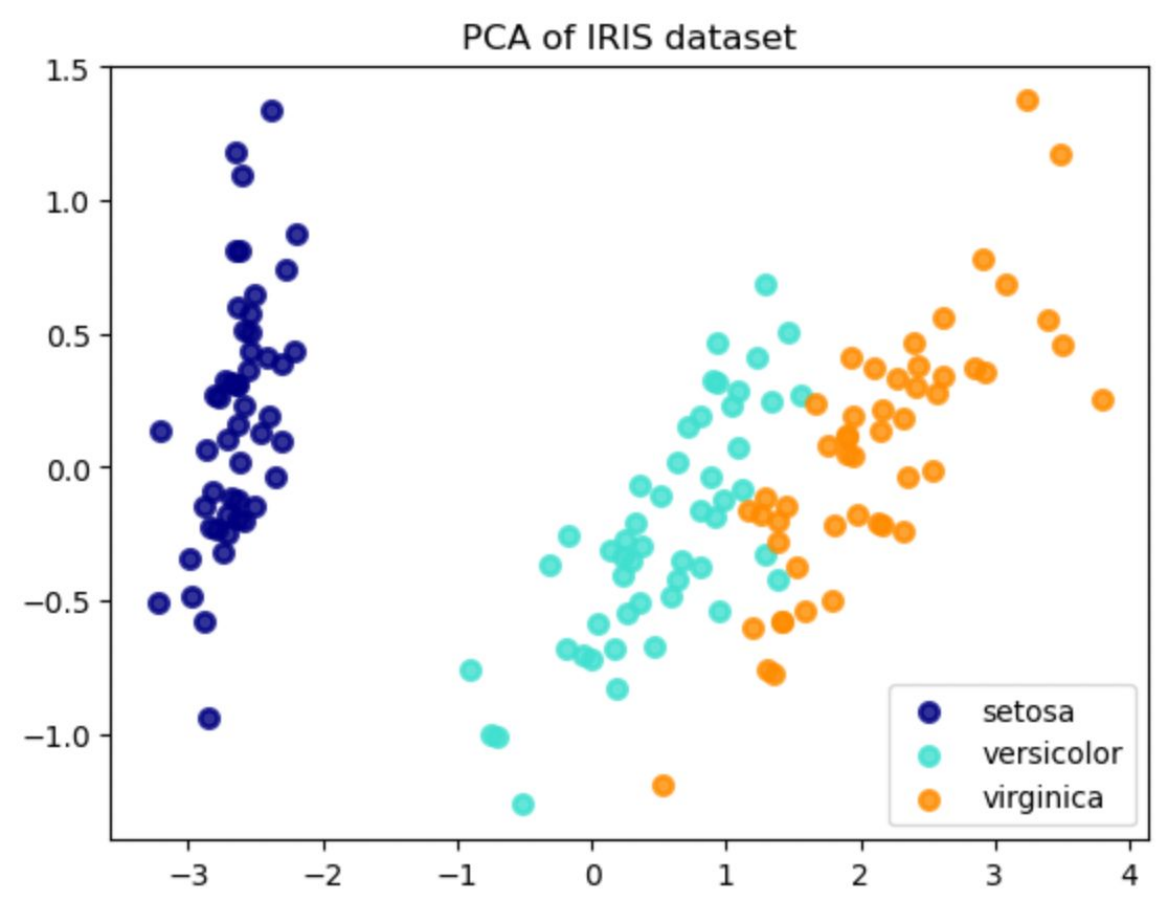
数据降维

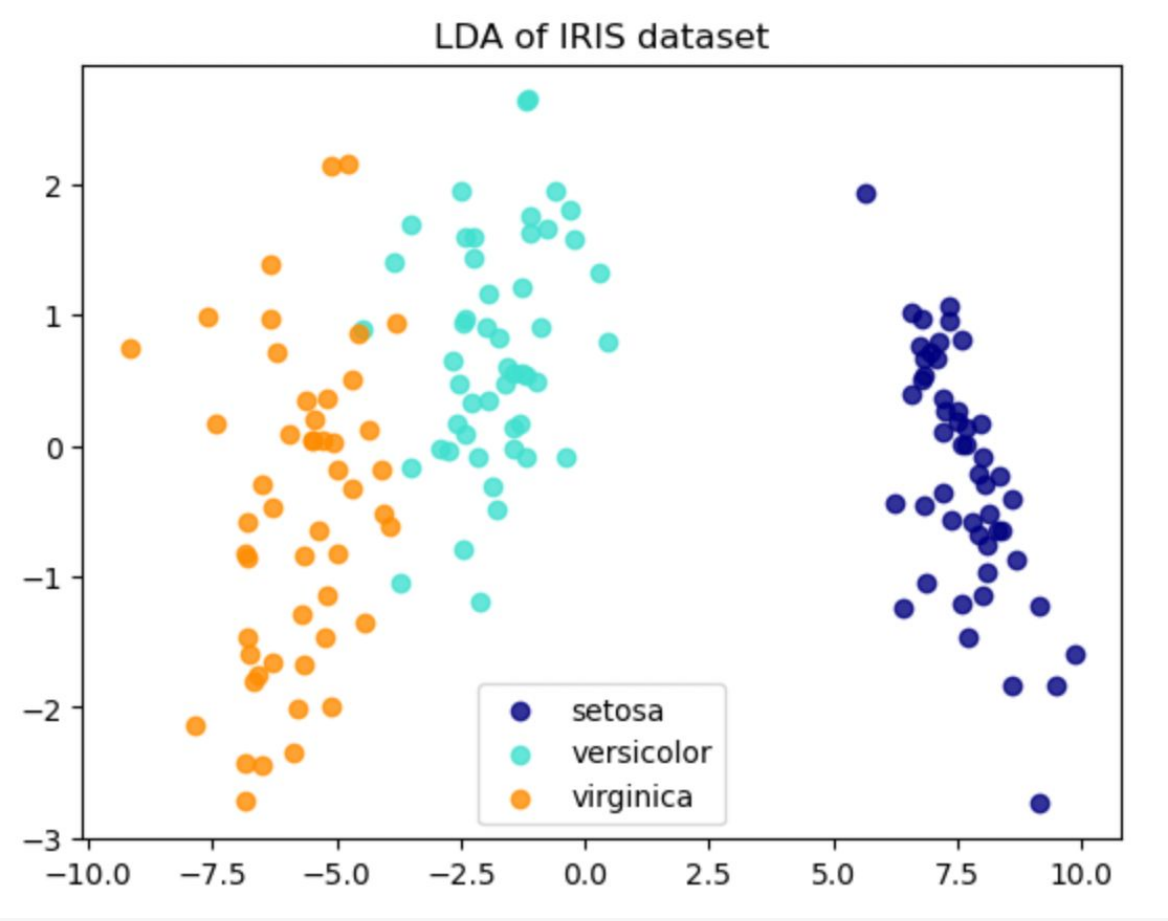
★ LDA和PCA的区别

- PCA是无监督降维算法，LDA是有监督降维和分类算法，现实中标签并不容易获得
- PCA可以保证每个分量线性无关，而LDA选择分类能力最大的投影方向
- LDA最高只能降维至类别数量k-1的维度，PCA没有这个限制



PCA vs. LDA





基于scikit-learn 的代码实现：

http://10.123.4.32:8887/notebooks/example/PCA%20and%20LDA.ipynb

时间序列数据

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

- 通常也采用表格的形式，但在行维度按照时间顺序递增；

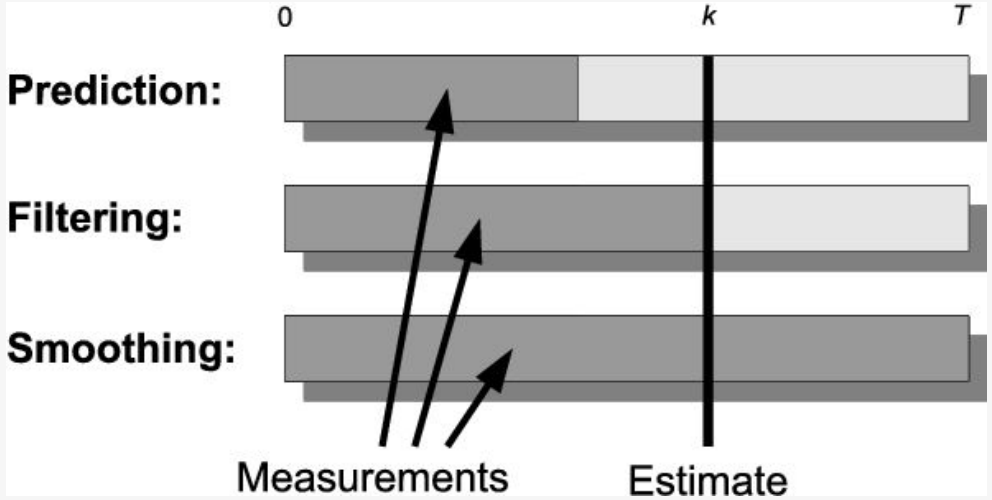
Date	Ozone ($\mu g/m^3$)	Temperature (°C)	Relative humidity (%)	*n* deaths
1 Jan 2002	4.59	−0.2	75.7	199
2 Jan 2002	4.88	0.1	77.5	231
3 Jan 2002	4.71	0.9	81.3	210
4 Jan 2002	4.14	0.5	85.4	203
5 Jan 2002	2.01	4.3	93.5	224
6 Jan 2002	2.4	7.1	96.4	198
7 Jan 2002	4.08	5.2	93.5	180
8 Jan 2002	3.13	3.5	81.5	188
9 Jan 2002	2.05	3.2	88.3	168
10 Jan 2002	5.19	5.3	85.4	194
11 Jan 2002	3.59	3.0	92.6	223
12 Jan 2002	12.87	4.8	94.2	201

社会科学往往关注时间颗粒度大，但是跨度长的数据！

SJTU

时间序列数据的分析

- 数据性质
  - 数据的时间维度特性：季节性、平稳性、自相关
  - 时间序列分解
  - 数据概率分布随时间的变化
- 对不同时刻数据的处理
  - 平滑：对0-T时刻中历史数据的估计；
  - 滤波：根据0-T时刻的历史数据，对T时刻状态进行估计；
  - 预测：根据0-T时刻的历史数据，对T+k时刻的数据进行估计；
- 时间序列相似性
  - DTW、Correlation



读取时间序列数据

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

空气质量数据

```
ts = pd.read_csv("AirQualityUCI.csv")
```

```
print(ts.columns)
```

```
Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)',
       'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)',
       'PT08.S5(O3)', 'T', 'RH', 'AH'],
      dtype='object')
```

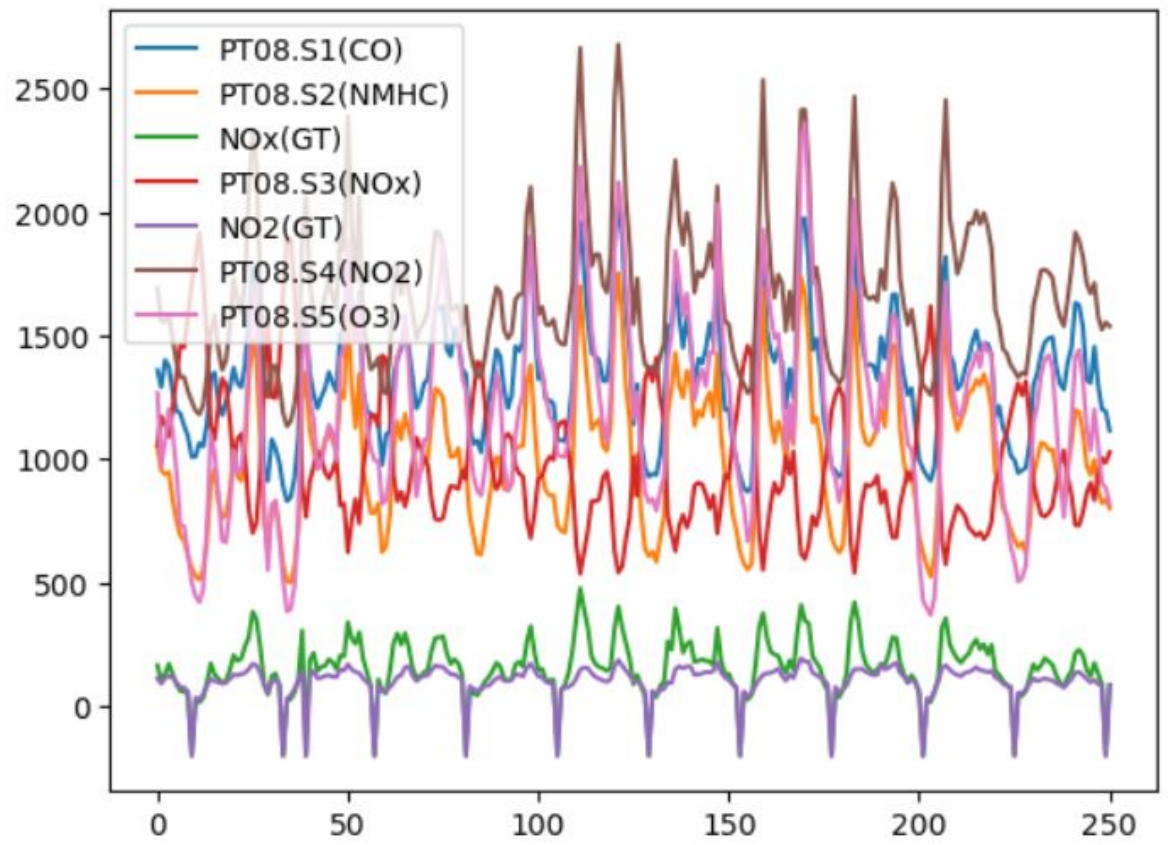
```
ts.head(3)
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH
0	2004-03-10	18:00:00	2.6	1360.00	150	11.881723	1045.50	166.0	1056.25	113.0	1692.00	1267.50	13.6	48.875001
1	2004-03-10	19:00:00	2.0	1292.25	112	9.397165	954.75	103.0	1173.75	92.0	1558.75	972.25	13.3	47.700000
2	2004-03-10	20:00:00	2.2	1402.00	88	8.997817	939.25	131.0	1140.00	114.0	1554.50	1074.00	11.9	53.975000

观察时间序列数据

使用loc方法对数据切片后，用plot方法绘制时序曲线，横轴为时间

```
: ts.loc[:250,['PT08.S1(CO)', 'PT08.S2(NMHC)','NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)',
       'PT08.S5(O3)']].plot()
: <Axes: >
```



时间序列相似性

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

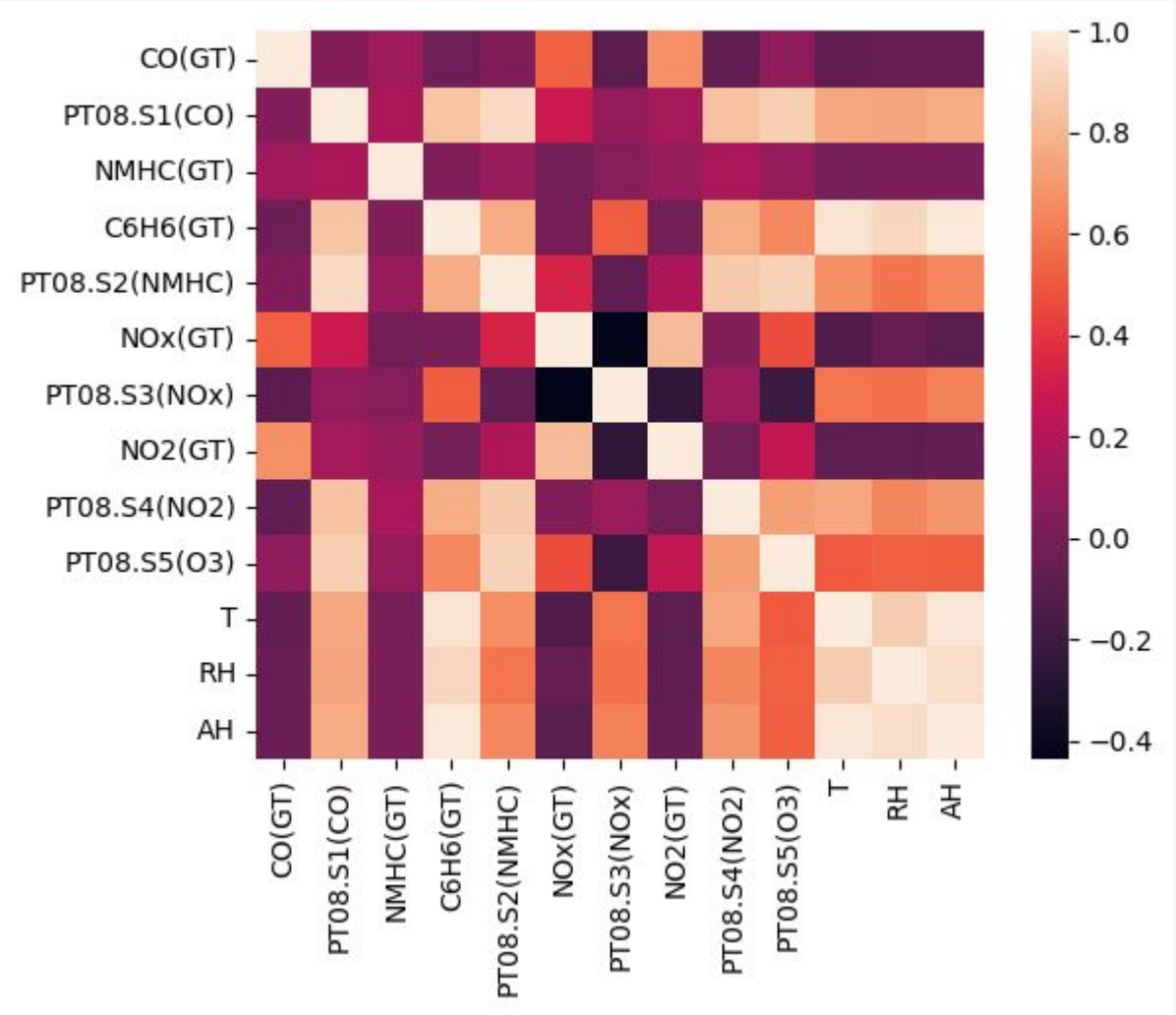
```
ts.corr(numeric_only=True)
```

	GDP(100M)	GDP-YOYGrowth(%)	PrimaryIndustry(100M)	PrimaryIndustry-YOYGrowth(%)	SecondaryIndustry(100M)	SecondaryIndustry-YOYGrowth(%)	TertiaryIndustry(100
GDP(100M)	1.000000	-0.145164	0.972694	0.244734	0.995647	-0.120284	0.9955
GDP-YOYGrowth(%)	-0.145164	1.000000	-0.116006	0.753765	-0.120036	0.969692	-0.1662
PrimaryIndustry(100M)	0.972694	-0.116006	1.000000	0.211030	0.982313	-0.101938	0.9501
PrimaryIndustry-YOYGrowth(%)	0.244734	0.753765	0.211030	1.000000	0.261949	0.816309	0.2342
SecondaryIndustry(100M)	0.995647	-0.120036	0.982313	0.261949	1.000000	-0.096940	0.9827
SecondaryIndustry-YOYGrowth(%)	-0.120284	0.969692	-0.101938	0.816309	-0.096940	1.000000	-0.1387
TertiaryIndustry(100M)	0.995520	-0.166289	0.950107	0.234226	0.982729	-0.138777	1.0000
TertiaryIndustry-YOYGrowth(%)	-0.159892	0.966861	-0.116668	0.628012	-0.134006	0.876893	-0.1835
GDP(100M)-Seasonal	0.736723	-0.217244	0.589720	0.301512	0.691351	-0.121211	0.7833
PrimaryIndustry(100M)-Seasonal	0.992238	-0.160886	0.978983	0.243397	0.991865	-0.121831	0.9828
SecondaryIndustry(100M)-Seasonal	0.778963	-0.171955	0.640329	0.377690	0.745313	-0.073619	0.8152
TertiaryIndustry(100M)-Seasonal	0.587365	-0.237784	0.420661	0.236829	0.529283	-0.140230	0.6480

时间序列相似性

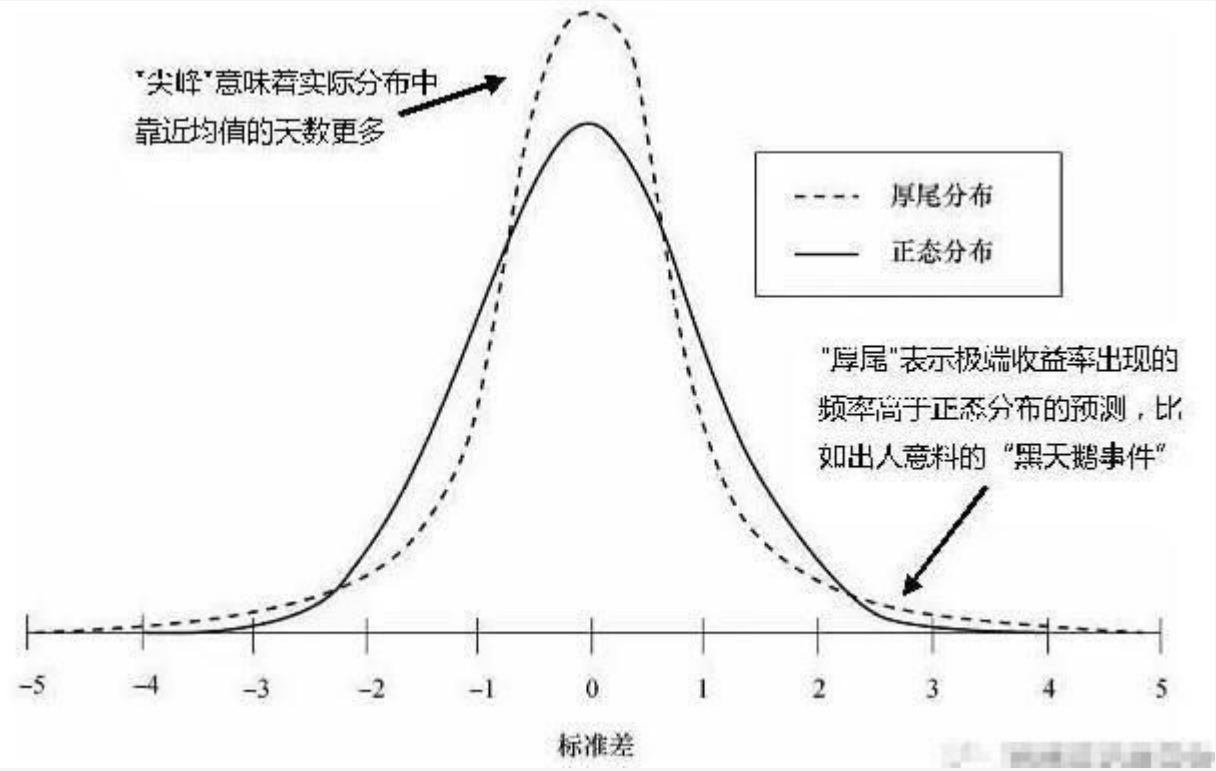
seaborn库是对matplotlib库的封装，往往可以更方便地绘制更好看的图像

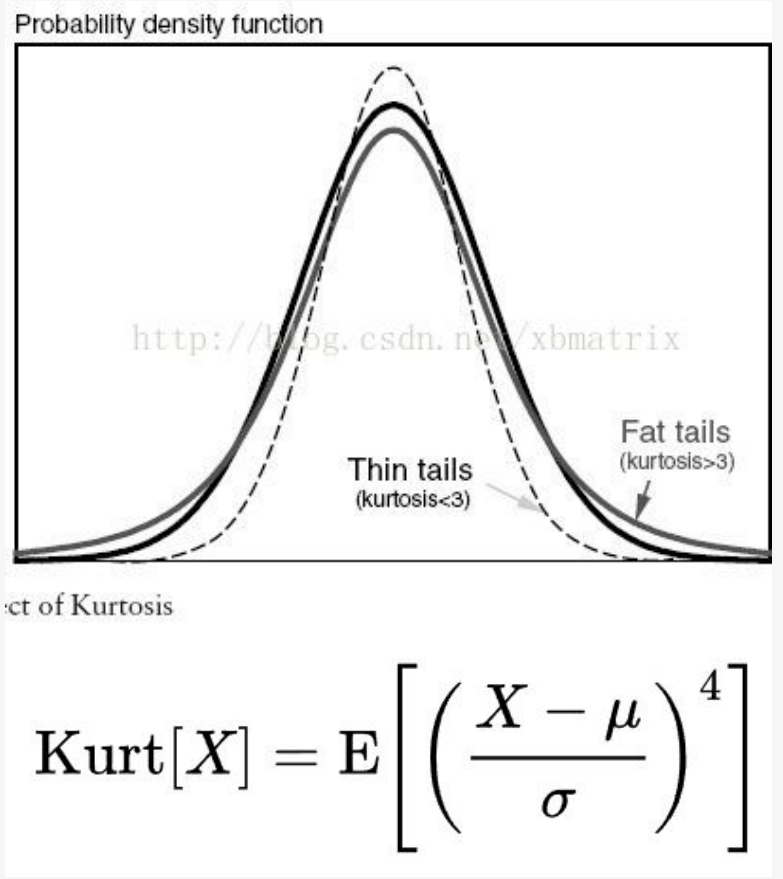
```
import seaborn as sns
sns.heatmap(data=ts.corr(numeric_only=True),square=True)
```



峰度kurtosis

峰度(kurtosis)，表征概率密度分布曲线在平均值处峰值高低的特征数。直观看来，峰度反映了峰部的尖度。正态分布峰度为3，小于3瘦尾，大于3厚尾。





ct of Kurtosis

$$\mathrm{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$

https://towardsdatascience.com/skewness-kurtosis-simplified-1338e094fc85

异常数据分析

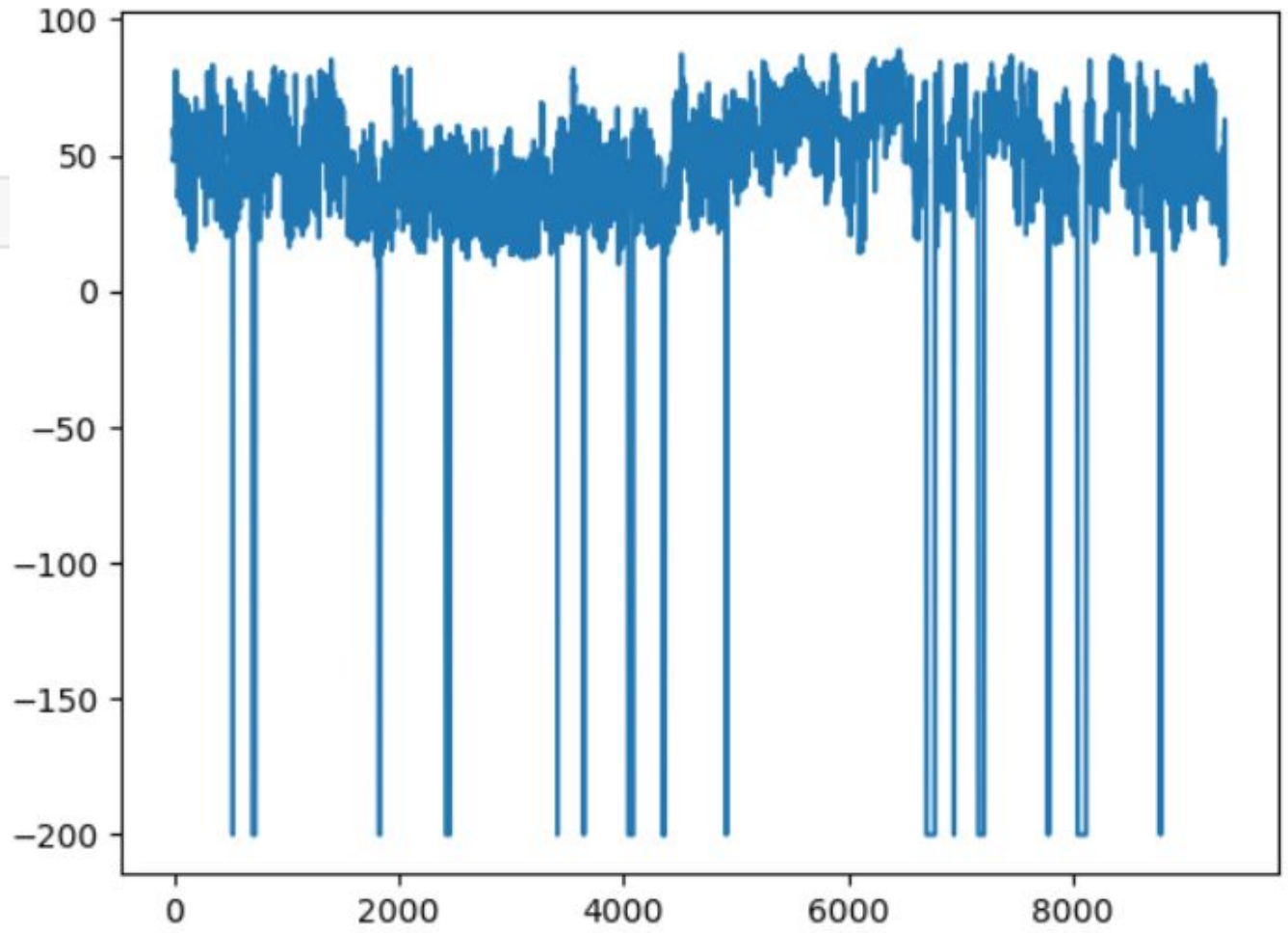
利用峰度检测异常值，以RH为例

```
ts.kurtosis(numeric_only=True)#kurtosis of normal == 0.0
```

```
CO(GT)            0.778306
PT08.S1(CO)       5.835512
NMHC(GT)         18.863824
C6H6(GT)         19.188714
PT08.S2(NMHC)     2.369425
NOx(GT)           1.505566
PT08.S3(NOx)      3.104215
NO2(GT)           0.275717
PT08.S4(NO2)      3.266388
PT08.S5(O3)       0.637957
T                18.774475
RH               15.764326
AH               20.613092
dtype: float64
```

```
ts.loc[:,'RH'].plot()
#可以发现某些时刻出现约为-200的异常值，一般是机械或程序故障时显式标志出的异常值
```

```
<Axes: >
```

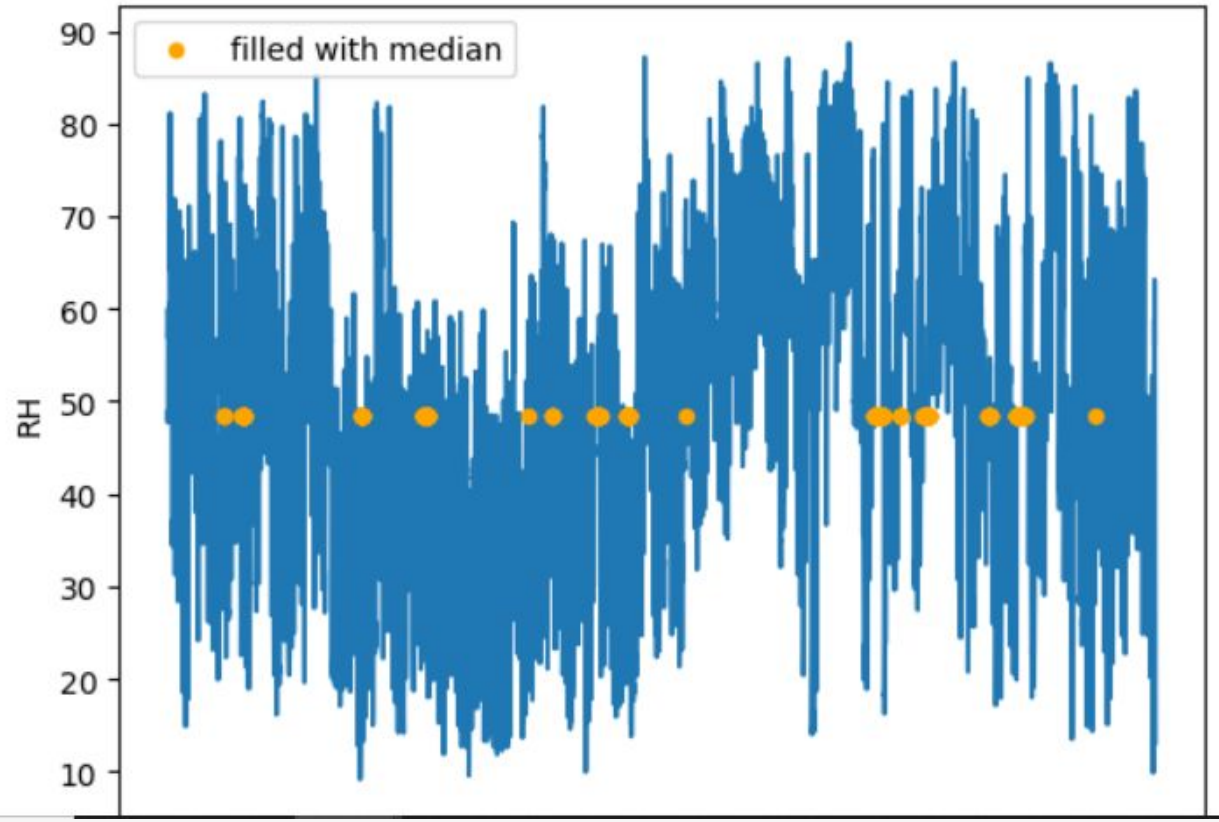


少量数据用中位数填充

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
temp = ts.loc[:,"RH"].copy()
temp.loc[ts.loc[:,'RH']<-100]= ts.loc[:,"RH"].median()#采用所有数据中的中位数填充
fig,ax = plt.subplots()
temp.plot(ax=ax,zorder=0)
temp.loc[ts.loc[:,'RH']<-100].reset_index().plot(x='index',y='RH',kind='scatter',ax=ax,color='orange',zorder=1,label='filled with median')
```

```
<Axes: xlabel='index', ylabel='RH'>
```

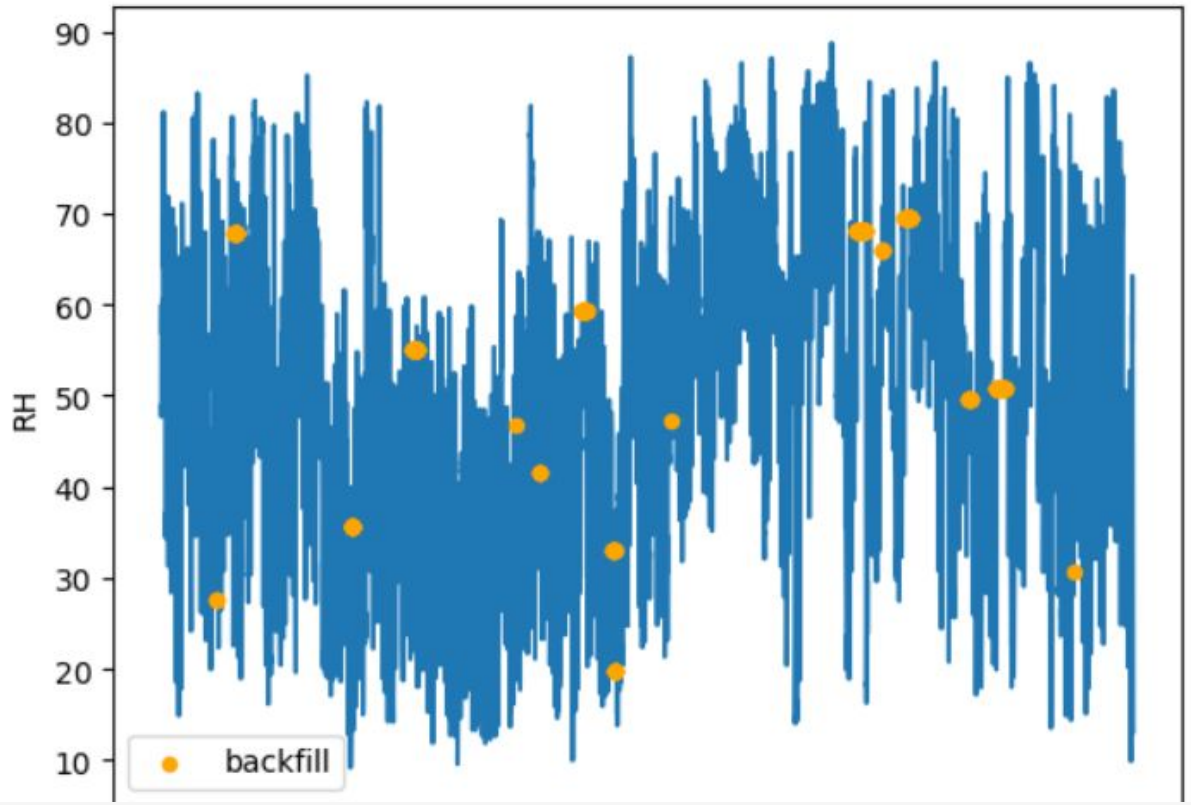


时间序列，可用相邻数据填充

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
temp = ts.loc[:,"RH"].copy()
temp.loc[ts.loc[:,'RH']<-100]=np.nan
temp.fillna(method='backfill',inplace=True)
fig,ax = plt.subplots()
temp.plot(ax=ax,zorder=0)
temp.loc[ts.loc[:,'RH']<-100].reset_index().plot(x='index',y='RH',kind='scatter',ax=ax,color='orange',zorder=1,label='backfill')
```

```
<Axes: xlabel='index', ylabel='RH'>
```



读取季度GDP数据

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

```
import seaborn as sns
import numpy as np
```

```
ts = pd.read_csv("GDP.csv",index_col=0)
```

```
print(ts.columns)
```

```
Index(['Time', 'GDP(100M)', 'GDP-YOYGrowth(%)', 'PrimaryIndustry(100M)',
       'PrimaryIndustry-YOYGrowth(%)', 'SecondaryIndustry(100M)',
       'SecondaryIndustry-YOYGrowth(%)', 'TertiaryIndustry(100M)',
       'TertiaryIndustry-YOYGrowth(%)'],
      dtype='object')
```

```
ts.head(3)
```

Index	Time	GDP(100M)	GDP-YOYGrowth(%)	PrimaryIndustry(100M)	PrimaryIndustry-YOYGrowth(%)	SecondaryIndustry(100M)	SecondaryIndustry-YOYGrowth(%)	TertiaryIndustry(100M)	TertiaryIn YOYGro
1	2012第1季度	117357.6	8.1	6446.0	3.7	52316.2	9.5	58595.5	
2	2012第1-2季度	248678.3	7.9	16357.3	4.3	113749.4	8.7	118571.6	
3	2012第1-	386767.9	7.8	31013.4	4.2	176005.6	8.3	179749.0	

季度GDP数据

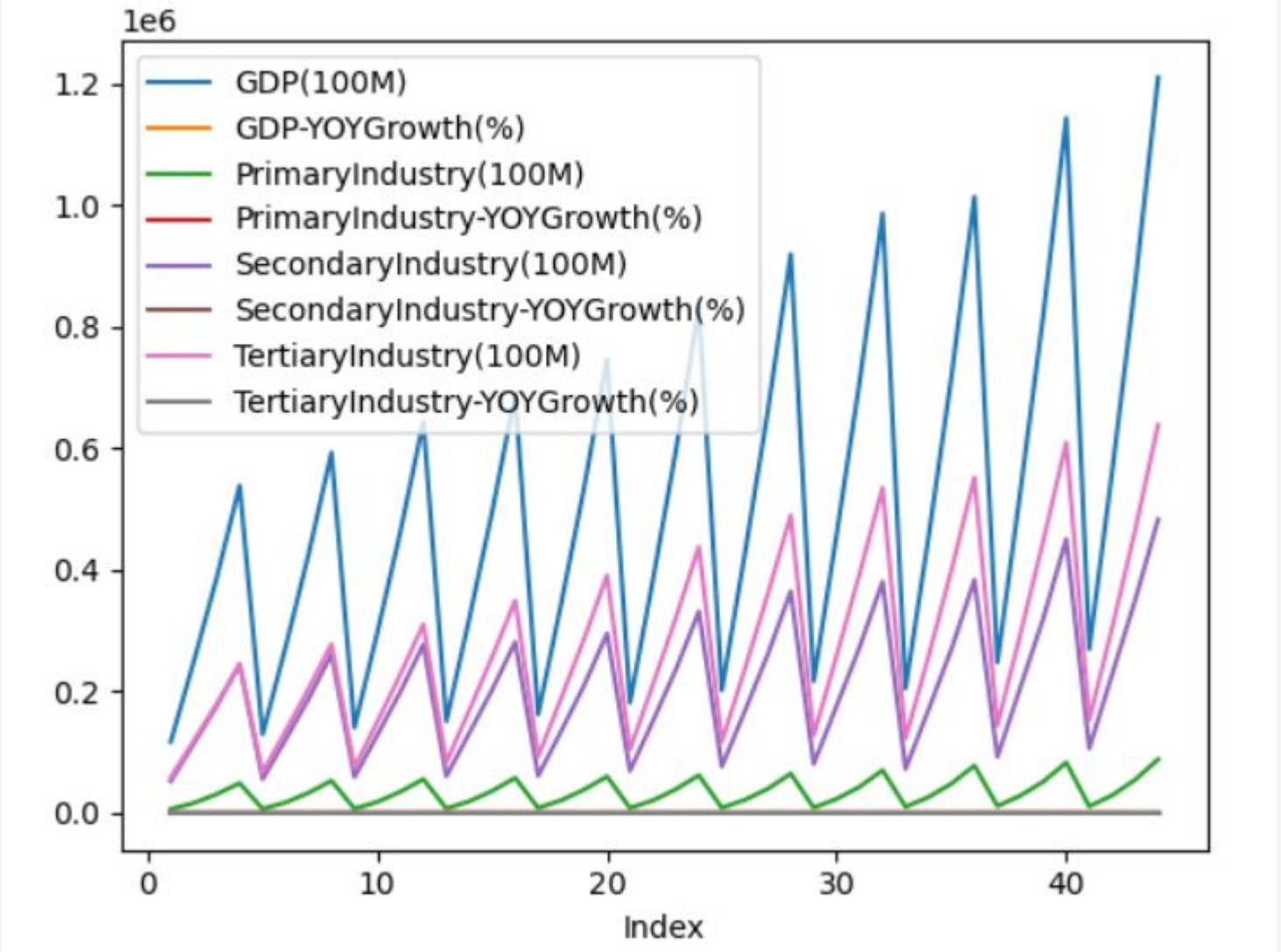
包括每季度发布的**本年内GDP数值**及同比增长率；

1	2012第1季度
2	2012第1-2季度
3	2012第1-3季度
4	2012第1-4季度
5	2013第1季度
6	2013第1-2季度
7	2013第1-3季度
8	2013第1-4季度
9	2014第1季度
10	2014第1-2季度
11	2014第1-3季度
12	2014第1-4季度
13	2015第1季度
14	2015第1-2季度
15	2015第1-3季度
16	2015第1-4季度

P:农业
S:工业
T:服务业

```
ts.plot()
```

```
<Axes: xlabel='Index'>
```



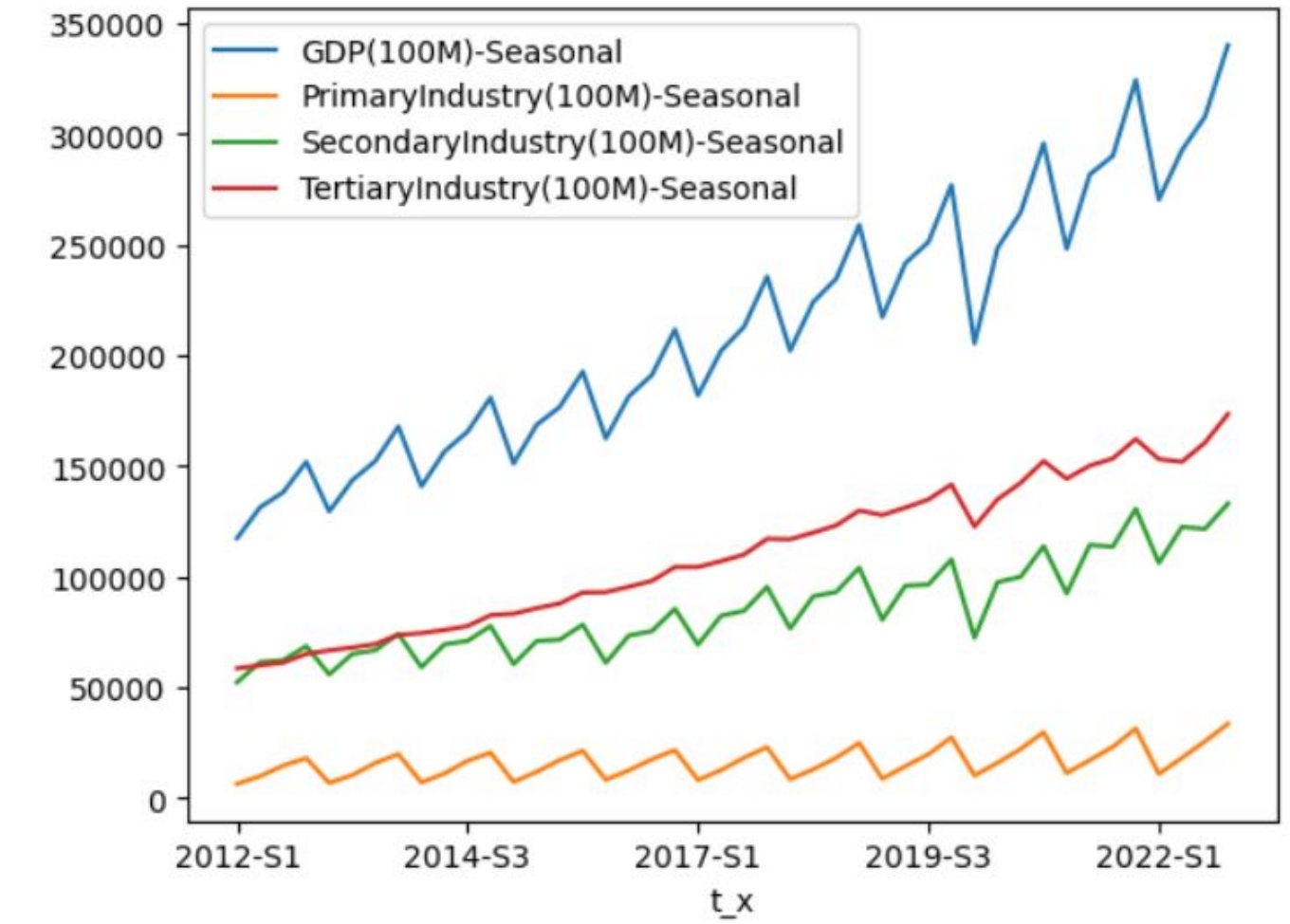
季度GDP数据

在每年内进行差分，可以得到每季度GDP，代码中将新增的列名添加Seasonal后缀；

P:农业
S:工业
T:服务业

```
ts.plot(x='t_x',y=[item for item in ts.columns if 'Seasonal' in item])
```

```
<Axes: xlabel='t_x'>
```



发现异常值

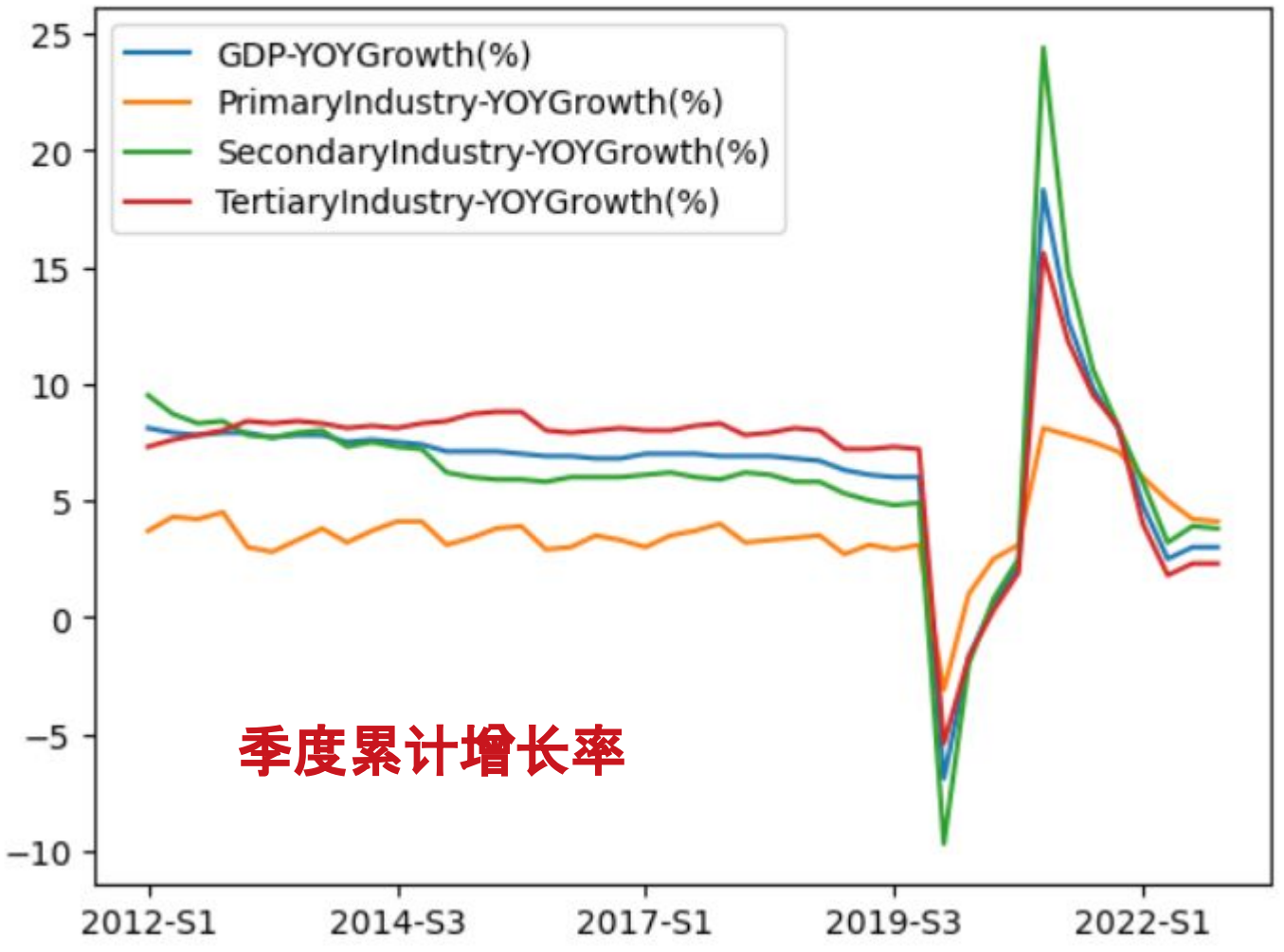
发现是由2020年初疫情导致的负增长率，不做处理

```
ts.kurtosis(numeric_only=True)#kurtosis of normal == 0.0
```

```
GDP(100M)                              -0.328819
GDP-YOYGrowth(%)                        6.281786
PrimaryIndustry(100M)                  -0.557533
PrimaryIndustry-YOYGrowth(%)            5.769640
SecondaryIndustry(100M)                -0.379549
SecondaryIndustry-YOYGrowth(%)          9.107084
TertiaryIndustry(100M)                 -0.319710
TertiaryIndustry-YOYGrowth(%)           3.931020
GDP(100M)-Seasonal                     -0.839283
PrimaryIndustry(100M)-Seasonal         -0.274121
SecondaryIndustry(100M)-Seasonal       -0.589181
TertiaryIndustry(100M)-Seasonal        -1.232704
dtype: float64
```

```
temp = ts.kurtosis(numeric_only=True)

ts.loc[:,list(temp[temp>3].index)+['t_x']].plot(x='t_x')
#国内生产总值季度累计同比增长（%）
#也就是今年的1-2季度累积GDP相比去年的1-2季度累积GDP增长多少
```

```
<Axes: xlabel='t_x'>
```

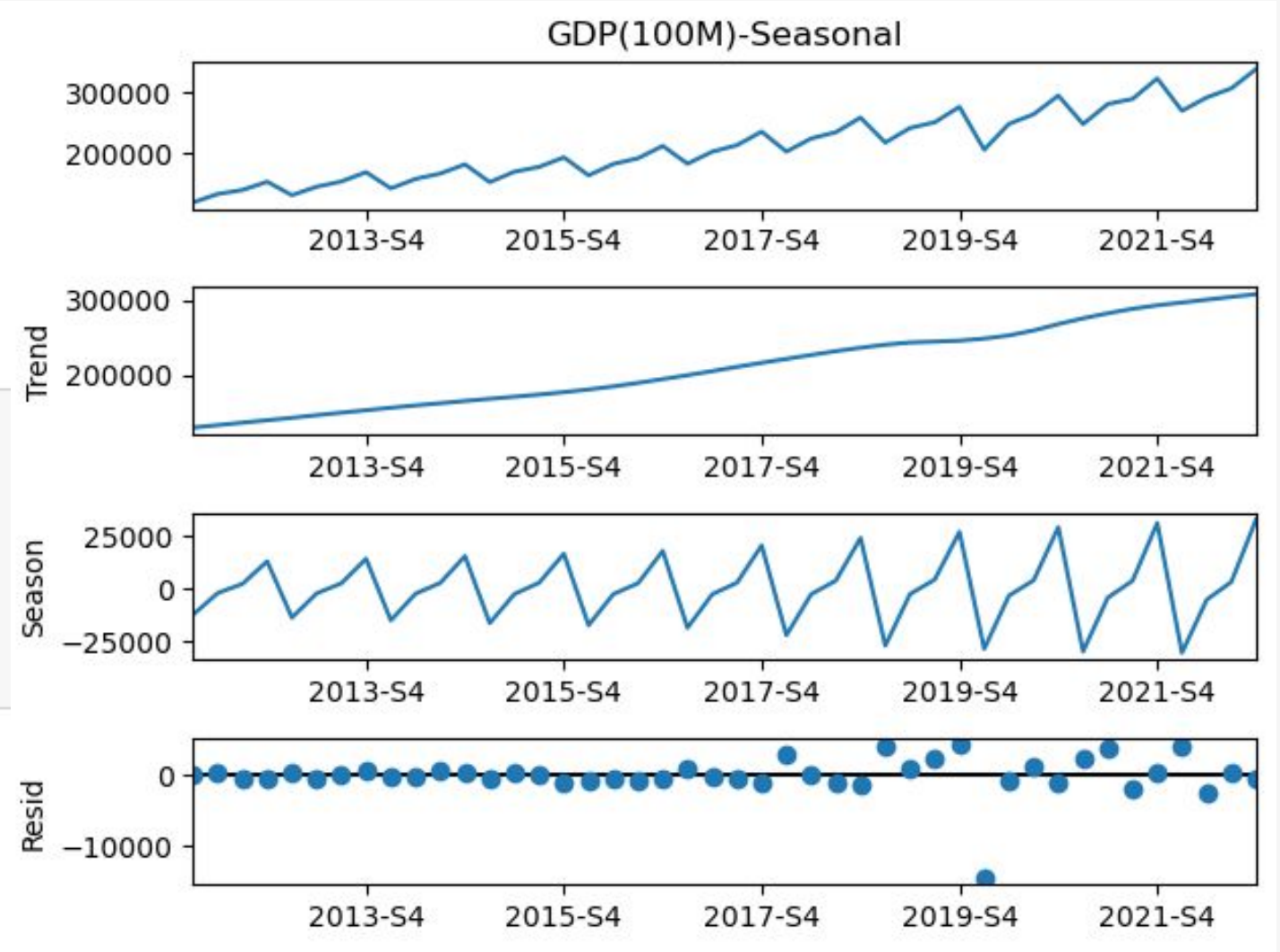


时间序列分解

使用STL模型对每季度GDP数据进行分解，分解出趋势项和季节项（周期为4）

```
from statsmodels.tsa.seasonal import STL
data = ts.loc[:,"GDP(100M)-Seasonal"]
res = STL(data,period = 4).fit()
f = res.plot()
for ax in f.get_axes():
    ax.set_xticks(res.observed.index,ts['t_x'])
    ax.xaxis.set_major_locator(ticker.MultipleLocator(base=8))
plt.show()
```

分解每季度GDP



时间序列分解

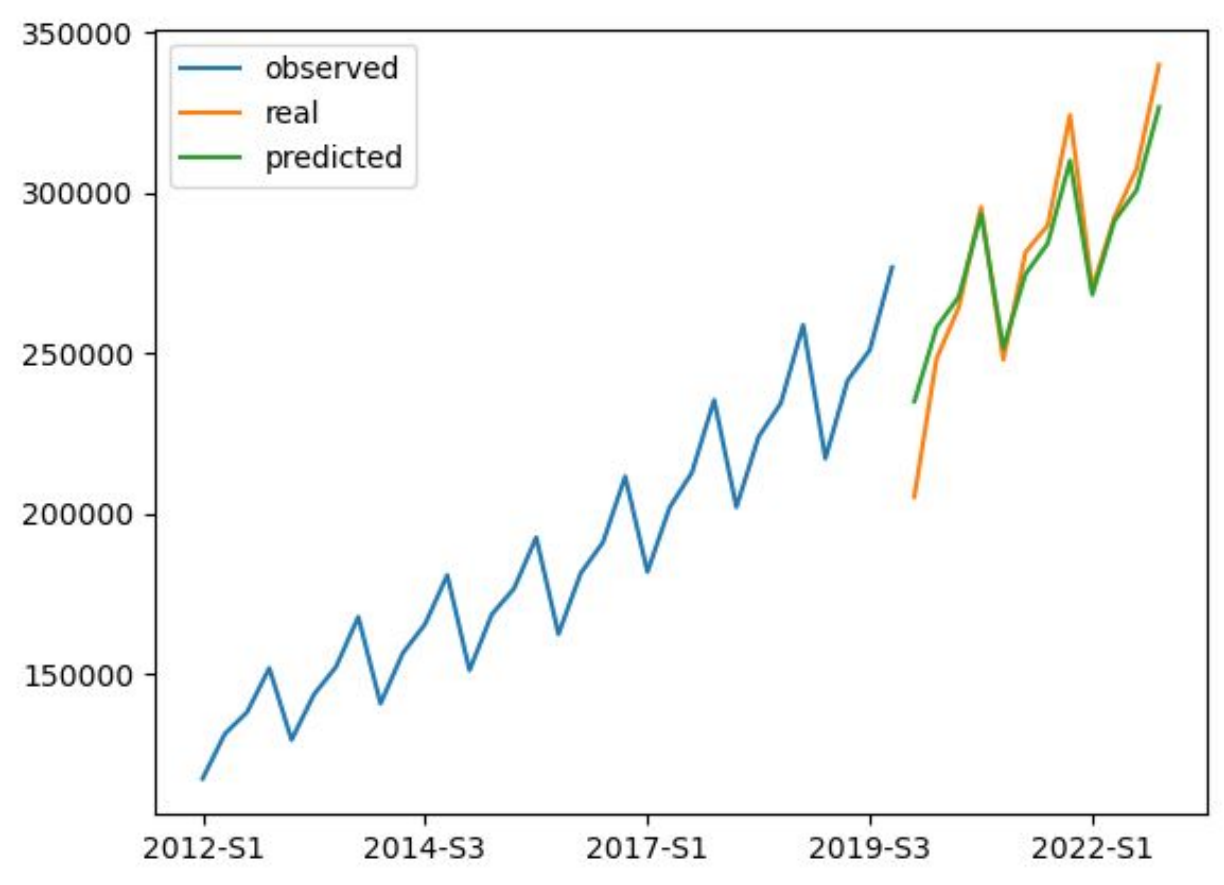
- 借助STL模型预测出的季节性规律，用ARIMA模型拟合2020年前的趋势项，预测如果没有疫情的季度GDP；

```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.forecasting.stl import STLForecast


stlf = STLForecast(data[0:8*4], ARIMA,period=4, model_kwargs=dict(order=(2, 2, 0)))
stlf_res = stlf.fit()

forecast = stlf_res.forecast(12)
```

ARIMA模型参数

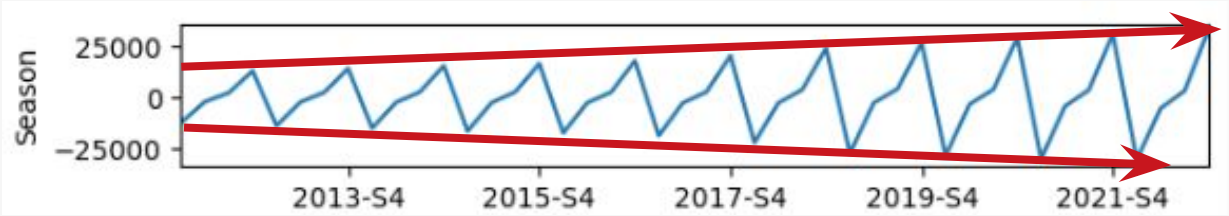
时间序列分解

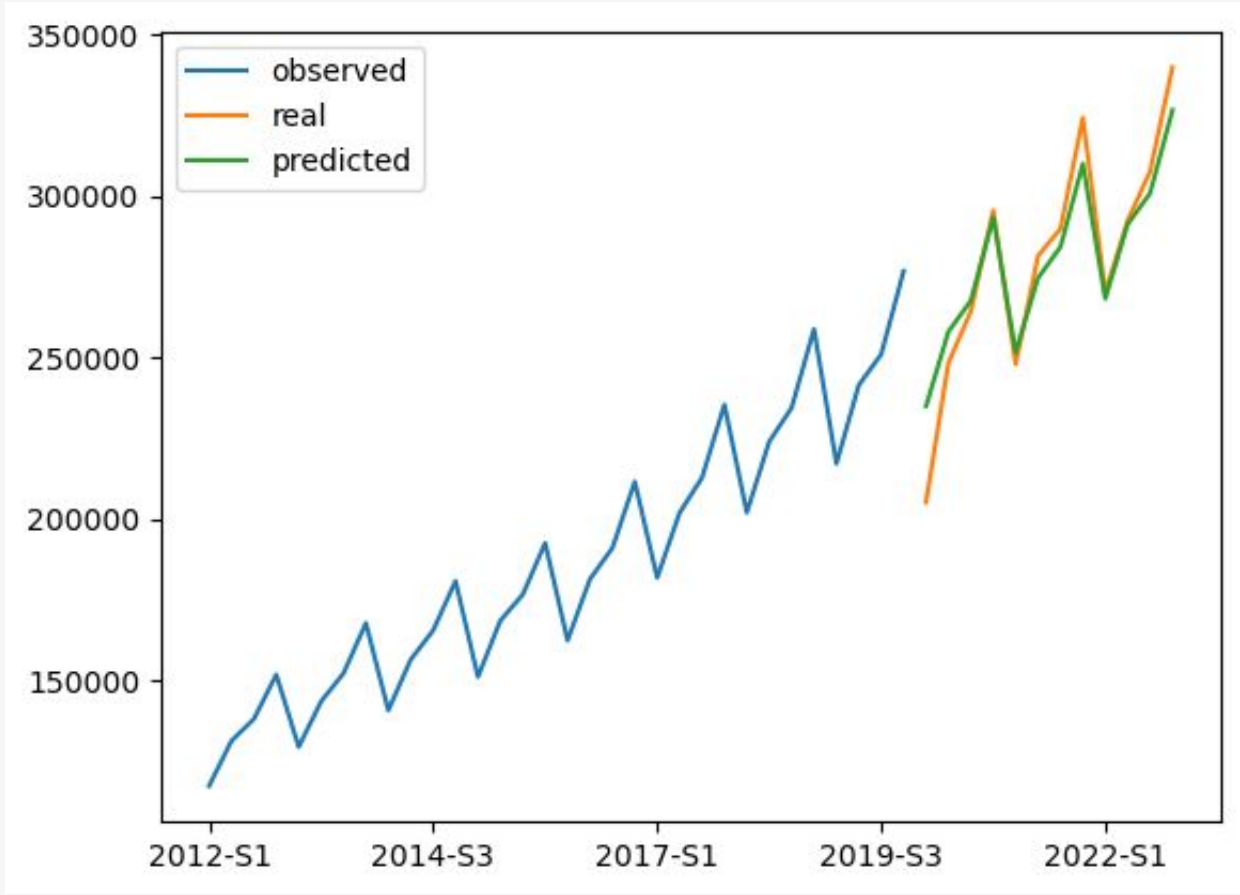
- 借助STL模型预测出的季节性规律，用ARIMA模型拟合2020年前的趋势项，预测如果没有疫情的季度GDP；



时间序列分解

- 借助STL模型预测出的季节性规律，用ARIMA模型拟合2020年前的趋势项，预测如果没有疫情的季度GDP；
- 模型预测低于真实情况的可能原因？
  - STL没有建模季节项的不稳定性；

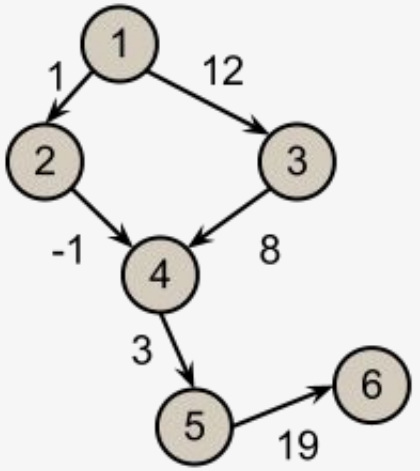




图网络数据

- 一个图中的元素：
  - 节点 V
  - 边　E
  - G = (V,E)
- 节点的连接关系可以由邻接矩阵来描述

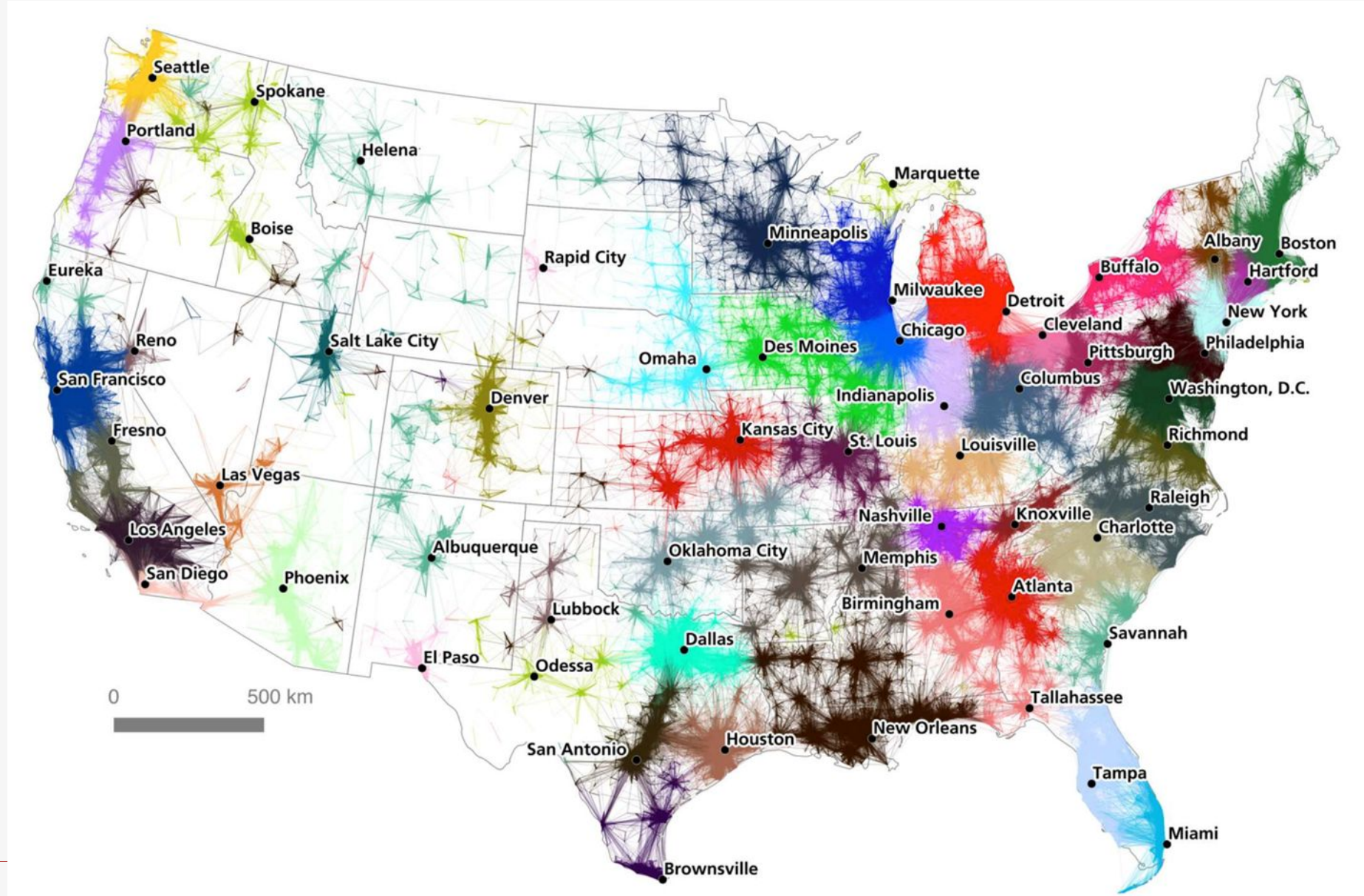
Weighted Directed Graph & Adjacency Matrix



	1	2	3	4	5	6
1	0	1	12	0	0	0
2	-1	0	0	-1	0	0
3	-12	0	0	8	0	0
4	0	1	-8	0	3	0
5	0	0	0	-3	0	19
6	0	0	0	0	-19	0

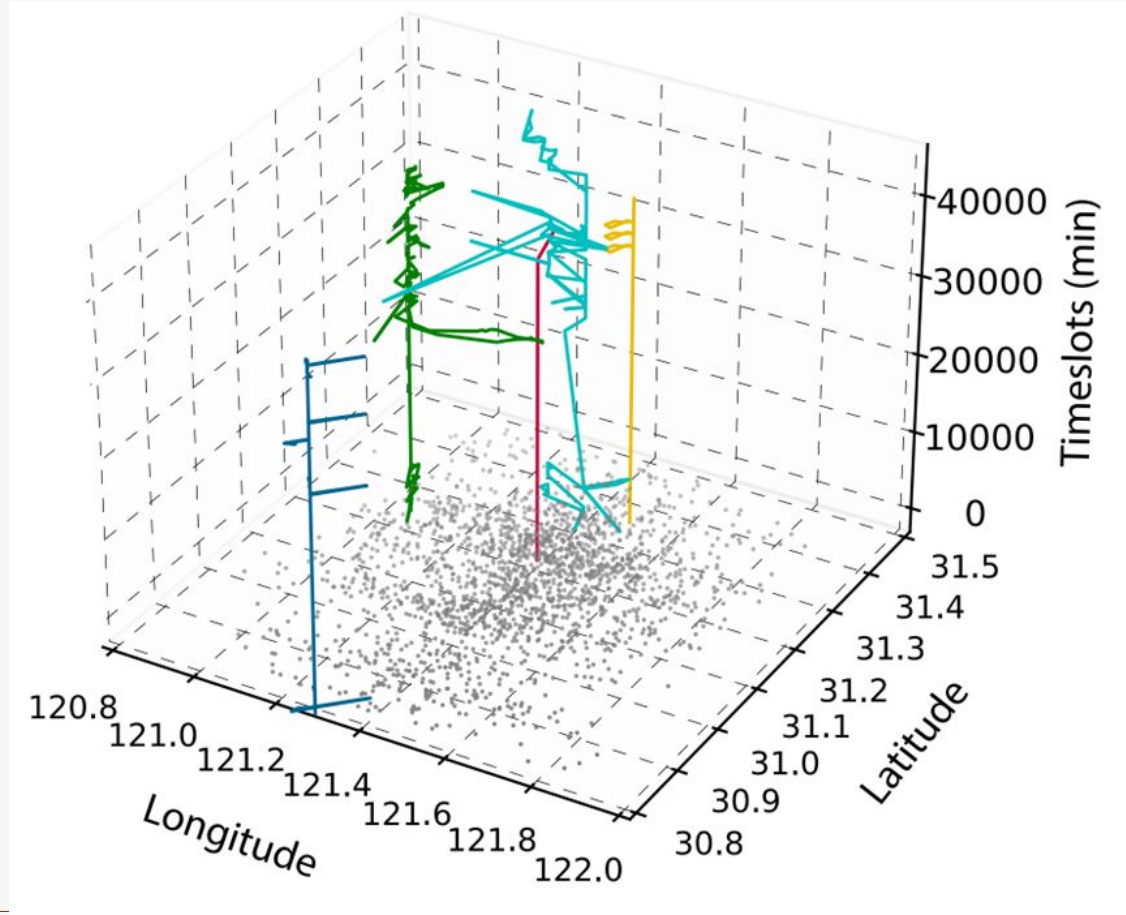
Weighted Directed Graph　　Adjacency Matrix

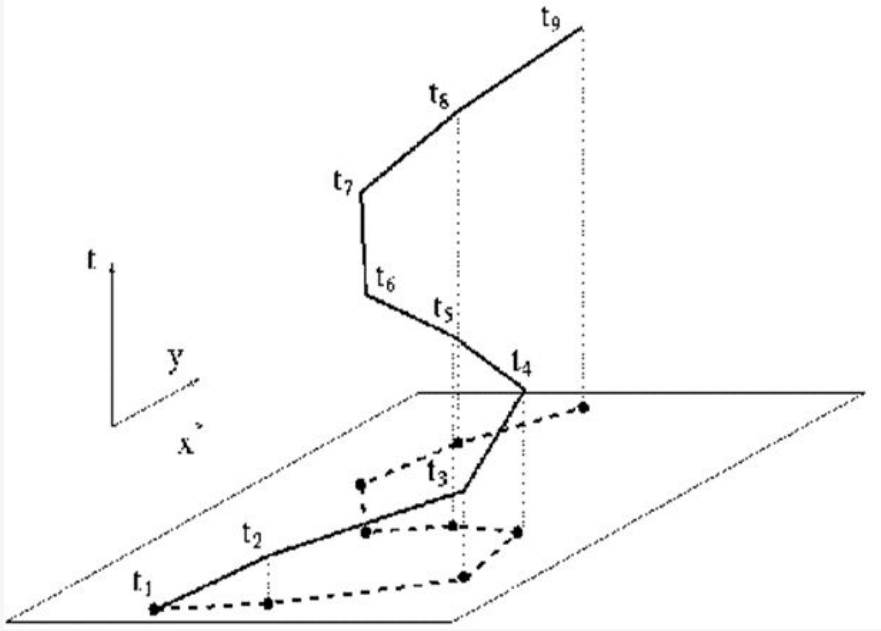
美国通勤网络


Seattle
Spokane
Portland
Helena
Boise
Eureka
Rapid City
Minneapolis
Marquette
Albany
Boston
Hartford
Buffalo
Milwaukee
Detroit
New York
Cleveland
Chicago
Reno
Salt Lake City
Des Moines
Pittsburgh
Philadelphia
Omaha
San Francisco
Columbus
Washington, D.C.
Denver
Indianapolis
Fresno
Kansas City
St. Louis
Louisville
Richmond
Las Vegas
Raleigh
Nashville
Knoxville
Charlotte
Los Angeles
Albuquerque
Oklahoma City
Memphis
San Diego
Phoenix
Atlanta
Lubbock
Birmingham
Savannah
Dallas
El Paso
Odessa
0
500 km
Tallahassee
New Orleans
Houston
San Antonio
Tampa
Miami
Brownsville

时空数据

- 基本形式：轨迹T = (t1,x1,y1),(t2,x2,y2)...(tn,xn,yn)
  - 不同时刻在二维平面中的位置串联形成轨迹





Tid	position (x,y)		time (t)
1	48.890018	2.246100	08:25
1	48.890018	2.246100	08:26
...	...		...
1	48.890020	2.246102	08:40
1	48.888880	2.248208	08:41
1	48.885732	2.255031	08:42
...	...		...
1	48.858434	2.336105	09:04
1	48.853611	2.349190	09:05
...	...		...
2	...		...

半结构化数据

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

- 相比于结构化数据，并不具备固定结构，不遵循传统表格数据形式；
- 但通过某些约定的规则和标记可以“描述”自身结构，例如XML和json格式的数据；
- 在右侧的两个数据中，两个人的属性个数和类别都不同，相比于表格形式，灵活性和扩展性都更高；

```
<person>
    <name>A</name>
    <age>13</age>
    <gender>female</gender>
</person>
#一个名字是A，年龄13岁，性别为女的人

<person>
    <name>B</name>
    <job>lawyer</job>
</person>
#一个名字是B，职业为律师的人
```

文本数据

- 训练大规模语言模型的第一步就是获取足够多的文本数据！

RU how do you use text data to train your language model?

There are several steps involved in using text data to train a language model, and the specific approach can vary depending on the type of language model you want to create. However, in general, the following steps are involved:

1. Collecting text data: The first step is to gather a large amount of text data that represents the language the model is intended to understand and generate. This can include sources such as books, articles, social media posts, and other online content.

S2.2.3 Approach to Tweet Deletion

As shown in previous work, tweet deletion may impact the results of rumor studies on Twitter [56]. We therefore included all tweets that were made available to us by the full Twitter historical archives. Since our data is anonymized and since we have a direct relationship with Twitter, we can continue to include in our analysis any tweet that was deleted after we received our data, which means our analysis is less prone to errors from tweet deletions than other studies of rumor cascades on Twitter.

网页内容解析

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

使用requests库获取网页HTML文本。HTML是一种用于创建网页的标准标记语言。

```
import requests
from bs4 import BeautifulSoup
import json
```

```
html_txt = requests.get("https://movie.douban.com/subject/1292063/",headers={"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
print(html_txt)
```

```
<!DOCTYPE html>
<html lang="zh-CN" class="ua-windows ua-webkit">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="renderer" content="webkit">
    <meta name="referrer" content="always">
    <meta name="google-site-verification" content="okOwCgT20tBBgo9_zat2iAcimtN4Ftf5ccsh092Xeyw" />
    <title>
        美丽人生 (豆瓣)
</title>

    <meta name="baidu-site-verification" content="cZdR4xxR7RxmM4zE" />
    <meta http-equiv="Pragma" content="no-cache">
    <meta http-equiv="Expires" content="Sun, 6 Mar 2005 01:00:00 GMT">

    <link rel="apple-touch-icon" href="https://img1.doubanio.com/f/movie/d59b2715fdea4968a450ee5f6c95c7d7a2030065/pics/movie/apple-touch-i
con.png">
    <link href="https://img1.doubanio.com/f/shire/204847ecc7d679de915c283531d14f16cfbee65e/css/douban.css" rel="stylesheet" type="text/cs
s">
```

网页内容解析

使用BeautifulSoup4进行网页内容解析，调用find_all方法获取所有id属性为info的div标签。

```
soup = BeautifulSoup(html_txt)

soup.find_all('div',{"id":"info"})
```

```
[<div id="info">
 <span><span class="pl">导演</span>: <span class="attrs"><a href="/celebrity/1041004/" rel="v:directedBy">罗伯托·贝尼尼</a></span></span><br/>
 <span><span class="pl">编剧</span>: <span class="attrs"><a href="/celebrity/1042013/">温琴佐·切拉米</a> / <a href="/celebrity/1041004/">罗伯托·贝尼尼</a></span></span><br/>
 <span class="actor"><span class="pl">主演</span>: <span class="attrs"><a href="/celebrity/1041004/" rel="v:starring">罗伯托·贝尼尼</a> / <a href="/celebrity/1000375/" rel="v:starring">尼可莱塔·布拉斯基</a> / <a href="/celebrity/1000368/" rel="v:starring">乔治·坎塔里尼</a> / <a href="/celebrity/1082051/" rel="v:starring">朱斯蒂诺·杜拉诺</a> / <a href="/celebrity/1278654/" rel="v:starring">赛尔乔·比尼·布斯特里克</a> / <a href="/celebrity/1002839/" rel="v:starring">玛丽萨·帕雷德斯</a> / <a href="/celebrity/1036601/" rel="v:starring">霍斯特·布赫霍尔茨</a> / <a href="/celebrity/1060607/" rel="v:starring">利迪娅·阿方西</a> / <a href="/celebrity/1125178/" rel="v:starring">朱利亚娜·洛约迪切</a> / <a href="/celebrity/1283951/" rel="v:starring">亚美利哥·丰塔尼</a> / <a href="/celebrity/1352558/" rel="v:starring">彼得·德·席尔瓦</a> / <a href="/celebrity/1352559/" rel="v:starring">弗朗西斯·古佐</a> / <a href="/celebrity/1352560/" rel="v:starring">拉法埃拉·莱博罗尼</a> / <a href="/celebrity/1352579/" rel="v:starring">克劳迪奥·阿方西</a> / <a href="/celebrity/1035248/" rel="v:starring">吉尔·巴罗尼</a> / <a href="/celebrity/1066720/" rel="v:starring">马西莫·比安奇</a> / <a href="/celebrity/1375179/" rel="v:starring">恩尼奥·孔萨尔维</a> / <a href="/celebrity/1375181/" rel="v:starring">吉安卡尔洛·科森蒂诺</a> / <a href="/celebrity/1375180/" rel="v:starring">阿伦·克雷格</a> / <a href="/celebrity/1098805/" rel="v:starring">汉尼斯·赫尔曼</a> / <a href="/celebrity/1115749/" rel="v:starring">弗兰科·梅斯科利尼</a> / <a href="/celebrity/1128312/" rel="v:starring">安东尼奥·普雷斯特</a> / <a href="/celebrity/1147135/" rel="v:starring">吉娜·诺维勒</a> / <a href="/celebrity/1337751/" rel="v:starring">理查德·塞梅尔</a> / <a href="/celebrity/1152853/" rel="v:starring">安德烈提多娜</a> / <a href="/celebrity/1047318/" rel="v:starring">迪尔克·范登贝格</a> / <a href="/celebrity/1056773/" rel="v:starring">奥梅罗·安东努蒂</a></span></span><br/>
 <span class="pl">类型:</span> <span property="v:genre">剧情</span> / <span property="v:genre">喜剧</span> / <span property="v:genre">爱情</span> / <span property="v:genre">战争</span><br/>
 <span class="pl">制片国家/地区:</span> 意大利<br/>
 <span class="pl">语言:</span> 意大利语 / 德语 / 英语<br/>
 <span class="pl">上映日期:</span> <span content="2020-01-03(中国大陆)" property="v:initialReleaseDate">2020-01-03(中国大陆)</span> / <span content="1997-12-20(意大利)" property="v:initialReleaseDate">1997-12-20(意大利)</span><br/>
 <span class="pl">片长:</span> <span content="116" property="v:runtime">116分钟(国际版)</span> / 125分钟<br/>
 <span class="pl">又名:</span> 一个快乐的传说(港) / Life Is Beautiful<br/>
 <span class="pl">IMDb:</span> tt0118799<br/>
 </div>]
```

网页内容解析

调用find_all方法获取所有type属性为application/ld+json的script标签，内容是一个json字符串，用json库解析得到一个Python中的字典对象；

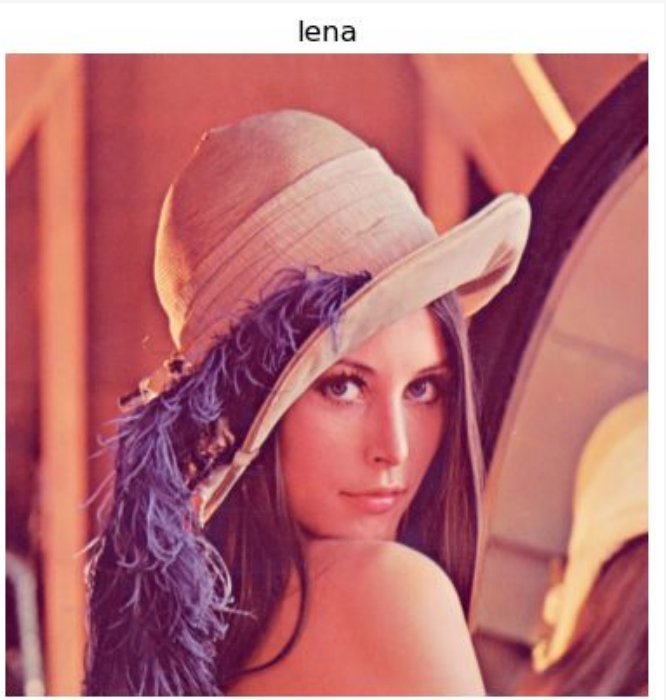
```
temp = soup.find_all('script',{"type":"application/ld+json"})[0]

js_dict = json.loads(temp.contents[0].strip(),strict=False)
```

```
js_dict
```

```
{'@context': 'http://schema.org',
 'name': '美丽人生 La vita è bella',
 'url': '/subject/1292063/',
 'image': 'https://img2.doubanio.com/view/photo/s_ratio_poster/public/p2578474613.jpg',
 'director': [{'@type': 'Person',
   'url': '/celebrity/1041004/',
   'name': '罗伯托·贝尼尼 Roberto Benigni'}],
 'author': [{'@type': 'Person',
   'url': '/celebrity/1042013/',
   'name': '温琴佐·切拉米 Vincenzo Cerami'},
  {'@type': 'Person',
   'url': '/celebrity/1041004/',
   'name': '罗伯托·贝尼尼 Roberto Benigni'}],
 'actor': [{'@type': 'Person',
   'url': '/celebrity/1041004/',
   'name': '罗伯托·贝尼尼 Roberto Benigni'},
  {'@type': 'Person',
   'url': '/celebrity/1000375/',
   'name': '尼可莱塔·布拉斯基 Nicoletta Braschi'},
```

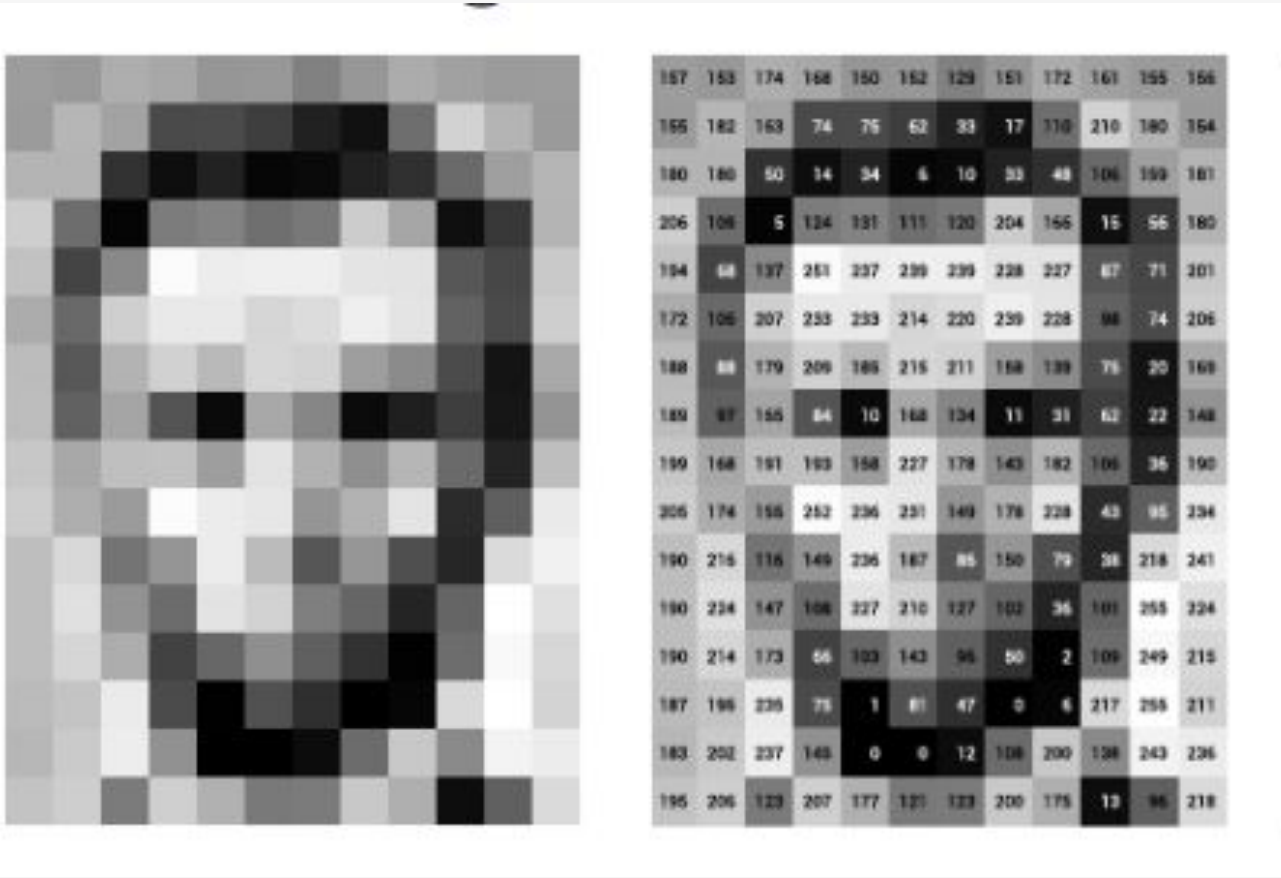
图像数据

- 在计算机系统中，颜色、亮度等信息都被数字化表示，最常见的RGB图像在python中一般被表示为Height*Width*3的3维numpy数组。
- 其中Height和Width分别表示垂直和水平方向的像素数目，3表示三个颜色通道。



图像数据

- 特别地，对于灰度图像，可以只用Height*Width的2维numpy数组来表示。
- 数组中每个数字的范围也叫图像深度，例如8bit图像即每个数字的范围是0-255，可以用8个比特(0-1)表示。



图像在计算社会科学中的应用场景

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

- 人脸识别
  - 情绪检测
- 卫星图像、航拍图像
  - 大气污染监测
  - 退耕还林实时监测
  - 违法建筑实时监测
- 文档智能
  - 文档数字化

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY