自回归建模和趋势拟合建模

一、时间序列基础

1.1 概述

什么是时间序列？时间序列是按时间的先后顺序排列的一串数值。

比如在 M5 这个比赛中，sales 按照时间排列的这一串数值就是时间序列。

以上描述，是时间序列一个比较直观的感受。从严格的数学意义上来讲，我们把时间序列看成了一串随机变量。

$$(X_1, X_2, \cdots, X_t)$$

每一个 X_i 都代表了一个随机变量（注意这里和机器学习问题的区别P(Y|X)）。一个随机变量一般需要采样到多次才能比较好的估计这个随机变量的一些性质。但对于时间序列问题，我们只能采样到一次（因为历史不可回溯，也就是观察到它的值一次），采样到的就是时间序列的观察值。这一条观察值形成的序列，也称为轨道。假设我们从上帝视角来看，如果我们可以回到过去，那么这条时间序列理论上有无数条观察值序列（轨道），但现实中我们只能观察到一条轨道。所以时间序列的难点就是如何从一条轨道，尽量的得到更多的“信息”。

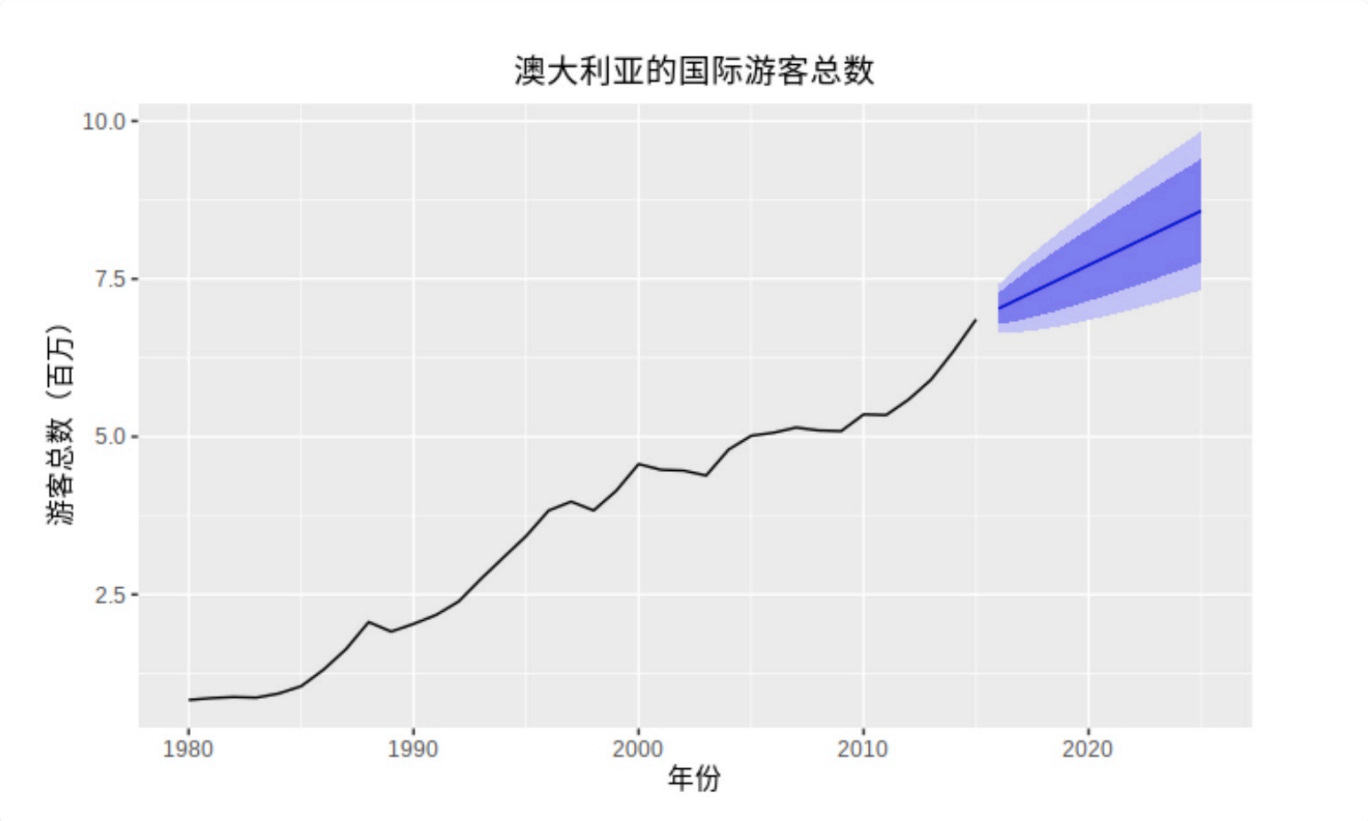
我们研究时间序列的目的，主要是为了去对时间序列进行预测。

时间序列的预测分为两种：

- 点预测
- 区间预测

点预测，就是预测时间序列未来的值。比如这次的 M5 比赛，就是点预测比赛。

区间预测，就是预测时间序列未来值的范围。比如 M5 的另外一个比赛，不确定性程度比赛。本质上可以看作是区间预测。



前面说过，在数学上，我们可以把时间序列观测值看成是随机变量的采样。那么时间序列的建模就是想办法通过观测值还原这些随机变量。所以时间序列未来的值，也应该是随机变量。那么点预测一般就会是这个随机变量的均值。区间预测就是这个随机变量的比较大的概率下的取值范围。

回到上面的图，能看到，越往后的预测，区间会越大。原因是越往后，预测会越不确定。

并不是所有的时间序列都可以预测？那什么样的时间序列是可以预测的呢?

- 我们知道哪些因素会影响时间序列;
- 有大量的数据是可用的;
- 预测不会反向影响我们试图预测的事物。（无内生性，如果有内生，那么是 VAR 建模。e.g）

如果我们想定量的来预测时间序列，本质上会有如下的假设：

- 关于过去的数据是可以用的；
- 有理由假设过去的一些模式会在未来延续下去。

换句话说，我们不能期望去预测从没发生过的模式。（比如这一次的疫情，可能在历史上没有出现过）

对于 M5 ，预测的事物是 sales，从 EDA 来看是满足这些假设的。

1.2 建模方式

回归模型

$$\text{销量}_{t+1} = f(\text{日期}_{t+1}, \text{价格}_{t+1}, \text{误差}_{t+1})$$

当下时刻的预测变量由当下时刻的其他变量所决定。

这里的误差包含两部分，一部分是随机波动，另一部分是没有被其他变量所解释的信息。

自回归模型

$$\text{销量}_{t+1} = f(\text{日期}_{t+1}, \text{价格}_{t+1}, \text{销量}_{t}, \text{销量}_{t-1}, \text{销量}_{t-2}, \text{误差}_{t+1})$$

对未来的预测是基于变量的过去值，而不是基于可能影响系统的外部变量。“误差”项允许随机波动和不包含在模型中的相关变量的影响。

动态回归模型

$$\text{销量}_{t+1} = f(\text{销量}_{t}, \text{销量}_{t-1}, \text{销量}_{t-2}, \text{误差}_{t+1})$$

前面的 baseline 可以看作是回归模型。不同的是，我们还抽取了窗口特征。这个会在后面机器学习建模部分讲解。

1.3 名词释义

- 趋势
- 季节性

- 周期性

趋势
当一个时间序列数据长期增长或者长期下降时，表示该序列有 趋势 。在某些场合，趋势代表着“转换方向”。例如从增长的趋势转换为下降趋势。

季节性
当时间序列中的数据受到季节性因素（例如一年的时间或者一周的时间）的影响时，表示该序列具有 季节性 。季节性总是一个已知并且固定的频率。注意，季节性可能是复合的。

周期性
当时间序列数据存在不固定频率的上升和下降时，表示该序列有 周期性 。这些波动经常由经济活动引起，并且与“商业周期”有关。

许多初学者都不能很好的区分季节性和周期，然而这两个概念是完全不同的。当数据的波动是无规律时，表示序列存在周期性；如果波动的频率不变并且与固定长度的时间段有关，表示序列存在季节性。一般而言，周期的长度较长，并且周期的波动幅度也更大。

许多时间序列同时包含趋势、季节性以及周期性。当我们选择预测方法时，首先应该分析时间序列数据所具备的特征，然后再选择合适的预测方法抓取特征。

后面，我们用 T(Trend) 表示趋势，S(Season) 表示季节和周期，R(Residual) 表示误差。

1.4 数学释义

- 滞后
- 自相关
- 偏自相关
- 平稳性
- 白噪声

在时间序列中，我们经常会去研究当下时刻的
y_t 和以前的某个
y_{t-k} 的关系。
y_{t-k} 称为
y_t 的 k 阶滞后，记为
lag。M5 比赛中我们就用了很多 lag 的特征。

```
# df 为 pandas 中的 DataFrame 类型或者 Series 类型
df.shift(1)
```

在随机变量中，有相关性的概念，相关性度量的是两个随机变量的相关程度。比如相关系数就可以衡量两个变量之间的线性相关关系。

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

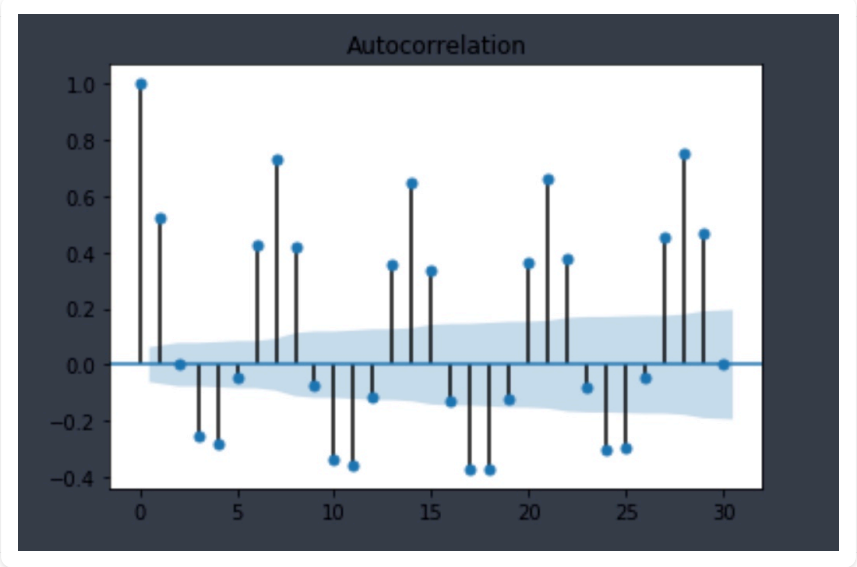
我们说过时间序列也是一系列的随机变量。那么我们期望研究时间序列中
y_t 和
y_{t-k} 之间的关系。

$$r_k = \frac{\sum(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum(y_t - \bar{y})^2}$$

r_k 表示的是 y_t 和他的 k 阶滞后的相关程度。 r_k 称为 ACF (Autocorrelation Coefficient)
所以自相关性研究的是一个时间序列在不同时间点的值的关系。

如果时间序列不存在自相关性。那就说明用历史的值来预测未来的值是没有意义的。从逻辑上来讲当下时刻 t 肯定和上一个时刻会有比较强的相关性。随着时间的推移，这种相关性会越来越弱。但是如果有季节性，那就并不是如此了，这里要特别注意。

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(data)
```



X 轴是延迟阶数，Y 轴是相关性 (–1,1) 之间。深色区域是置信区间（默认情况下，置信区间这被设置为95%。简单理解就是这个深色区域外的就可以认为是相关的）。

ACF 图和趋势性和季节性之间的关系：

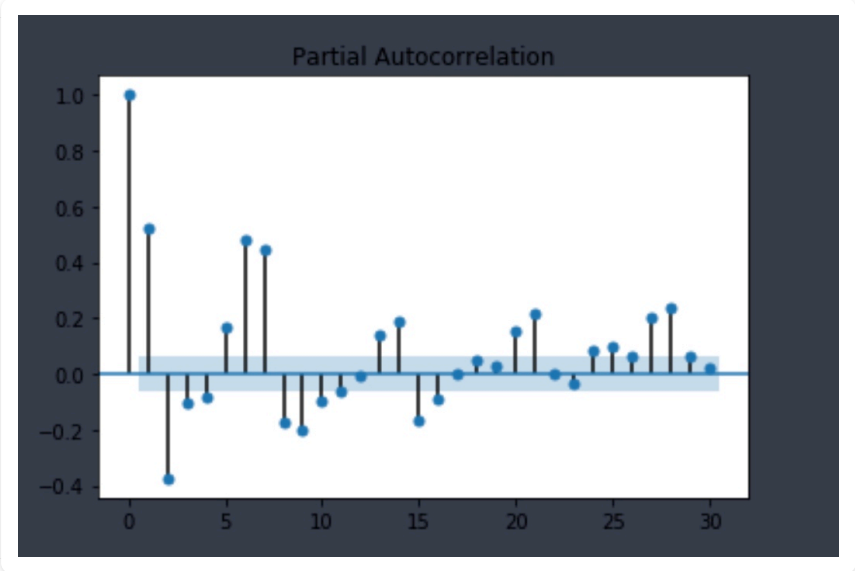
当数据具有趋势性时，短期滞后的自相关值较大，因为观测点附近的值波动不会很大。时间序列的ACF一般是正值，随着滞后阶数的增加而缓慢下降。

当数据具有季节性时，自相关值在滞后阶数与季节周期相同时（或者在季节周期的倍数）较大。

当数据同时具有趋势和季节性时，我们会观察到组合效应。

自相关描述的是
y_t 和
y_{t-k} 之间的关系。这种关系包含了直接关系和间接的关系。什么意思呢？ 假设
y_t 和
y_{t-k} 相关。那么如果
y_t 和
y_{t-1} 相关，则
y_{t-1} 和
y_{t-2} 也相关，则
y_t 就可能和
y_{t-2} 相关，这种导致的
y_t 和
y_{t-2} 的相关性是间接的关系。是由于
y_t 和
y_{t-1} 相关导致。所以我们有时候需要排除掉这种间接的关系。想去研究
y_t 和
y_{t-k} 是否有直接的关系。这个就是偏自相关性。

偏自相关性的估算公式较复杂，这里就不再展开了。



平稳性

并不是所有时间序列都具有平稳性。但有平稳性的时间序列，能帮助我们更好的通过观察值来估计随机变量的性质。

在统计学习中，有一个核心的假设，就是假设样本是独立同步分布，这样通过大数定理，就可以通过有限的样本（观察值）来估计一些性质（e.g 用经验风险去逼近期望风险）。为什么要假设独立同步分布，是因为我们只能采集到有限的样本。

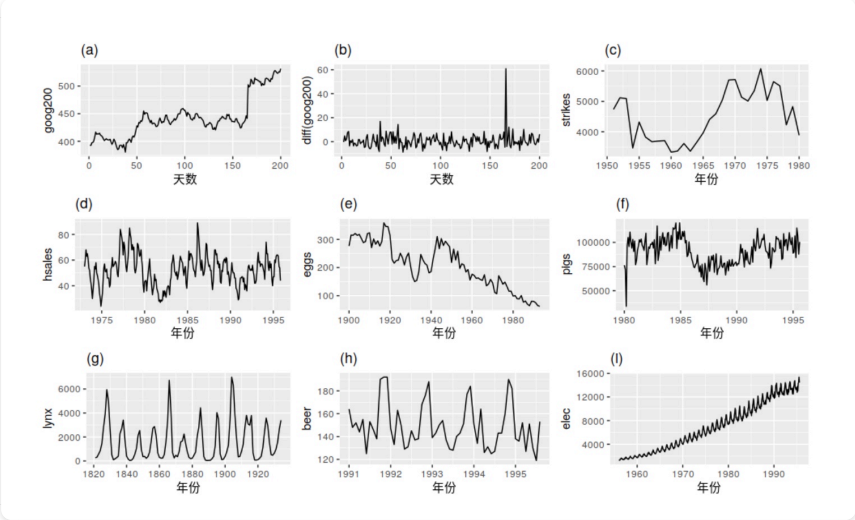
在时间序列中，也是一样的，理论上一条时间序列，能有无穷条轨道。但是我们只能采样到一条轨道。所以如何才能通过一条轨道的信息（有限的样本），来进行有效的估计?

大数定理是空间上的用有限逼近无限的保障。
在时间序列上也有一个类似的定理，是让时间序列能从时间上用有限来逼近无限。

这个定理就是时间序列遍历性定理。所以如果一条时间序列满足遍历性，我们就可以通过对一条轨道在时间上有限样本来估计一些性质。那如何保证时间序列具有遍历性呢？ 那就是这条时间序列是平稳的，且任意两点的相关性随着间隔的长度会逐渐降低。

简单点说，平稳的时间序列的性质不随观测时间的变化而变化，比如方差和均值。

所以平稳的时间序列，一定是没有趋势和季节性的。



计算相邻观测值之间的差值，这种方法被称为差分，通过查分的方法可以让时间序列变平稳。

判断时间序列平稳的方法有两种，一种是通过看图大概判断。一种是严格的通过统计方法来判别。

1. 看原始时序图，看acf和pacf的图，相关性会快速下降到0附近。
2. 单位根检验

```
from statsmodels.stats.diagnostic import unitroot_adf
```

```
unitroot_adf(data)
```

第二个值就是 p 值。p 越小代表代表我们拒绝原假设，那么该时间序列就越该平稳（一般是threshold 取0.05 或者0.01）

前面讲过所有的模型都会有误差，一个理想的模型是这个误差是一个白噪声。

“白噪声”是一个对所有时间其自相关系数为零的随机过程。即白噪声是完全随机的，不可预测的。所有可预测的信息都已经被我们蕴含到了模型之中。

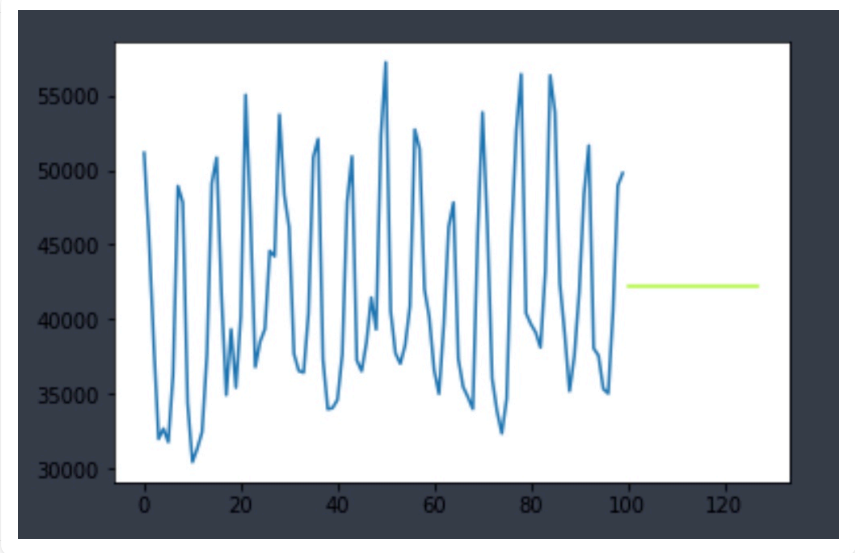
二、简单规则模型

下面我们来看一些简单的规则模型，这些规则模型虽然简单，但是在有的场景下也能得到非常好的效果。

均值法：

未来的预测值等于历史上值(一个窗口)的均值：

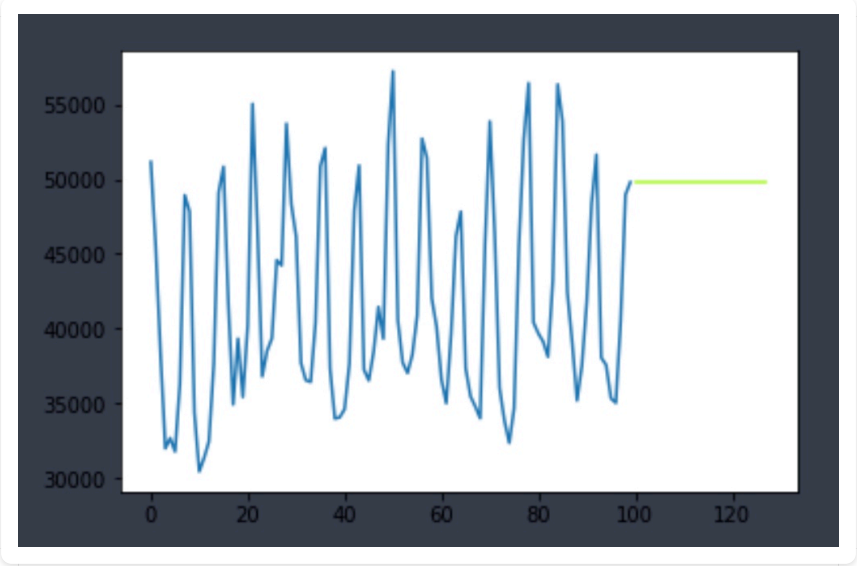
$$\hat{y_{T+h}} = \frac{1}{W}\sum_{i=1}^{W} y_{T-W+i}$$



朴素预测法：

拿最近一次的预测值，作为后面的预测值：

$$\hat{y_{T+h}} = y_T$$

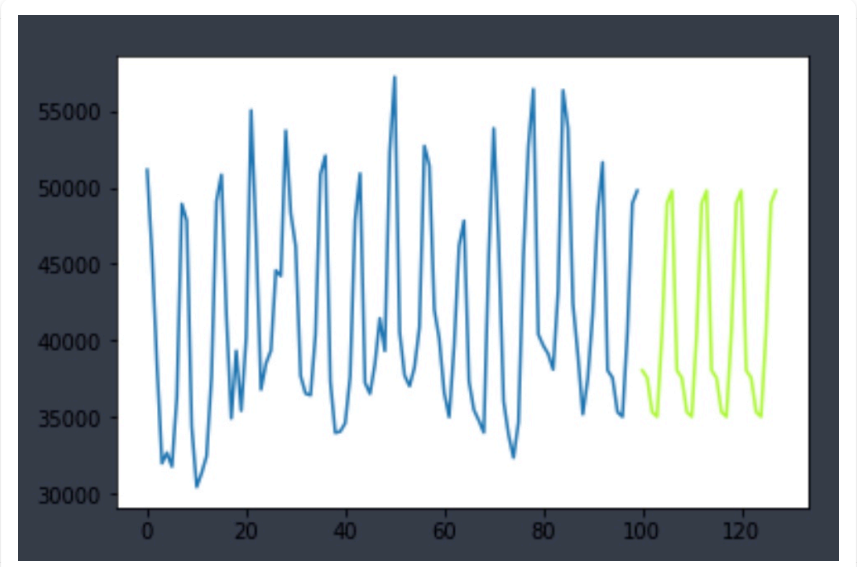


季节性朴素预测法：

我们考虑到时间序列的季节性

$$\hat{y_{T+h}} = y_{T-km}$$

m 是周期性，k 是一个整数。

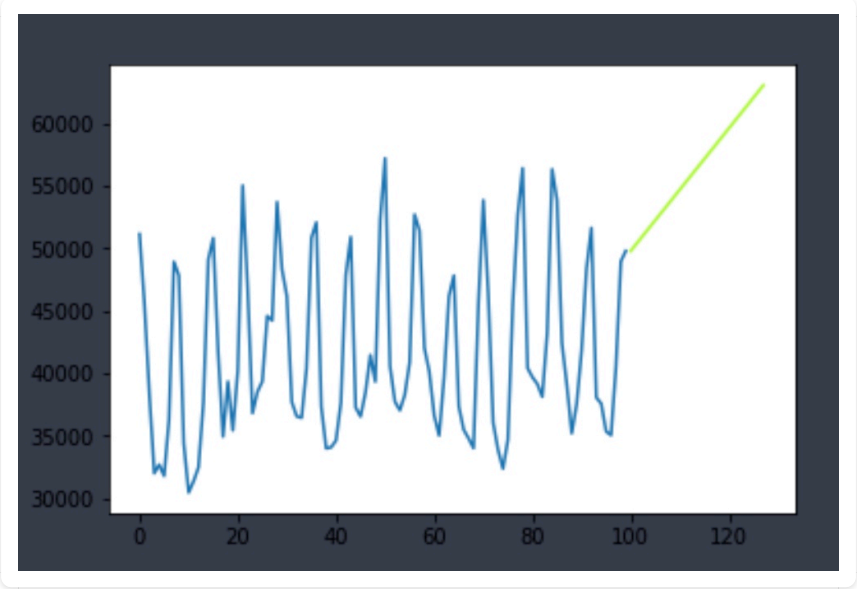


漂移法：

我们考虑到时间序列具有趋势性

$$y_{T+h} = y_T + h\frac{y_T - y_{T-W}}{W}$$

有一条向上或者向下的直线来模拟趋势，本质上是朴素预测法的推广。



数据的调整

有时候我们需要对时间序列数据进行调整，因为时间序列可能会受到一些其他潜在因素的影响。而这些潜在的因素在传统时间序列建模中，可能需要进行调整。
比如以 M5 为例。 M5 的销量数据的趋势是逐年上升，这个逐年上升的趋势，可能还有一个潜在的影响。就是人口数量的增加。人口数量的增加必然会导致物品销量的增加。（对于28天的短期预测，可以近似忽略。）

再比如，房价的预测模型，我们可能就需要将房价剔除掉通货膨胀所带来的影响。

三、时间序列分解

2.1 一般范式

时间序列数据通常有很多种潜在模式，因此一种有效的处理时间序列的方式是将其分解为多个成分，其中每个成分都对应某一种基础模式。

当我们想要把时间序列分解为多个成分时，我们通常将趋势和周期组合为“趋势–周期”项（有时也简单称其为趋势项）。因此，我们认为时间序列包括三个成分：趋势–周期项，季节项和残差项（残差项包含时间序列中其它所有信息）。

时间序列一般可以写成两种分解模式：

$$y_t = S_t + T_t + R_t$$

$$y_t = S_t \times T_t \times R_t$$

乘法模式和加法模式可以通过取 log 进行转换。

后面我们假设处理的都是加法模型。

1. 首先估计出趋势周期项
2. 将原始时间序列减去趋势周期项
3. 估计季节项

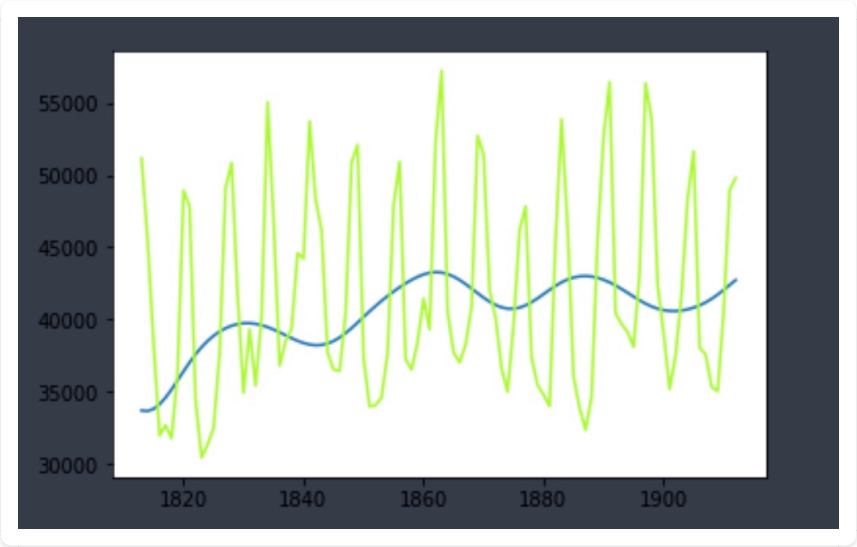
2.2 通过 MA 估计趋势周期项

将时间序列的观察值具有一定的随机性，通过使用移动平均，能很好的消除随机性，对数据做平滑，得到趋势周期项。

$$\hat{y}_t = \frac{1}{m}\sum_{i=-k}^{k} y_{t+k}$$

m 为窗口的大小，是为一个奇数 $m = 2k + 1$。

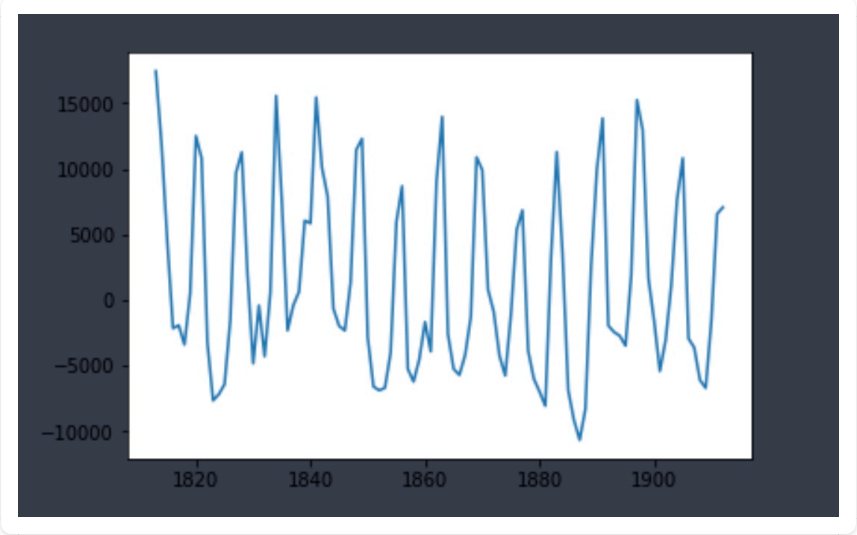
移动平均可以进行多次的累加，即做了一次移动平均之后，再做一次移动平均。



MA 可以推广到加权的移动平均，加权的移动平均可以推广到一维的CNN。

PS：高阶 MA 就是一种加权的移动平均

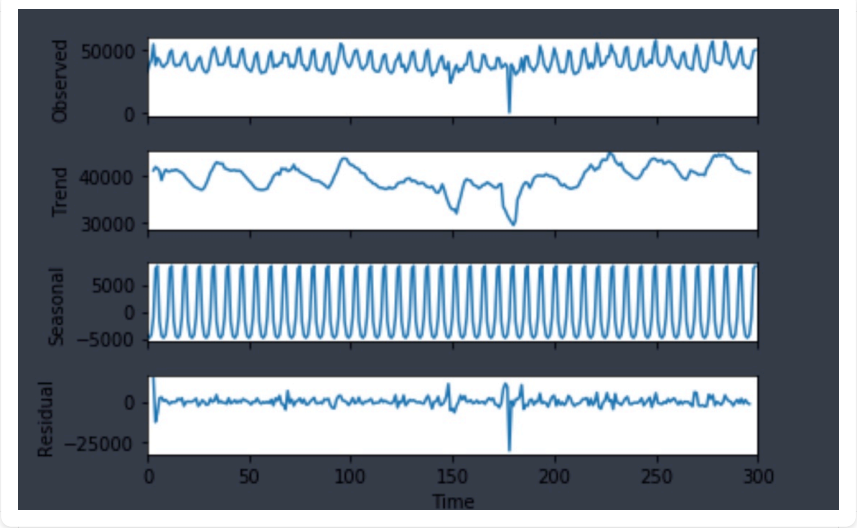
下图是减去趋势项看周期项



时序分解一个很出名的算法 STL 就是基于上面的思路来进行的。

Cleveland, Robert B., William S. Cleveland, and Irma Terpenning. "STL: A seasonal–trend decomposition procedure based on loess." Journal of Official Statistics 6.1 (1990): 3

```
import statsmodels.api as sm
rd = sm.tsa.seasonal_decompose(total_sum[-300:], freq=7)
```



有了时间序列的分解，我们可以想办法来度量时间序列的趋势性和季节性。看他是趋势性更强还是季节性更强。

$$y_t = S_t + T_t + R_t$$

下面公式度量趋势性的强度：

$$Score_T = max(0, \frac{Var(R_t)}{Var(T_t + R_t)})$$

这个值越大，说明趋势性越强。

下面公式度量度量季节性的强度：

$$Score_S = max(0, \frac{Var(R_t)}{Var(S_t + R_t)})$$

这个值很实用，当我们有很多时间序列的时候，可以通过这个方法来对时间序列做一些分类。（这个启发我们可以对30490条时间序列按照趋势性强度或者周期性强度分类，然后分别建模）。

前面提过，LGB 等树模型并不能很好的学到趋势信息。所以可以考虑把趋势信息拆分出来后，再用 lgb 模型建模。然后再用一个模型预测趋势。将lgb的结果和趋势模型融合。

四、指数平均模型

下面我们来看一类很实用的模型，指数平均模型。

前面说过，移动平均模型可以推广到加权的移动平均模型。那么有一种比较有意义的加权方法，就是离当下时刻越远，则他的权重就越小。

4.1 一阶指数平滑

$$\hat{y_{t+1}} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-1} + \cdots$$

$0 \leq \alpha \leq 1$ 很容易证明，权重的和事收敛于 1 的。
α 等于 1 则退化为了朴素预测方法。

这个指数平滑是最终的展开式，我们在实际使用中ing不会使用这个公式。而是使用

- 递归表达

$$\hat{y_{t+1}} = \alpha y_t + (1-\alpha)\hat{y_t}$$

- 分量表达

$$\hat{y_{t+1}} = l_t$$
$$l_t = \alpha y_t + (1-\alpha)l_{t-1}$$

$$
\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t \\
\text{Smoothing equation} \quad & \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \ ,
\end{aligned}
$$

从分量表达式能看出为什么我们叫它是一阶的。

```
# 一阶指数平滑模型
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
alpha = 0.15
simpleExpSmooth_model = SimpleExpSmoothing(total_sum[-10:]).fit(smooth
ing_level=alpha,optimized=False)
```

一阶指数平滑，无法预测趋势项和季节项，如果需要预测这两个，则需要引入二阶和三阶指数平滑模型。

4.2 二阶指数平滑

线性二阶指数平滑

$$
\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\
\text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \ ,
\end{aligned}
$$

阻尼二阶指数平滑

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + (\phi + \phi^2 + \cdots + \phi^h)b_t \\
\ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}。
\end{aligned}
$$

线性二阶指数平滑的问题在于趋势会一直增长，所以阻尼二阶指数平滑就解决了这个问题。当
$h \to \infty$ 的时候，
b_t 前的系数收敛到
$\frac{\phi}{1-\phi}$ 所以我们会要求

$\phi \in (0, 1)$

```
# 二阶指数平滑模型
from statsmodels.tsa.holtwinters import ExponentialSmoothing
HW_ExpSmooth_model = ExponentialSmoothing(total_sum[-120:], seasonal_p
eriods=7, trend='add', seasonal='add', damped=True).fit()
```

4.3 三阶指数平滑

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t-m+h_m^+} \\
\ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \ ,
\end{aligned}
$$

当季节变化在该时间序列中大致保持不变时，通常选择加法模型；而当季节变化与时间序列的水平成比例变化时，通常选择乘法模型。

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
HW_ExpSmooth_model = ExponentialSmoothing(total_sum[-120:], seasonal_p
eriods=7, trend='add', seasonal='add', damped=True).fit()
```

三阶指数模型就是出名的 holt-winter 模型。这个模型可以考虑时间序列的趋势，季节性。是一个很常用的指数模型。

五、自回归建模

与指数平滑模型针对于数据中的趋势（trend）和季节性（seasonality）不同，自回归模型旨在描绘数据的自回归性（autocorrelations）。最出名的自回归模型就是 ARIMA 模型。

首先自回归模型的参数要求较高，且对时间序列有很强的要求，不然无法对模型的参数进行有效的估计。

一般我们要求时间序列具有平稳性（弱平稳即可）。如果时间序列不是平稳的，则需要对时间序列进行差分处理。

5.1 AR 模型

AR 模型是 auto correlation 的简称。在自回归模型中，我们是基于目标变量历史数据的组合对目标变量进行预测。自回归一词中的自字即表明其是对变量自身进行的回归。

自回归模型的公式表达如下：

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots \phi_p y_{t-p} + \epsilon_t$$

ϵ_t 是白噪声。这个模型称为 p 阶的自回归模型。简称 $AR(p)$

该模型需要作用在平稳时间序列上，且对回归的系数 ϕ 有一些约束。在这里我们就不做过多的讨论。有兴趣同学可以参考相关的资料。

5.2 MA 模型

MA 模型是 moving average 的简称。移动平均模型和自回归模型类似，他是通过历史的误差来建立一个类似回归的模型。

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3} \cdots \theta_q \epsilon_{t-q}$$

ϵ_t 是白噪声。注意，在预测过程中，ϵ_t 一般会设置为 0 。ϵ_{t-k} 是预测的误差。

和 AR 模型一样，MA 模型需要作用在平稳时间序列上，且对回归的系数 ϵ 有一些约束。在这里我们就不做过多的讨论。有兴趣同学可以参考相关的资料。

5.3 ARIMA 模型

ARIMA 模型就是把 AR 模型和 MA 模型合起来，并通过差分的方式，使得不平稳的时间序列变平稳。（ I 表示 Integrated 在这里指差分的逆过程）。

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots \phi_p y_{t-p} + \epsilon_t + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3} \cdots \theta_q \epsilon_{t-q}$$

以上模型称为 ARIMA 模型，简记为 $ARIMA(p, d, q)$， p 表示 AR 部分的阶数。 q 表示 MA 部分的阶数。 d 表示差分的阶数。

构建 ARIMA 模型的三个步骤：

- 做差分，使得时间序列平稳
- 估计阶数 p 和 q
- 训练模型，学习参数

通过 acf 和 pacf 可以快速的确定 p 和 q。

表15-6　选择ARIMA模型的方法

模　型	ACF	PACF
ARIMA(p, d, 0)	逐渐减小到零	在p阶后减小到零
ARIMA(0, d, q)	q阶后减小到零	逐渐减小到零
ARIMA(p, d, q)	逐渐减小到零	逐渐减小到零

PS：个人建议，如果阶数不大，可以把所有 p, q 的排列组合都尝试一遍。

对于小白，可以用一个更简单的方式。使用 python 的 auto_arima 包。具体使用见 jupyter 代码。这里就不进一步阐述了。

最周强调一下，ARIMA 模型在预测阶段，可以做点预测，也可以做区间预测。做点预测的时候，**预测值来代替未来的观测值，用零代替未来的预测误差，用对应的残差代替历史误差。**

5.4 SARIMA 模型

ARIMA 模型无法捕捉到季节性，所以我们需要将 ARIMA 模型进行推广，推广到季节性的 ARIMA 模型，也就是 SARIMA 模型。SARIMA 模型只是在 ARIMA 模型的基础上引入了季节项，所以 SARIMA 会包含两套参数。一套参数是 ARIMA 模型的，一套是季节项模型的。

所以有
p, d, q 和
P, D, Q 共记 6 个阶。小写的表示非季节性部分，大写的表示季节性部分，简记为 $ARIMA(p, d, q)(P, D, Q)_m$，m 表示周期。这时候可能会有同学问，他的数学公式是什么样子的？这时候由于有了符合，用前面的数学公式表达就会非常的复杂。所以一般会引入延迟算子来表示 SARIMA 的数学公式表达。

比如 ARIMA(p,d,q) 可以表达为下面公式：

$$\underset{\uparrow}{(1-\phi_1 B-\cdots-\phi_p B^p)}\quad \underset{\uparrow}{(1-B)^d y_t}\quad =\quad \underset{\uparrow}{c+(1+\theta_1 B+\cdots+\theta_q B^q)\varepsilon_t}$$
$$\text{AR}(p)\qquad\quad d\text{ differences}\qquad\quad \text{MA}(q)$$

$ARIMA(1, 1, 1)(1, 1, 1)_4$

$$(1-\phi_1 B)\,(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1+\theta_1 B)\,(1+\Theta_1 B^4)\varepsilon_t.$$

在这里不做过多的讨论。

SARIMA 的建模，同样可以使用 python 的 auto_arima 包。具体使用见 jupyter 代码。这里就不进一步阐述了。

六、回到赛题的思考

现在我们已经学会了一些常用的时间序列建模方法。那么该如何使用这些方法呢?

一方面，我们可以根据前面学习到的时间序列的性质，30490 条时间序列进行分类。期望用不同的模型来学习不同类别的时间序列。甚至不同的类别的时间序列的后处理是不一样的。（e.g 这一部分时间序列的后处理乘的值偏大，令一些偏小，而不是baseline中统一乘一个数字）

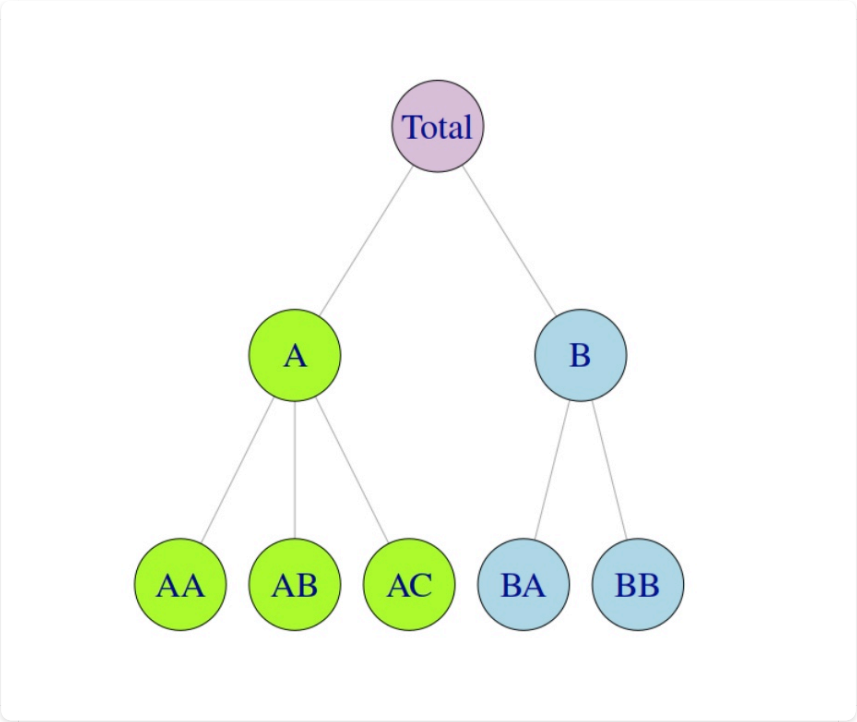
另一方面，虽然以上的传统模型都指用到了预测变量的信息，没有用到其他变量的信息（e.g 价格，节假日等，实际上 Arimax 模型是可以引入这种外生变量）。但对我们仍然有一些启发意义。因为我们可以对一些聚合后的时间序列进行预测，聚合后的时间序列，价格之类的外生变量就没意义了，节假日的大部分信息也大部分反应在了历史数据中（后面学习 prophet 模型将更好的引入节假日）。对聚合后的值的预测，能帮助我们知道后续预测变量的大致大小。

PS：当然，你也可以尝试直接对每一条时间序列建立上述模型，但会花费比较多的运行时间。

七、层次时间序列

7.1 概述

所谓层次时间序列，是指我们有很多的底层的时间序列。这些底层的时间序列可以通过聚合，形成一个层次结构。比如下图，就是一个简单的层次时间序列：



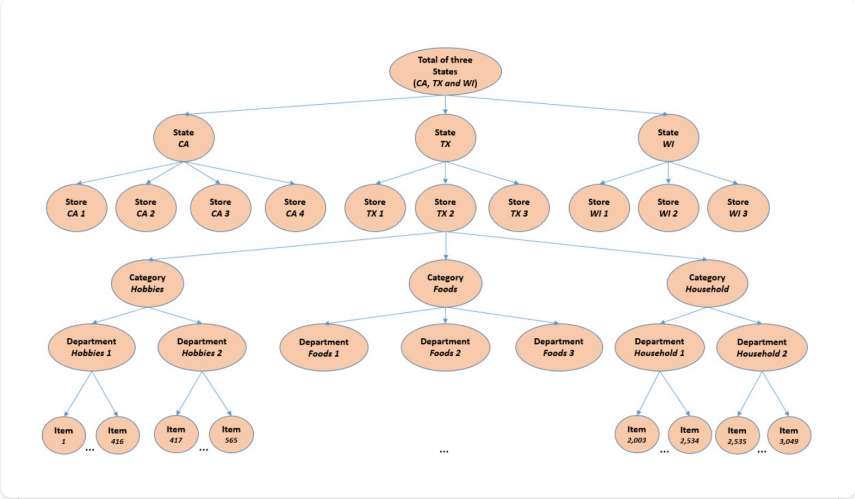
这个时间序列，底层一共有 5 条时间序列。但通过一层一层的聚合，我们一共可以得到， $5+2+1=8$ 条时间序列。

那么如何是原始的 5 条时间序列快速的得到 8 条层次时间序列呢？最简单的方式就是通过映射矩阵：

$$\begin{bmatrix} y_t \\ y_{\mathrm{A},t} \\ y_{\mathrm{B},t} \\ y_{\mathrm{AA},t} \\ y_{\mathrm{AB},t} \\ y_{\mathrm{AC},t} \\ y_{\mathrm{BA},t} \\ y_{\mathrm{BB},t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{\mathrm{AA},t} \\ y_{\mathrm{AB},t} \\ y_{\mathrm{AC},t} \\ y_{\mathrm{BA},t} \\ y_{\mathrm{BB},t} \end{bmatrix}$$

通过这个映射矩阵，我们可以把底层的 5 条时间序列，生成 8 条层次时间序列。

对于 M5 赛题，其实也是一个层次时间序列问题。



所以，我们不仅仅要预测准底层的时间序列，我们更需要预测准聚合后的时间序列。

有时候原始的时间序列无法进行很好的预测（具有更大的随机性），很可能聚合后的时间序列才有更好的模式。想象这么一个例子，有一条固定模式生成的时间序列，接着按照随机划分，将这条固定的时间序列拆分成5条时间序列。在这种情形下，想通过分别预测 5 条时间序列是不太现实的。

7.2 三种预测方式

对与分层时间序列的预测，大致可以有三种方案。

- 自下而上的预测
- 自上而下的预测
- 自中向上下的预测

以 M5 比赛为例，我们分别取预测底层的 30490 条时序列。然后通过映射矩阵聚合得到 42840 条时间序列。这是一种自下而上的预测方法。

自下而下的预测方法的核心在于预测准每一条底层的时间序列。

除了自下而上的预测方法以外，我们可以反其道而行之，使用自上而下的预测方法。

举一个例子，我们得到聚合的时间序列（全部聚合得到一条），并对该时间序列进行预测建模。将预测的值通过某种方式分配给底层的时间序列。根据分配方式的不同，有不同的实现方案。

1. 使用历史平均比例。

通过遍历历史值，得到每一条底层时间序列占这条总的时间序列的比值。

1. 对分配比例进行预测

通过模型，去预测底层时间序列占这条总的时间序列比值的变化情况。

最后一种方式是自中向上下，我们可以选择一个中间聚合的时间序列进行预测。比如按店铺，甚至是按商品品类。因为过多的聚合，会导致信息的丢失，所以我们可以选择一个中间层次的聚合时间序列进行预测，然后自上自下进行展开预测。

下节课会讲 Prophet 模型建模 和机器学习建模(lgb)