时间序列预测

利用历史数据预测未来数据走向就是时间序列预测的本质。

预测是时间序列分析的最常见任务，主要目标是利用过去的数据来预测未来的数据。也称为"Forecasting".

时序有监督学习包括时间序列回归和时间序列分类两种有监督学习方法（Time Series Regression/ Time Series Classification），指的是在多变量时间序列上，基于时间或时间相关的数据，完成回归或分类任务，这类任务对样本与样本之间的时间顺序、相关等都没有要求、反而更关注特征之间的关系，因此这类任务要求：

- 数据中必须存在除时间之外的其他特征，且特征越多越好
- 可以完成"去时序化"处理，将时序数据彻底变为一般机器学习数据

  这一类任务是学习特征值、预测标签值，并不关心样本。

1.1 概念

时间序列是按时间顺序排列的数据集合。这些数据可以在任何时间间隔（例如，每秒、每分钟、每小时、每天、每月、每年等）观察和收集。在统计学中，时间序列分析包括了一系列用于分析时间序列数据的方法，目的是提取出数据中的有意义的统计信息并对未来进行预测。

时间序列数据的一个关键特性是其有序性，即数据点之间的顺序非常重要。在此例中，如果我们打乱了月份的顺序，那么就无法从数据中看出销售额的变化趋势了。

单变量数据

月份	销售额
一月	1200
二月	1400
三月	1600
四月	1500
五月	1700
六月	1800
七月	2000
八月	2200
九月	2400
十月	2600
十一月	2800
十二月	3000

1.2 分类：单变量、多变量时间序列数据

时间序列模型(一)：时序的相关概念_时间序列分析 数据是不是必须得按时间间隔排列-CSDN博客

1.2.1 单变量时间序列

单变量时间序列只涉及一个变量随时间的变化。这个变量在每个时间点都有一个观察值。例如，每天的股票价格、每月的销售额、每小时的气温等，这些都是单变量时间序列的例子。

1.2.2 多变量时间序列

多变量时间序列涉及到两个或更多的变量随时间的变化。在每个时间点，这些变量都有观察值。例如，一个公司可能会记录每天的销售额、访客数量、广告支出等多个变量，这就构成了一个多变量时间序列。在多变量时间序列预测中，我们不仅仅使用一个变量的历史数据，还会使用其他变量的历史数据来预测某个变量的未来值。比如，我们可能会使用过去的销售额和广告支出的数据来预测未来的销售额。

多变量数据

月份	销售额	广告费用	产品数量	销售额
一月	1200	200	100	1200
二月	1400	220	120	1400
三月	1600	240	140	1600
四月	1500	130	130	1500
五月	1700	250	150	1700
六月	1800	260	160	1800
七月	2000	280	180	2000
八月	2200	300	200	2200
九月	2400	320	220	2400
十月	2600	340	240	2600
十一月	2800	360	260	2800
十二月	3000	380	280	3000

在处理多变量时间序列数据时，我们需要分析不同变量之间的相互影响，以及这些影响如何随时间变化。例如，我们可能会对广告费用增加是否会导致销售额增加感兴趣，或者是否存在销售额增加但产品数量减少的情况。通过理解这些关系，我们可以更好地预测未来的趋势，并做出更明智的商业决策。

1.3 时序模型 pk 机器学习

项目	单变量时间序列模型	传统机器学习模型
数据的维度	基于一个输入特征(X)来预测一个输出(Y)	基于一个输入特征(X)来预测一个输出(Y)
模型训练	使用历史数据来训练模型	使用历史数据来训练模型
数据的依赖性	数据点之间存在时间依赖性	数据点之间假设相互独立
输入特征的定义	输入特征(X)通常是前一步或前几步的值	输入特征(X)通常是预定义的，并且与时间无关
数据的划分	按时间顺序将数据划分为训练集和测试集	随机地划分训练集和测试集

总的来说，单变量时间序列建模与只有一个X和Y的机器学习建模都涉及到使用一个输入特征来预测一个输出，但由于时间序列数据的特性，它们在数据依赖性、输入特征的定义和数据划分方式上存在显著的区别。

所以通过上述的结论，我们也就能明白：为什么会出现一些适合于时间序列数据本身的算法，根本的原因还是在于传统时间序列数据在机器学习模型下的建模结果往往不尽如人意。

项目	时间序列模型	传统机器学习模型
学习过程	通过历史数据学习模式进行预测	通过历史数据学习模式进行预测或分类
评估指标	平均绝对误差（MAE）、均方根误差（RMSE）、精确度等	平均绝对误差（MAE）、均方根误差（RMSE）、精确度等
训练/测试拆分	将数据集拆分为训练集和测试集	将数据集拆分为训练集和测试集
数据结构	数据点之间存在时间依赖性	数据点之间假设相互独立
特征工程	提取趋势、季节性、周期性成分或使用滞后特征	创建新特征、删除不相关特征或对特征进行转换
模型类型	包括自回归模型（AR）、移动平均模型（MA）、ARMA、ARIMA、SARIMA等	包括随机森林，支持向量机，神经网络等
训练/测试拆分方式	按时间顺序拆分数据集	随机拆分训练集和测试集

1.3.1 时序建模规范

（1）划分训练集、测试集原则

- 时间顺序： 因为我们的目标是预测未来，所以在构建训练集和测试集时，我们不能使用未来的信息来预测过去。也就是说，我们需要确保训练集的所有数据在时间上都早于测试集的数

据。对单变量时序数据而言这是说样本顺序是绝对不能被打乱的，测试集的样本时间永远要晚于训练集

- 无重叠： 在实际场景中，通常不会有时间既在训练集又在测试集中出现。这就意味着训练集和测试集应该是互斥的，即他们在时间上没有交集。

覆盖日期	作用
1月-9月	训练集的真实标签
10月-12月	测试集的真实标签

（2）时序模型的训练方式

2.1 单步预测

将时间数据按照普通回归数据的方式进行训练和预测。即训练集、验证集和测试集是完全分割的三段时间，在训练集上训练、在验证集上调参，在测试集上输出最终预测结果。这种方式被称为“单步预测”。

使用历史数据来预测未来的一个时间步的值。这个“步”可以是任意时间单位，比如分钟、小时、天、月，等等。具体来说，我们只预测未来一个时间步长（例如，下一个小时、下一天、下一年等）的值，而不考虑其后的情况。

2.2 多步预测

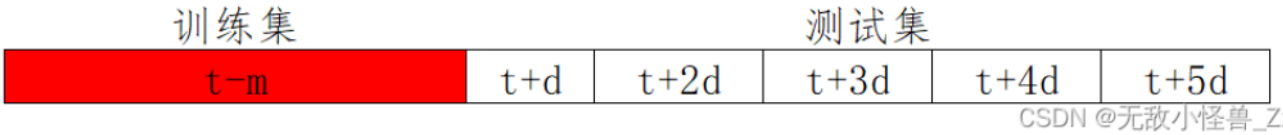
多步预测指的是使用历史数据来预测未来多个时间步的值。与单步预测只预测未来一个时间步长的值不同，多步预测将预测范围扩展到了未来多个时间步长。

例如，假设我们有一系列的历史销售数据，现在我们想要预测未来三天的销售量：

日期	销售量
1月1日	100
1月2日	120
1月3日	130
1月4日	110
1月5日	115

在多步预测中，我们对未来的多个时间点的预测结果都感兴趣。也就是说，我们关心的不仅仅是1月6日的销售量预测值，我们还关心1月7日和1月8日的销售量预测值。这就是多步预测的主要特点。

很明显，离现在越远的未来越难预测，因此应该要尽量压缩训练集与测试集之间的时间差，如：



将测试集上的时间分割为5段，假设t是当前的时间，先使用训练好的模型预测出t+d时间段的结果，将该结果加入训练集、构成全新的训练数据。即是说，我们将预测的值作为真实值加入到训练集中再对下一个单位时间进行预测，这样累加可以让训练数据的时间点与测试数据的时间点尽量接近。

缺陷也是很明显的。多步预测中可能会导致误差累加，如果在最开始预测时就存在很大的误差，那预测效果会变得越来越差且无法挽回。

1.4 时序算法总结

算法类别	算法名称	主要原理
经典统计方法	AR(自回归模型)	基于过去的p个时间点的观测值进行预测。
	MA(移动平均模型)	基于过去的q个时间点的误差项进行预测。
	ARMA(自回归移动平均模型)	结合了AR模型和MA模型，基于过去的p个时间点的观测值和过去q个时间点的误差项进行预测。
	ARIMA(自回归积分移动平均模型)	ARIMA模型是ARMA模型的推广，可以处理非平稳序列预测。
	SARIMA(季节性自回归积分移动平均模型)	SARIMA模型是ARIMA模型的推广，可以处理带季节性的非平稳序列预测。
	Holt-Winters模型	一种双指数平滑模型，可以处理具有趋势和季节性的时间序列数据。

机器学习	随机森林	随机森林是一种集成学习方法，通过集成多个决策树的预测结果来做出最终的预测。
	支持向量机	支持向量机通过在高维空间中找到最优超平面，实现对数据的分割或预测。

	神经网络	神经网络通过模拟大脑神经元连接的方式，进行信息处理，可以捕获和学习数据中的复杂模式。
深度学习	RNN(循环神经网络)	RNN能够处理序列数据，通过隐藏状态来捕获序列中的时间依赖关系。
	LSTM(长短期记忆网络)	LSTM是RNN的一种，通过增加一个记忆单元，解决了RNN处理长序列时的梯度消失问题。
	GRU(门控循环单元)	GRU是RNN的一种，结构比LSTM更简单，但在许多任务上的表现与LSTM相近。
	TCN(时序卷积网络)	TCN通过一维卷积，有效地处理序列数据。
	Transformer	Transformer通过自注意力机制有效地处理序列数据，广泛应用于NLP领域，也开始用于时序预测任务。

1.5 AR模型

时间序列模型(二)：AR模型_ar预测-CSDN博客

1.5.1 概念

AR模型，即自回归模型（Autoregressive Model），是一种被广泛用于分析时间序列数据的统计模型。

从严格的定义上来说：在统计学中，一个自回归模型的形式是描述某个变量与其过去值的关系的。对于一个时间序列数据，自回归模型的一阶形式可以表示为：

$$Y_t = c + \varphi Y_{t-1} + \xi_t$$

其中，Y_t 是在时间 t 的观察值，c 是常数，φ 是自回归系数，Y_{t-1}是在时间 t-1 的观察值，ξ_t是误差项。

当自回归模型的阶数为 p 时，模型可以表示为：

$$Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + ... + \varphi_p Y_{t-p} + \xi_t$$

其中，$\psi_1, \psi_2, ..., \psi_P$ 是自回归系数。

这一公式被我们称之为是p阶的自回归模型

- Y_t：表示在时间点t的时间序列值。也就是在时间点t时的标签（训练时这里使用真实标签，预测时这里输出预测标签）
- c：表示模型的常数项，也叫做截距项。在某些情况下，这个值可能被设置为0，使得模型不包含常数项。
- $\varphi_1 Y_{t-1}, \varphi_2 Y_{t-2}, ..., \varphi_p Y_{t-p}$：这些项表示过去的p个时间点的时间序列值对当前时间点的影响。其中，φ_i是第i个滞后项的系数，代表了第i个滞后项对当前时间点的相对影响力。Y_{t-i}则是在时间点t-i的时间序列值。可以这样理解：Y_{t-1}表示在时间点t的前一个时间点$t-1$时的标签值。我们可以根据不同的场景规定t与$t-1$之间具体的时间间隔大小，但在同一个时间序列中，t与$t-1$之间的间隔一定是等同于$t-n$与$t-(n-1)$之间的间隔
- ξ_t：这是在时间点t的误差项，也被称为白噪声项。在标准的AR模型中，最严格的情况下，只有均值为0、方差为特定σ^2、服从正态分布的才能被称之为白噪声

自回归模型，顾名思义，就是使用一个变量的过去值来预测其未来值的模型。简单来说，它就是假设你明天的状态，或者一个序列的下一项，会依赖于今天，或者前面几项的状态。例如，明天的天气可能会受到今天天气的影响，股票的明天价格可能会受到过去几天价格的影响。

自回归模型假设过去的值与未来的值之间存在线性关系。也就是说，它假设你能通过给过去的值加权求和来预测未来的值，这些权重就是模型的参数。

1.5.2 AR模型的前提假设

1.时序依赖性：在AR模型中，我们假设不同时间点的标签值之间存在强相关性。这意味着一个给定时间点的标签值受到其过去的标签值的显著影响。在数学上，这表现为两个时间点的标签值之间的相关系数较大。换言之，过去的信息对预测未来具有重要影响。

2.时序衰减：另一个基础假设是，两个时间点之间的距离越远，他们之间的关联性越弱。例如，昨天的天气可能对今天的天气影响很大，但三个月前的某一天的天气，对今天的天气的影响就相对微弱。

1.5.3 AR模型和多元线性回归

AR模型与多元线性回归模型在形式上非常相似，都可以看作是因变量对一组自变量的线性回归。线性回归：

$$y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_p x_p$$

- AR模型（自回归模型）主要应用于时间序列数据。它的基本假设是，一个时间点的观测值可以由其历史观测值线性表示。这意味着AR模型假设当前的时间序列值由过去的值以及一个误差项（通常假设为白噪声）决定。例如，今天的股票价格可能与过去几天的价格相关。
- 多元线性回归模型，与此不同，主要用于截面数据或面板数据，它假设因变量可以由一组自变量线性表示，自变量之间没有严格的时间顺序。例如，房价可能被建筑面积、卧室数量、地理位置等因素影响
- 在这两种模型中，时间的顺序和依赖性在AR模型中起着关键作用，而在多元线性回归中通常不考虑。

多元线性回归假设自变量之间互相独立，即不存在多重共线性。而在AR模型中，由于历史观测值之间可能存在时间依赖性，这一假设通常不成立。总的来说：AR模型与多元线性回归模型在形式上相似，但在实际应用和假设上有很大区别。选择哪种模型取决于你的数据类型和你的研究问题。

1.5.4

在自回归模型中每个自变量y 都是一个样本的数值，要求解的标签y也是一个样本的数值。

一个自回归模型只能得出一个样本的结果。因此，使用AR完成一个时间序列预测，是需要建立多个AR模型的。因此在自回归模型中，我们需要不断地建模来求解“下一个”时间点上的数值，以构成序列数据

AR(p)模型模型在时间区间[0,t]上进行训练，在时间区间[t+1,t+m]上进行预测，t 为现在的时间点，则有：

- **训练求解** φ:

$$
\begin{aligned}
Y_1 &= c + \varphi_1 Y_0 \\
Y_2 &= c + \varphi_1 Y_1 + \varphi_2 Y_0 \\
Y_3 &= c + \varphi_1 Y_2 + \varphi_2 Y_1 + \varphi_3 Y_0 \\
&\vdots \\
Y_t &= c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p}
\end{aligned}
$$

- **测试求解** $\{Y_{t+1}, Y_{t+2}, \ldots, Y_{t+m}\}$:

$$
\begin{aligned}
\hat{Y}_{t+1} &= c + \varphi_1 Y_t + \varphi_2 Y_{t-1} + \ldots + \varphi_p Y_{t-p+1} \\
\hat{Y}_{t+2} &= c + \varphi_1 \hat{Y}_{t+1} + \varphi_2 Y_t + \ldots + \varphi_p Y_{t-p+2} \\
\hat{Y}_{t+3} &= c + \varphi_1 \hat{Y}_{t+2} + \varphi_2 \hat{Y}_{t+1} + \ldots + \varphi_p Y_{t-p+3} \\
&\vdots \\
\hat{Y}_{t+m} &= c + \varphi_1 \hat{Y}_{t+m-1} + \varphi_2 \hat{Y}_{t+m-2} + \ldots + \varphi_p \hat{Y}_{t+m-p}
\end{aligned}
$$

- p 是自回归模型中的参数，它表示在预测当前时刻的值时，我们将考虑过去多少个时刻的值。
- 在预测过程中，我们有时需要使用实际值，有时需要使用预测值。

以 AR(3) 模型为例，假设我们正在预测从时间 t+1 开始的未来四个时刻的值（即预测 t+1, t+2, t+3 和 t+4 时刻的值）

预测的时刻	使用的真实数据	使用的预测数据
t+1	Y_t, Y_{t-1}, Y_{t-2}	-
t+2	Y_t, Y_{t-1}	$\hat{Y}_{t+1}$
t+3	Y_t	$\hat{Y}_{t+1}, \hat{Y}_{t+2}$
t+4	-	$\hat{Y}_{t+1}, \hat{Y}_{t+2}, \hat{Y}_{t+3}$

1.6 MA模型

时间序列模型(三)：MA模型_时间序列分析数据具有局部变方差特征时,可以考虑用什么进行建模-CSDN博客

1.6.1 概念

移动平均模型（MA模型）是时间序列分析中的一种模型，它描述的是当前时间点的数据与过去噪声的关系。严格定义上来讲：其模型的定义是基于白噪声序列的假设。白噪声是一种特殊的时间序列模型，每个时间点的数据都是独立且服从相同分布的，且具有常数的均值和方差。给定一个白噪声序列 ϵ_t ，MA模型定义为：

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

这一公式被称之为是q阶的移动平均模型，写作MA(q),其中，

- Y_t 是我们感兴趣的时间序列，在时间点t的观察值。
- μ 是时间序列的均值或期望值，这个值对所有的时间点都是相同的。在许多实际的时间序列分析中，我们假设时间序列已经通过某种方式（例如，通过差分或去趋势）转换为均值为零的序列。但在完整的MA模型中，这个均值项μ是存在的
- ϵ_t , $\epsilon_(t-1)$, $\epsilon_(t-2)$, …,$\epsilon_(t-q)$: 这些是所谓的白噪声项，每个时间点的值都是独立同分布的，通常假设为正态分布。这些项的均值为0，方差为σ²（一个常数）。ϵ_t 是当前时刻的白噪声，$\epsilon_(t-1)$是上一时刻的白噪声，依此类推，$\epsilon_(t-q)$是q个时刻前的白噪声。
- θ_1 , θ_2 , …, θ_q : 这些是MA模型的参数，每个参数θ都对应于一个白噪声项。它们衡量的是对应的白噪声对当前时间点的影响程度。
- q 是阶数，表示有多少个过去的白噪声项被纳入模型，指的是在模型中包含的过去白噪声项的数量。例如，如果q=2，那么模型就包含了$\epsilon_(t-1)$和$\epsilon_(t-2)$两个白噪声项。

这个公式的含义是：在时间点t，观察到的值Y_t是由过去q个时间点的白噪声的线性组合加上一个常数（即均值μ）和当前时间点的白噪声决定的。各个白噪声项的权重由参数θ决定。

通俗理解一下：移动平均模型的思想是，我们可以把一个时间序列看作是过去若干期噪声的加权平均，即当前的观察值是由过去的白噪声通过一定的线性组合得到的。

从定义上来看：MA模型是不同于AR模型的时序模型。MA模型的基本思想和基本假设与AR模型大不相同。MA模型的基本思想是：大部分时候时间序列应当是相对稳定的。在稳定的基础上，每个时间点上的标签值受过去一段时间内、不可预料的各种偶然事件影响而波动。即在一段时间内，时间序列应该是围绕着某个均值上下波动的序列，时间点上的标签值会围绕着某个均值移动，因此模型才被称为“移动平均模型 Moving Average Model”

- 均值稳定：时间序列的均值或期望值是恒定的，不随时间变化。这就是公式中的μ项，它对所有时间点都是相同的。这也是为什么会说“时间序列应该是围绕着某个均值上下波动的序

列”。在许多实际的时间序列分析中，我们可能需要通过一些预处理步骤（如差分或去趋势）将原始时间序列转换为均值稳定的序列。

- 方差稳定：时间序列的方差也是恒定的，不随时间变化。换句话说，时间序列的波动程度是一致的，不会在不同的时间点表现出明显的扩大或缩小。在MA模型中，这个特性主要由白噪声项ϵ_t来保证，因为白噪声的方差是常数。

无自相关：在理想的MA模型中，不同时间点的观察值之间没有自相关性。这意味着过去的值不能用来预测未来的值，除非你考虑到了白噪声项。这就是为什么会说“每个时间点上的标签值受过去一段时间内、不可预料的各种偶然事件影响而波动”。

1.6.2 AR模型和MA模型的区别

- 对于自回归（AR）模型来说：在这种模型中，当前值是过去值的函数。也就是说，我们是在使用过去的"实际"观察值来预测现在的值。AR模型的基本思想是过去的观测值会对未来的观测值产生影响，即未来的观测值是过去观测值的加权和。

- 对MA模型来说：在这种模型中，当前值是过去噪声（或称之为误差或冲击）的函数。这里的“白噪声”实际上是模型无法解释的随机部分，是未能被模型捕获的信息。换句话说，MA模型是在试图用过去的"错误"或"冲击"来预测现在的值。

1.6.3 MA模型的前提假设

移动平均模型（MA）的基本假设可以从以下几个方面来理解：

1.平稳性：MA模型假设时间序列是平稳的。这意味着序列的主要统计属性，如均值和方差，不随时间变化。这个假设强调了序列在长期内保持稳定的行为，而在短期内可能会受到随机因素的影响。

2.白噪声：MA模型假设存在一个白噪声序列。白噪声是随机误差项，它的均值为0，方差为常数，且各个时间点上的值是相互独立的。这个假设强调了在一段较短的时间内，时间序列的波动可能受到不可预测的随机因素的影响。

3.线性：MA模型假设时间序列可以被过去的白噪声项的线性组合表示。这就是模型被称为“移动平均”模型的原因，因为它的预测值是过去白噪声的加权平均。

4.有限历史影响：MA模型假设只有过去的q个白噪声才对当前时间点的值有影响，其中q是模型的阶数。换句话说，过去更久的白噪声对当前值没有直接影响。

5.标签值的关联性与白噪声的独立性：MA模型假设不同时间点的标签值之间是关联的，这反映了历史标签影响时间序列的长期趋势。而偶然事件在不同时间点上产生的影响（即白噪声）是相互独立的，这反映了在短期内，时间序列的波动可能受到不可预测的随机因素的影响。

影响明日会不会下雨的真正因素并不是“今天”或“昨天”这些时间概念本身，而是风、云、日照等更加客观和科学的因素（这些其实就是MA模型认为的“偶然因素”）。不过也能够理解，随着季节的变化、时间自有自己的周期，因此天气也会存在季节性的周期，因此从长期来看时间序列的趋势是恒定的。

1.6.4 MA模型的建模流程

对于一个MA(q)模型，其形式表示为：

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

我们可以分别讨论训练和预测阶段：

MA(q)模型在时间区间[0,t]上进行训练，在时间区间[t+1,t+m]上进行预测，t tt为现在的时间点，则有：

1. **训练求解** θ**：**

在训练阶段，我们需要估计参数θ和白噪声序列ϵ。这是一个复杂的过程，因为白噪声项是未知的。一般来说，我们可以通过最大似然估计等方法来估计这些参数。参数估计完毕后，我们可以得到模型的残差，它将作为未来预测的一部分。

$$\begin{aligned}
Y_1 &= \mu + \epsilon_1 + \theta_1 \epsilon_0 + \theta_2 \epsilon_{-1} + \ldots + \theta_q \epsilon_{(1-q)} \\
Y_2 &= \mu + \epsilon_2 + \theta_1 \epsilon_1 + \theta_2 \epsilon_0 + \ldots + \theta_q \epsilon_{(2-q)} \\
Y_3 &= \mu + \epsilon_3 + \theta_1 \epsilon_2 + \theta_2 \epsilon_1 + \ldots + \theta_q \epsilon_{(3-q)} \\
&\cdots \\
Y_t &= \mu + \epsilon_t + \theta_1 \epsilon_{(t-1)} + \theta_2 \epsilon_{(t-2)} + \ldots + \theta_q \epsilon_{(t-q)}
\end{aligned}$$

2. **测试求解** $\{Y_{t+1}, Y_{t+2}, \cdots, Y_{t+m}\}$:

对于预测阶段，我们需要知道过去的白噪声序列。然而，这些白噪声项通常是未知的，所以我们通常用训练阶段估计出的残差作为过去的白噪声来进行预测。预测公式如下：

$$\begin{aligned}
\hat{Y}_{(t+1)} &= \mu + \theta_1 \epsilon_t + \theta_2 \epsilon_{(t-1)} + \ldots + \theta_q \epsilon_{(t+1-q)} \\
\hat{Y}_{(t+2)} &= \mu + \theta_1 \epsilon_{(t+1)} + \theta_2 \epsilon_t + \ldots + \theta_q \epsilon_{(t+2-q)} \\
\hat{Y}_{(t+3)} &= \mu + \theta_1 \epsilon_{(t+2)} + \theta_2 \epsilon_{(t+1)} + \ldots + \theta_q \epsilon_{(t+3-q)} \\
&\cdots \\
\hat{Y}_{(t+m)} &= \mu + \theta_1 \epsilon_{(t+m-1)} + \theta_2 \epsilon_{(t+m-2)} + \ldots + \theta_q \epsilon_{(t+m-q)}
\end{aligned}$$

- AR模型中的预测是基于过去的观测值和预测值，而MA模型中的预测则是基于过去的误差项。

让我们以MA(3)模型为例，我们要预测从时间 t+1 开始的未来四个时刻的值（即预测 t+1, t+2, t+3 和 t+4 时刻的值）：

预测的时刻	使用的真实误差项	使用的预测误差项
t+1	$\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}$	-
t+2	$\epsilon_t, \epsilon_{t-1}$	ϵ_{t+1}
t+3	ϵ_t	$\epsilon_{t+1}, \epsilon_{t+2}$
t+4	-	$\epsilon_{t+1}, \epsilon_{t+2}, \epsilon_{t+3}$

在MA模型中，我们使用过去的误差项（ϵ）来预测未来的值。然而，真实的误差项是未知的，所以在实际预测中，我们使用过去的预测误差项。比如，对于t+2时刻的预测，我们使用的是t时刻和t-1时刻的真实误差项，以及t+1时刻的预测误差项。

1.6.5 如何求解ϵ 和θ

对于移动平均模型（MA），需要一次性求解出所有的θ，这部分是在模型训练时完成的。但对于误差项ϵ，情况稍微复杂一些。在模型训练时，实际上是通过最小化残差（也就是观察值和模型预测值之间的差异）的方式来进行的。这就意味着，训练过程实际上就是在估算出一系列的ϵ，使得预测值与观察值之间的差异最小。然而，对于预测阶段，ϵ 是未知的。通常，我们会假设在时间序列的未来部分，ϵ 是零。然后，我们就能够使用我们的MA模型（及其估计出的θ 参数）来进行预测。

在模型训练阶段，模型试图在给定观测数据的情况下，找出使模型误差最小的θ参数。在这个过程中，我们可以把ϵ看作是模型无法解释的部分，即模型的残差。模型训练阶段的目标就是找到一组θ参数，使得这些残差（即ϵ）尽可能的小。

然后，当模型训练完成，我们获得了θ 参数后，我们就可以用这个模型去预测未来的数据。但是，因为ϵ 是白噪声，也就是随机误差，所以在预测阶段我们无法准确知道未来的ϵ是什么。我们只能假设未来的ϵ的期望为0，因为这是白噪声的性质。

简单来说，θ是我们在训练阶段通过历史数据估计出来的，是模型的固定参数。ϵ 是模型的随机部分，在训练阶段我们可以计算其值，但在预测阶段，我们只能假设其期望值为0。

在MA(1)模型中，我们有这样的公式：

$$Y_t = \mu + \epsilon_t + \theta\epsilon_{t-1}$$

在这里，μ是常数项，ϵ_t是在时间t的白噪声误差项，ϵ_{t-1}是在时间$t-1$的白噪声误差项。

因为ϵ_t是白噪声序列，所以它们是不可预测的。但是，一旦我们有了数据Y_t，我们可以使用这个公式来解出ϵ_t，如下：

$$\epsilon_t = Y_t - \mu - \theta\epsilon_{t-1}$$

然后，我们可以使用这个新解出的ϵ_t来计算下一个时刻的Y_{t+1}：

$$Y_{t+1} = \mu + \epsilon_{t+1} + \theta\epsilon_t$$

这是一个递归的过程。一旦我们知道了模型的参数μ和θ，我们就可以从时间$t=1$开始，通过不断地解出ϵ_t和Y_{t+1}来进行预测。

需要注意的是，为了启动这个过程，我们需要知道初始的ϵ_0。在实践中，我们通常假设$\epsilon_0 = 0$，或者使用前一段时间的平均Y值作为ϵ_0。

而对于μ和θ，我们需要使用历史数据进行参数估计，常用的方法包括极大似然估计等。

1.7 差分过程（I）

时间序列模型(四)：ARIMA模型_arima模型表达式-CSDN博客

1.7.1 差分的概念

差分是一种数学操作，用于计算一组数值序列中相邻数据点的差值。在时间序列分析中，差分常用于将非平稳序列转化为平稳序列，也就是减小或消除时间序列的趋势和季节性变化。

当我们对一个序列进行差分运算，就意味着我们会计算该序列中的不同观测值之间的差异

简单地说，如果我们有一个时间序列Y_t,那么该序列的一阶差分就可以定义为：

$$\Delta Y_t = Y_t - Y_{t-1}$$

这样，我们得到一个新的时间序列，其每一个值都是原时间序列中相邻两个值的差。

假设我们有以下一组时间序列数据：

$$Y = 4, 8, 6, 5, 3, 4$$

我们可以看到，这个序列的长度是6。现在，我们希望对这个序列进行一阶差分。

第一步，我们计算第二个数据点和第一个数据点的差，也就是8-4=4。

第二步，我们计算第三个数据点和第二个数据点的差，也就是6-8=-2。

依次类推，我们计算出所有相邻数据点之间的差值，得到一个新的序列：

$$\Delta Y = 4, -2, -1, -2, 1$$

我们可以看到，差分后的序列比原序列短了一位，因为差分操作实际上计算的是原序列中的相邻数据点之间的差值。同时，差分后的序列相比于原序列，其趋势和季节性变化都得到了一定程度的消除。通常进行一次差分运算，原始的序列会变短1个单位。

在实际进行差分运算时，我们可以改变差分运算的两个相关因子来执行不同的差分：一个是差分的阶数（order），另一个是差分的滞后（lag）。

1.7.2 差分的阶数

二阶差分就是对一阶差分后的序列再次进行差分。如果我们有一个时间序列Y_t那么该序列的二阶差分就可以定义为：

$$\Delta^2 Y_t = \Delta(Y_t - Y_{t-1}) = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$$

这样，我们得到一个新的时间序列，其每一个值都是原时间序列中相邻两个值的差的差。

假设我们有以下一组时间序列数据：

$$Y = 4, 8, 6, 5, 3, 4$$

首先，我们进行一阶差分，就像我们之前讲解的，具体的计算步骤如下：

1. 计算第二个数据点和第一个数据点的差，也就是8-4=4。
2. 计算第三个数据点和第二个数据点的差，也就是6-8=-2。
3. 以此类推，我们计算出所有相邻数据点之间的差值。

所以，一阶差分的结果如下：

$$\Delta Y = 4, -2, -1, -2, 1$$

然后，我们对这个一阶差分序列进行二阶差分。同样，我们从头开始计算相邻数据点的差值：

1. 计算第二个数据点和第一个数据点的差，也就是(-2)-4=-6。
2. 计算第三个数据点和第二个数据点的差，也就是(-1)-(-2)=1。
3. 以此类推，我们计算出所有相邻数据点之间的差值。

所以，二阶差分的结果如下：

$$\Delta^2 Y = -6, 1, -1, 3$$

我们可以看到，二阶差分后的序列比一阶差分的序列又短了一位。实际上是需要对序列Y进行两次一阶差分，

因此，n阶差分就是在原始数据基础上进行n次一阶差分。在现实中，我们使用的高阶差分一般阶数不会太高。在ARIMA模型中，超参数d dd最常见的取值是0、1、2这些很小的数字。

1.7.3 滞后

滞后实际上是描述了时间序列数据点之间的时间差。

差分的滞后（lag）与差分的阶数完全不同。正常的一阶差分是滞后为1的差分（lag-1 Differences），这代表在差分运算中，我们让相邻的两个观测值相减，即让间隔为（lag-1）的两

个观测值相减。因此，当滞后为2时，则代表我们需要让相隔1个值的两个观测值相减。

在ARIMA模型中，我们经常需要计算滞后d期的时间序列数据。这就意味着我们需要查找在t时刻前d个时间单位的数据。

> 假设我们有以下一组时间序列数据：
>
> $$Y = 4, 8, 6, 5, 3, 4$$
>
> 如果我们想要计算这个时间序列的一阶滞后序列，我们只需要将原序列向右移动一个单位，然后删除掉移动后超出的数据点，具体操作如下：
>
> 1. 我们先将整个序列向右移动一个单位，得到 {_, 4, 8, 6, 5, 3, 4}。
> 2. 然后，我们删除掉移动后超出的数据点，得到滞后序列 {4, 8, 6, 5, 3}。
>
> 如果我们想要计算二阶滞后序列，我们可以按照同样的方式进行操作：
>
> 1. 我们先将一阶滞后序列再向右移动一个单位，得到 {_, _, 4, 8, 6, 5, 3}。
> 2. 然后，我们删除掉移动后超出的数据点，得到二阶滞后序列 {4, 8, 6, 5}。
>
> 通过以上的操作，我们可以得到任意阶的滞后序列。

滞后差分（多步差分） 滞后差分（Lag Differences）是在进行差分操作时，不是用相邻的观测值进行相减，而是用相隔一定数量（即滞后数量）的观测值进行相减。这种操作通常在时间序列具有周期性的情况下非常有用，例如，当我们处理的数据随季节有规律地波动或者随一周的时间有规律地波动时。

假设我们有一个时间序列：

$$X = [5, 4, 6, 7, 9, 12]$$

现在，我们想要计算这个时间序列的2步滞后差分（lag-2 Differences）序列。首先，我们让相隔1个值的两个观测值相减，具体操作如下：

1. 首先，我们对序列进行滞后2差分运算，即进行6-5、7-4、9-6、12-7的运算。
2. 最终我们得到的新的时间序列：

$$X_{lag2} = [1, 3, 3, 5]$$

通过这个例子，我们可以看出滞后差分的操作就是令序列中索引更大的值减去与其相隔（lag-1）个样本的索引更小的值。在实际操作中，我们可以根据数据的特性，选择合适的滞后阶数，来对数据进行滞后差分操作。

带滞后的差分也叫做多步差分，例如，滞后为2的差分就叫做2步差分。相比起平时不怎么使用的高阶差分，多步差分应用非常广泛。在时间序列中，标签往往具备一定的周期性：例如，标签可能随季节有规律地波动（比如在夏季标签值高、在冬季标签值较低等），也可能随一周的时间有规律地波动（比如在周末较高、在工作日较低等）

在时间序列中，标签往往具备一定的周期性：例如，标签可能随季节有规律地波动（比如在夏季标签值高、在冬季标签值较低等），也可能随一周的时间有规律地波动（比如在周末较高、在工作日较低等）。

差分运算可以消除数据中激烈的波动，因此可以消除时间序列中的季节性、周期性、节假日等影响.

一般我们使用滞后为7的差分消除星期的影响，而使用滞后为12的差分来消除月份的影响（一般这种情况下每个样本所对应的时间单位是月），我们也常常使用滞后4来尝试消除季度所带来的影响

在统计学中，差分运算本质是一种信息提取方式，其最擅长提取的关键信息就是数据中的周期性，和其他信息提取方式一样，它会舍弃部分信息、提炼出剩下的信息供模型使用。也因此，差分最重要的意义之一就是能够让带有周期性的数据变得平稳.

1.8 ARIMA模型

时间序列模型(四)：ARIMA模型_arima模型表达式-CSDN博客

- AR模型，即自回归模型，其优势是对于具有较长历史趋势的数据，AR模型可以捕获这些趋势，并据此进行预测。但是AR模型不能很好地处理某些类型的时间序列数据，例如那些有临时、突发的变化或者噪声较大的数据。AR模型相信"历史决定未来"，因此很大程度上忽略了现实情况的复杂性、也忽略了真正影响标签的因子带来的不可预料的影响。

- MA模型，即移动平均模型，可以更好地处理那些有临时、突发的变化或者噪声较大的时间序列数据。但是对于具有较长历史趋势的数据，MA模型可能无法像AR模型那样捕捉到这些趋势。MA模型相信"时间序列是相对稳定的，时间序列的波动是由偶然因素影响决定的"，但现实中的时间序列很难一直维持"稳定"这一假设。

1.8.1 ARIMA模型的基本思想

ARIMA模型全称为自回归差分移动平均模型（Autoregressive Integrated Moving Average Model）。ARIMA模型主要由三部分构成，分别为自回归模型（AR）、差分过程（I）和移动平均模型（MA）。

- AR部分用于处理时间序列的自回归部分，它考虑了过去若干时期的观测值对当前值的影响。

- I部分用于使非平稳时间序列达到平稳，通过一阶或者二阶等差分处理，消除了时间序列中的趋势和季节性因素。

- MA部分用于处理时间序列的移动平均部分，它考虑了过去的预测误差对当前值的影响。

ARIMA模型的基本思想是利用数据本身的历史信息来预测未来。一个时间点上的标签值既受过去一段时间内的标签值影响，也受过去一段时间内的偶然事件的影响，这就是说，ARIMA模型假设：标签值是围绕着时间的大趋势而波动的，其中趋势是受历史标签影响构成的，波动是受一段时间内的偶然事件影响构成的，且大趋势本身不一定是稳定的

ARIMA模型既可以捕捉到数据的趋势变化，又可以处理那些有临时、突发的变化或者噪声较大的数据。

1.8.2 ARIMA模型的数学表达式

AR和MA模型的数学表达式：

$$AR:\ Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + ... + \varphi_p Y_{t-p} + \xi_t$$

$$MA:\ Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

如果我们暂时不考虑差分（即假设d=0），那么ARIMA模型可以被看作是AR模型和MA模型的直接结合，形式上看，ARIMA模型的公式可以表示为：

$$Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

在这个公式中：

- Y_t 是我们正在考虑的时间序列数据。
- φ_1 到φ_p 是AR模型的参数，这些参数用来描述当前值与过去p个时间点值之间的关系。
- θ_1 到θ_q 是MA模型的参数，这些参数用来描述当前值与过去q个时间点的误差之间的关系。
- ε_t 是在t时间点的误差项。
- c是一个常数项。

> 这个公式基本上是将AR模型和MA模型的公式组合在一起：
>
> - AR部分（即$\varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p}$）表示当前值$Y_t$与它过去的值有关，这个部分的形式与AR模型的公式一致。
> - MA部分（即 $\theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$）表示当前值$Y_t$与它过去的误差项有关，这个部分的形式与MA模型的公式一致。

值得注意的是，MA模型中代表长期趋势的均值μ 并不存在于ARIMA模型的公式当中，因为ARIMA模型中"预测长期趋势"这部分功能由AR模型来执行，因此AR模型替代了原本的μ 。在ARIMA模型中，c可以为0。

另外，这个公式的基础是假设我们正在处理的时间序列是平稳的，这样我们可以直接应用AR和MA模型。如果时间序列是非平稳的，那么我们就需要考虑ARIMA模型中的I部分，也就是进行差分处理。

1.8.3 ARIMA(p,d,q)模型的参数

arima模型原理及实战-CSDN博客

大致思路就是：1，当前时刻点的取值与之前相邻一个或者多个时刻点自相关，并假设他们满足一个线性关系。2，由于每一个时刻点采集的数据都存在服从一定分布的误差，且当前时刻不仅由自

己的采集误差，也与之前一个或者多个时刻的误差相关，并假设他们满足一个线性关系。

在 ARIMA(p, d, q) 模型中：

- **p 代表 “自回归部分 (Autoregressive)”**：这部分描述了模型中使用的观测值的滞后值（即前面 p 个期的值）。自回归模型的出发点是认为观测值是它前面的 p 个值的线性组合。具体的数学形式如下：

$$AR:\ Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + ... + \varphi_p Y_{t-p} + \xi_t$$

其中，$\varphi_1, \varphi_2, ..., \varphi_p$ 是模型参数，c 是常数，ξ_t 是白噪声。这个方程的阶数 p 决定了模型回溯观测值的数量。

- **q 代表 “移动平均部分 (Moving Average)”**：这部分描述了模型中使用的错误项的滞后值（即前面 q 个期的值）。移动平均模型是将当前值和过去的白噪声之间建立关系。具体的数学形式如下：

$$MA:\ Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

其中，$\theta_1, \theta_2, ..., \theta_q$ 是模型参数，c 是常数，e_t 是当前时期的白噪声，$\epsilon_{t-1}, \epsilon_{t-2}, ..., \epsilon_{t-q}$ 是过去的白噪声。这个方程的阶数 q 决定了模型回溯白噪声的数量。

因此，ARIMA 模型将自回归模型（AR）和移动平均模型（MA）结合在一起，同时加入了差分（I）这个操作。而 p, d, q 这三个参数，分别代表了模型中的自回归部分、差分阶数、以及移动平均部分。

在进行ARIMA模型拟合前，我们需要先通过画图或者ADF检验等方式，确定最小的d使得数据平稳。在确定了d之后，我们就可以将d阶差分后的序列代入模型进行拟合。

d就是差分的阶数。差分的目标是将非平稳序列转变为平稳序列。具体的数学表达如下：

滞后运算是“向后移动一个单位”的运算，当用于时间序列时，它特指“向过去移动一个时间单位”的运算。大部分时候，滞后运算被简写为字母B（Backshift）或者字母L（Lag），我们可以对单一的时序样本或整个时间序列做滞后运算。假设有一时间序列y_t，定义滞后运算（lag operator）B，它将一个时刻的观测值转化为前一时刻的观测值：

$$By_t = y_{t-1}$$

我们可以扩展这个运算符的概念，使之滞后n个时间步长。例如：

$$B^n y_t = y_{t-n}$$

此外，我们可以利用滞后运算来表示一阶差分和n阶差分。一阶差分可以看做是相邻的标签值之间的差，它可以表示为：

$$\Delta y_t = y_t - y_{t-1} = y_t - By_t = (1-B)y_t$$

类似的，n阶差分就是相隔n-1个标签值进行相减：

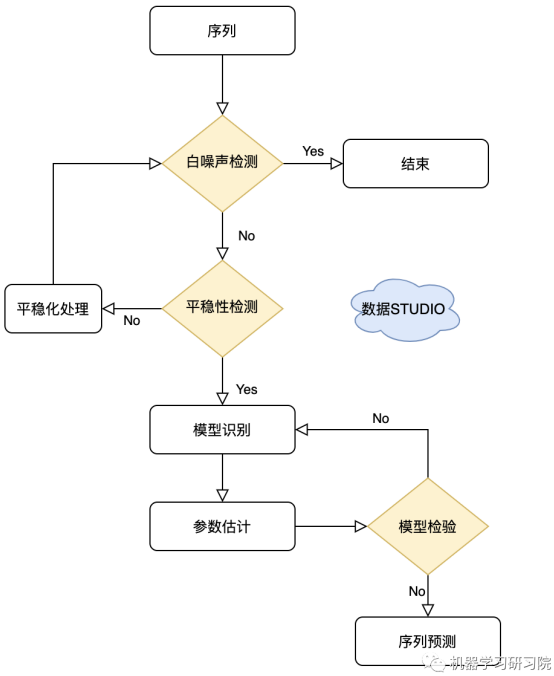
$$\Delta^n y_t = y_t - y_{t-n} = y_t - B^n y_t = (1-B^n)y_t$$

这个式子告诉我们，如果我们想对一组数据进行n阶差分，那么我们就可以使用滞后运算。

在实际使用中，我们经常将多步差分和高阶差分混用，最典型的就是在ARIMA模型建模之前：一般我们会先使用多步差分令数据满足ARIMA模型的基础建模条件，再在ARIMA模型中使用低阶的差分帮助模型更好地建模。例如，先对数据进行12步差分、再在模型中进行1阶差分，这样可以令数据变得平稳的同时、又提取出数据中的周期性，极大地提升模型对数据的拟合精度。

1.8.4 arima模型原理及实战

一般步骤 ① 首先需要对观测值序列进行平稳性检测，如果不平稳，则对其进行差分运算直到差分后的数据平稳； ② 在数据平稳后则对其进行白噪声检验，白噪声是指零均值常方差的随机平稳序列； ③ 如果是平稳非白噪声序列就计算ACF（自相关系数）、PACF（偏自相关系数），进行ARMA等模型识别； ④ 对已识别好的模型，确定模型参数，最后应用预测并进行误差分析。



ARIMA模型只能对平稳序列进行分析，无法对非平稳序列进行分析。所以拿到一个时间序列之后，首先需要进行平稳性检验和白噪声检验。

数据大概长这样：

	load
2014-12-30 00:00:00	0.33
2014-12-30 01:00:00	0.29
2014-12-30 02:00:00	0.27
2014-12-30 03:00:00	0.27

- **平稳性检验**

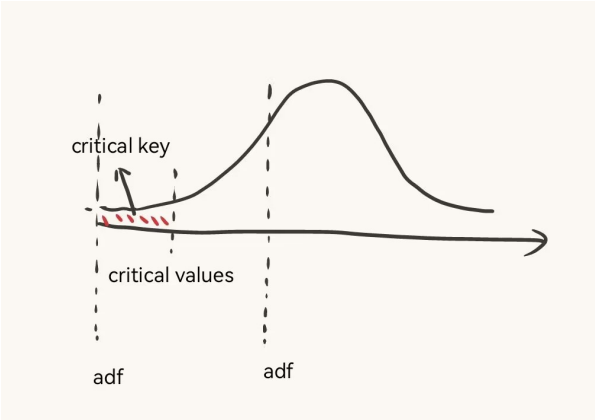
平稳性就是指记录的某个特征变量如果满足正态分布U(0,1),那么在任意时刻记录的值都服从Z(0,1)，而不是Z(0,2)。**一般使用ADF单位根进行平稳性检验。**

ADF是假设检验的应用。H0原假设：是序列不平稳，即存在单位根；H1备选假设：是序列平稳，不存在单位根。我们需要做的就是找证据来推翻原假设，如果证据不足无法推翻我们就能够避免用ARIMA去分析序列这种错误做法。而且我们设置了显著性水平α，如果我们找到了证据推翻了原假设，我们犯错的概率(第一类的弃真错误)也是很小的。

判断依据：ADF检验设计了一个ADF统计量，具体服从什么分布不清楚，一般默认t分布(用样本标准差代替总体方差)或者z分布。他是双侧检验，给定显著性水平之后，我们就能够知道拒绝域了，拒绝域的横坐标对应着相应的统计量计算值，只要计算出的ADF统计量在指定横坐标外侧我们就能够拒绝原假设，接受备选假设，即我们找到了证据证明序列不是不平稳的。

p值是比统计量更强的证据，假如统计量服从正态(0,1)分布，且拒绝域的横坐标是-2.56和2.56，但是我们计算出来的ADF统计量是3，那么p值就是X>3的概率值。其值越小越好，小于显著性水平α即可认为原假设不对，接受备选假设，即序列非平稳。

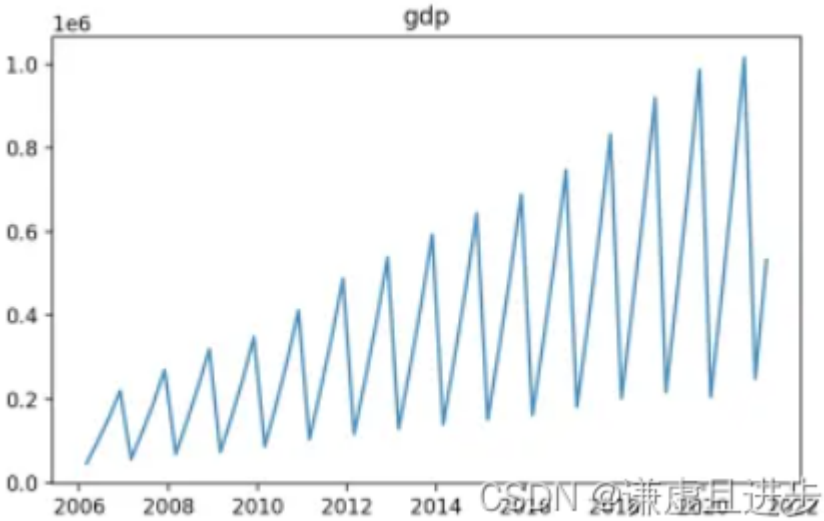
Augmented Dickey-Fuller (ADF)检验结果显示，测试统计量（ADF Statistic）为result[0]，这个值大于所有的临界值（在1%，5%和10%的显著性水平下，临界值分别为value1，value2和value3）。这表示我们不能拒绝原假设，即我们不能断定数据是平稳的。




```
from statsmodels.tsa.stattools import adfuller

result = adfuller(data)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
#ADF：float，测试统计
#pvalue：float，probability value：MacKinnon基于MacKinnon的近似p值
#critical values：dict，测试统计数据的临界值为1%，5%和10%。
```

将数据使用plt画出来，用眼睛看，如果趋势变化明显，随着时间前进波动幅度变大，一般都是非平稳。下面这种随着时间前进，最大值和最小值的距离越来越多，肯定非平稳。



使用ACF图和PACF图进行眼睛看，如果没有出现截尾或者拖尾，那肯定是非平稳。

经过检验序列非平稳，可以使用差分让数据变为平稳，一般只进行一阶差分，如果一阶差分序列还是非平稳序列，那么最好的做法是：赶紧换个模型(比如LSTM等神经网络模型)，别死磕ARIMA。

- 差分

使用pandas的diff()函数进行差分。这个函数会计算每一行与其前一行的差值。

```
# 首先导入了所需要的库和函数
from statsmodels.tsa.stattools import adfuller

# 定义一个名为calculate_diff的函数，接收三个参数：df是要处理的DataFrame，max_diff是最大的
差分步数，significance_level是判定平稳性的显著性水平
def calculate_diff(df, max_diff, significance_level=0.05):
    # 初始化最佳差分阶数和最小p值
    best_diff = None
    min_pvalue = 1.0
    min_variance = float('inf')  # 初始化最小方差
    min_adf_stat = float('inf')  # 初始化最小ADF统计量

    # 循环，差分阶数从1到max_diff
    for i in range(1, max_diff+1):
        # 对数据进行差分，并去除NA值
        df_diff = df['Sales'].diff(i).dropna()
        # 对差分后的数据进行ADF单位根检验
        result = adfuller(df_diff)
        # 打印出差分阶数，ADF统计量，p值，标准差和方差
        print(f'{i}步差分')
        print('ADF统计量: %f' % result[0])
        print('p值: %.10e' % result[1])
        print('标准差: %f' % df_diff.std())
        print('方差: %f' % df_diff.var())

        # 判断p值是否小于显著性水平，如果小于则认为差分后的数据可能是平稳的
        if result[1] < significance_level:
            print('=> 根据这个差分阶数，序列可能是平稳的')
            # 判断当前的p值是否小于最小p值，如果小于则更新最小p值和最佳差分阶数
            if result[1] < min_pvalue:
                min_pvalue = result[1]
                best_diff = i
                min_variance = df_diff.var()  # 更新最小方差
                min_adf_stat = result[0]  # 更新最小ADF统计量
        else:
            print('=> 根据这个差分阶数，序列可能是非平稳的')
        print('--------------------------------')

    # 如果找到了使数据平稳的差分阶数，打印出最佳差分阶数和其对应的p值
    if best_diff is not None:
        print(f'最佳差分阶数是: {best_diff}，p值为: {min_pvalue}，方差为:
{min_variance}，ADF统计量为: {min_adf_stat}')

# 使用函数对数据进行差分并测试其平稳性
calculate_diff(df, max_diff=24)
```

- **白噪声检验**

白噪声是指序列是纯随机数据，纯随机游走，是无意义的数据，继续研究毫无意义，比如使用random生成的服从正态分布的一串时间序列就属于白噪声。白噪声检验也称为纯随机性检验，当数据是纯随机数据时，再对数据进行分析就没有任何意义了。常使用acorr_ljungbox函数对序列进行白噪声检验。

```
# 数据的纯随机性检验函数
acorr_ljungbox(x, lags=None, boxpierce=False,
               model_df=0, period=None,
               return_df=True, auto_lag=False)
```

lags表示要检验的滞后阶数

boxpierce为True时表示除开返回LB统计量还会返回Box和Pierce的Q统计量

返回值

lbvalue: (float or array) 测试的统计量

pvalue: (float or array) 基于卡方分布的p统计量

bpvalue: ((optionsal), float or array) 基于 Box-Pierce 的检验的p统计量

bppvalue: ((optional), float or array) 基于卡方分布下的Box-Pierce检验的p统计量

```
from statsmodels.stats.diagnostic import acorr_ljungbox  # 随机性检验库
# 数据的纯随机性检验函数
lbvalue, pvalue=acorr_ljungbox(data,lags=1)
#lags=1:表示使用的滞后阶数为1，即检验序列中1阶滞后的自相关系数是否显著不为0
```

acorr_ljungbox的原理：原假设H0：是白噪声；备选假设H1：是非白噪声。

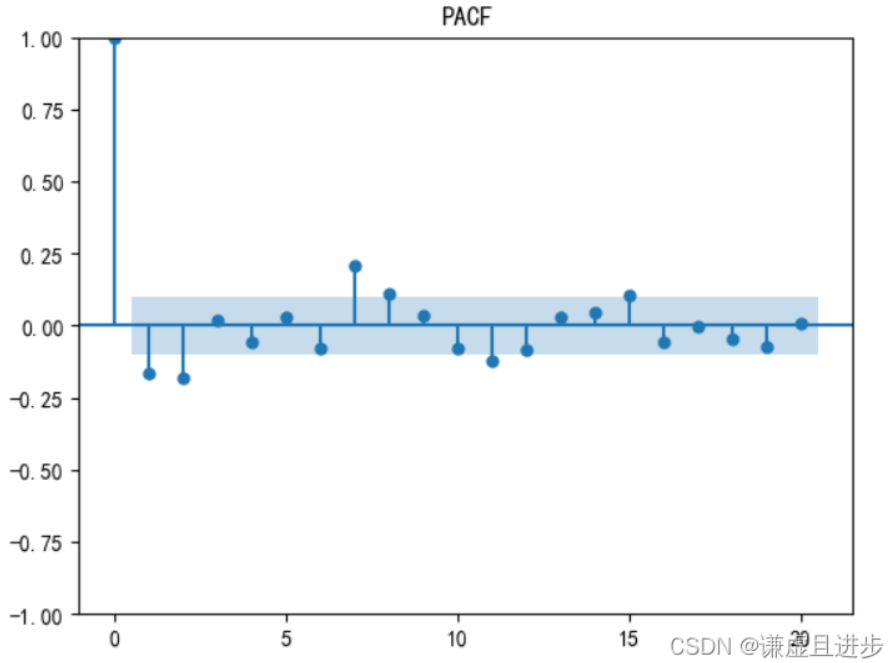
lbvalue:表示Ljung-Box统计量，用于检验序列的自相关性异否存在显著性。
pvalue:表示统计量的p值，用于判断序列是否为白噪声序列。如果p值小于显著性水平（道常为0.05），则拒绝原假设，即序列不是白噪声序列。 反之，则无法拒绝原假设，即序列为白噪声序列。

- **模型定阶，确定P和Q**

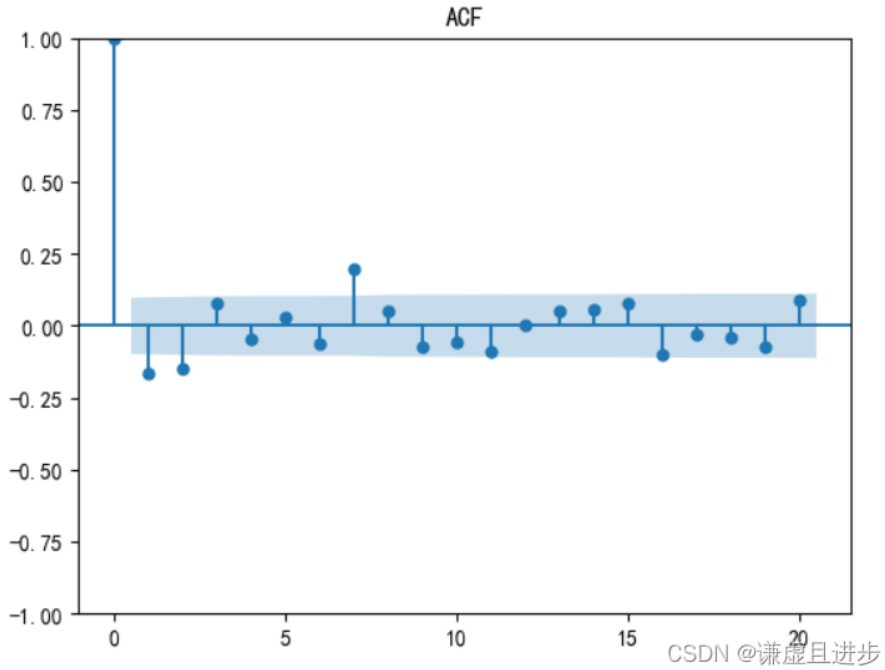
  如果原序列就是平稳的，定阶就使用原序列数据；如果进行了一阶差分变成了平稳序列，那么就使用一阶差分序列进行定阶。

方法1：根据平稳序列的PACF定P，ACF定Q

PACF的后截尾阶数就是P；ACF的后截尾阶数就是Q。PACF图中，最左边的相关系数是1，原因是自己跟自己的相关性当然是1。然后与其前1个、2个时刻的系数分别是-0.25左右，但是前第7个时刻由存在0.25左右的相关性。蓝色区域表示他们虽然存在相关性，但是可以忽略不计。



ACF图也类似，说明存在一定的周期相关性。



对任意时间序列，当ACF图像和PACF图像都不呈现拖尾状态时，无论图像是否截尾，时间序列都适用于ARIMA模型，且此时ACF和PACF图像无法帮助我们确定p和q的具体值，但能确认p和q一定都不为0。

方法2：暴力求解法

AIC和BIC定阶方法。由于我们已经得到了d的取值，要么0要么1。即P、Q、d我们知道了d，那么直接暴力遍历P、Q的取值，看一下那种组合下的AIC和BIC取值最小，我们就用那种组合。

```
import statsmodels.api as sma

# AR最大阶不超过6，MA最大阶不超过4。
aic_res= sma.tsa.arma_order_select_ic(diff1,max_ar=6,max_ma=4,ic=['aic'])
 # AIC
print('AIC', aic_res['aic_min_order'])

bic_res =sma.tsa.arma_order_select_ic(diff1,max_ar=6,max_ma=4,ic=['bic'])
 # BIC
print('BIC', bic_res['bic_min_order'] )
```

- 残差检验

  通过上述过程，确定P、Q、d之后就可以拿数据拟合模型了，注意：如果进行了差分拟合数据得是差分数据才行。

模型残差检验是想弄清楚模型是否完全的将数据特征提取完毕。可以这么认为：模型残差是白噪声，我们就认为序列特征已经被提取完毕。

```
modelf = sma.tsa.ARIMA(diff, order=(3, 4, 1))  # 传入参数，构建并拟合模型
result = modelf.fit()

# Ljung-Box检验
xx = acorr_ljungbox(result.resid, lags=10)
print(xx)
```