时 间 序 列 分 析
课 程 小 论 文

题　　目　任天堂：从谷底到颠峰

班　　级　信计 1901

专　　业　信息与计算科学

学　　号　1131190111

姓　　名　唐川淇

日　　期　2022 年　**6** 月　**11** 日

任天堂：从谷底到颠峰

唐川淇*

江南大学 信计 1901 班 学号 1131190111

【摘　要】 任天堂有限公司是一家日本的跨国视频游戏公司，它开发视频游戏和视频游戏机。本文收集了任天堂从 2015 年 10 月 28 日到 2022 年 5 月 27 日的股票数据，首先对其进行了基本统计分析，发现其在经历了下滑之后在 2016 年末到达谷底，之后强势反弹，虽然仍有波动，但是在之后的时间里保持了增长的趋势。本文首先通过线性模型 ARMA 和 ARIMA 对数据建模，发现模型的效果并不理想。然后考虑了数据可能具有的季节性，使用了三参数指数平滑模型和 ARIMA 季节模型对数据建模。考虑到数据可能异方差，所以还使用了异方差模型对股票数据建模。其中季节模型和异方差模型通过了模型显著性检验和参数显著性检验且对数据的预测准确率都较高。另外，本文还使用了现在流行的机器学习方法对数据建模，包括 MLP、LST 和 CNN，经过计算其中 LSTM 的预测效果最好。

【关键词】 **季节模型，异方差模型，时间序列分析，机器学习**

目录

1 引 言

由于投资回报率低，投资银行等传统投资机构对投资者不再具有吸引力[1]。股票市场现在是金融系统最重要的组成部分之一。当今社会，股票投资越来越受欢迎。在相对较短的时间内增加资金并获得通常高于平均水平的回报的愿望吸引了我们几乎所有人。反过来，提供股票的公司会获得用于自身发展的资金。然而，这往往是一个长期的过程，投资的资金可能需要几十年才能恢复。在某些情况下，投入的资本不会除了亏损之外，什么也得不到。因此，每一步都需要仔细考虑，购买股票时必须适当考虑。

任何进入金融市场的投资者都希望能够预测购买多少股票以及何时购买。然而，股票的数量也取决于投资者资产的价值。对投资基金进行后续估值的下一个重要步骤是对股票价格的历史发展以及我们想要购买其股票的公司进行适当的分析和评估。这包括其营业额、将新产品推向市场或其整体经济状况。有许多因素会影响后续价值的增加或减少。如果我们成功预测公司股票价格的发展，未来的价格可能会产生巨大的利润。这种感觉和正确预测的能力将成功的投资者与失败的投资者区分开来。

对股票的兴趣实际上决定了它们的价值。股票价格是在供求关系的基础上发展起来的。与市场上的任何商品一样，库存需求越高，供应越少，其价格就越高。供需因各种因素而变化；因此，股票价格也会随着时间而变化。可以使用多种方法和数据方法预测和评估这些因素。所谓的时间序列是检验随时间变化的动态和发展的最具决定性的统计方法之一。股票价格发展的时间序列对于评估影响股票价格发展并决定其过去表现的因素以及预测其未来发展至关重要。

2019 冠状病毒疾病的爆发对全世界产生了深远的影响。一切都变得严峻，使每个领域的条件更加不利。在这种情况下，大量公司受到影响，许多人失业，大量公司被迫关闭。此外，其他生存企业的原计划也被迫改变。

本文是根据 2015 年至 2022 年的数据，分析和评估任天堂公司股票价格的历史发展。

2 文献研究

任天堂（Nintendo）是一家主要从事电子游戏的开发、制造与发行的日本百年公司。于 1889 年在日本京都市创立，如图，其最初以生产花札起家，1970 年代后期投入电子游戏产业，在掌上游戏机 “Game & Watch” 与街机游戏《大金刚》获取商业成功后，于 1983 年推出家用游戏机 “Family Computer” 与在 1985 年推出游戏《超级马力欧兄弟》亦获取空前的成功，且旗下有多个著名游戏 IP，包括塞尔达传说、宝可梦和动物森友会等，为世界知名的电子游戏主机与软件开发商，亦是世界目前游戏机三大生产商之一。

正是因为任天堂公司活跃于金融市场，公司必须拥有完善的财务管理和决策支持系统。Khan 等人[1]认为，基于知识的财务管理决策支持系统是投资计划的重要组成部分。他们得出的结论是，由于投资回报率低，投资者避免投资银行等传统投资机构。

目前，股票市场是主要的投资领域之一。预测股票价格正成为日内交易者、投资者和数据科学家的一项艰巨任务。这些是影响价格变动动态的各种相互作用因素的复杂函数。政治、社会前景、公司销售和社会经济因素会影响全球对股票的看法，而这会受到供求平衡的影响。[2]也证实了这一点，他们补充道，股价和金融市场通常基于情绪，这导致研究人员利用 Facebook 和 Twitter 等社交网络上表达的公众情绪预测股市趋势。

股票价格的时间序列对于预测股票价格至关重要。时间序列分析是预测股票价格的基本方法，也是应用最广泛的研究方法[3]。然而，这种方法并不完全准确，因为它不包括可能影响股价发展的外部因素，如新闻、事件等。Rajesh 和 Gandy[4]专注于时间序列预测，引入了 CashTagNN 系统，该系统利用推特（包括 cashtags）的情绪和主观性得分来模拟股市走势，尤其是预测开盘股价。在这个系统中，他们使用了两种机器学习方法，即前馈神经网络和深度卷积神经网络。Ebadati 和 Mortazavi[5]也使用了神经网络的方法，应用遗传算法（GA）和人工神经网络（ANN）的混合方法开发了一种预测股票价格和时间序列的方法。在 GA 方法中，将输出值进一步转换为开发的 ANN 算法，以纠正精确点处的错误。分析表明，GA 和 ANN 可以通过较少

的迭代次数来提高精度。Gandhmal 和 Kumar[6]介绍了几种预测股市趋势的技术。作者进行了一项分析，描述了文档中使用的软件工具，这些工具也涉及价格预测。根据他们的研究，领先的软件工具包括 JavaScript、Python 和 Matlab。Vochozka、Horak 和 Krulicky[7]也同意，Matlab 系统是进行股价预测最常用的软件，并使用神经网络方法进行研究。Horak 和 Krulicky[8]旨在比较时间序列的指数平滑方法和使用神经网络作为工具预测公司未来价值发展的时间序列平滑方法。首先，他们分析数据集，然后生成神经网络，保留五个最佳特征。

图 1 位于京都的任天堂总部

3 数据基本统计分析

从雅虎财经网站，我搜集到了从 2015 年 10 月 28 日到 2022 年 5 月 27 日之间的任天堂股票数据，收集到的部分数据如表1。

表 1 部分任天堂股票数据

Date	Open	High	Low	Close	Adj C
2022/5/19	56.98	58.15	56.98	57.59	57.59
2022/5/20	58.24	58.24	57.09	57.63	57.63
2022/5/23	57.88	58.86	57.88	58.36	58.36
2022/5/24	58.36	58.86	58.04	58.40	58.40
2022/5/25	55.52	56.54	55.52	56.45	56.45
2022/5/26	56.20	57	56.20	56.79	56.79
2022/5/27	55.65	56.20	55.65	56.09	56.09

open，high，low 和 close 分别代表开盘价、最高价、最低价、收盘价。开盘价又称开市价，是指某种证券在证券交易所每个交易日开市后的第一笔每股买卖成交价格。收盘价是指某种证券在证券交易所一天交易活动结束前最后一笔交易的成交价格。如当日没有成交，则采用最近一次的成交价格作为收盘价。四种价格可以用 OHLC 图表示，如图2所示。adj close 是调整后收盘价，调整后的收盘价修正了股票的价格收盘价在对任何公司行为进行会计处理后，反映该股票的价值。另外数据中的 volumeV 是成交量指标，成交量是指个股或大盘的成交总手。

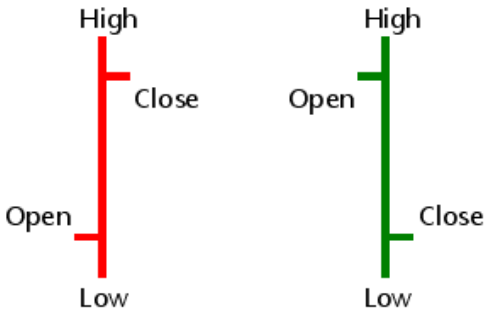


图 2 OHLC 图

3.1 描述性统计

求出数据的最小值、分位数、中位数、均值、最小值如表2。

表 2 描述性统计

最小值	均值	最大值
15.49	46.24	82.15
1/4 分位数	中位数	3/4 分位数
33.9	46.6	56.9

从数据基本统计量可以看出，数据最小值与最大值相差较大，中位数、四分之一分位数以及四分之三分位数均更接近于最小值，因此可以得知数据存在一定的偏态分布。为了更直观地观察数据特征，绘制直方图与核密度图如图3

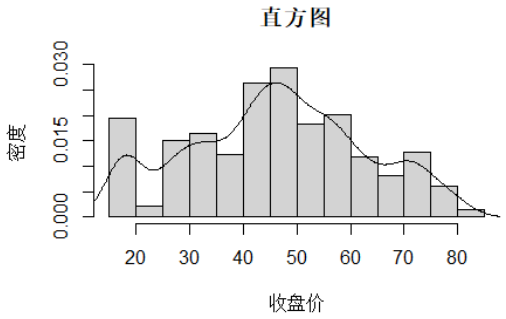


图 3 直方图

从直方图与核密度图可以看出，数据大多分布在 (30,60) 的区间内。整体略微成左偏分布。

3.2 数据可视化

3.2.1 原始序列

根据所收集的数据绘制出每日收盘价格的时序图，如图4所示。

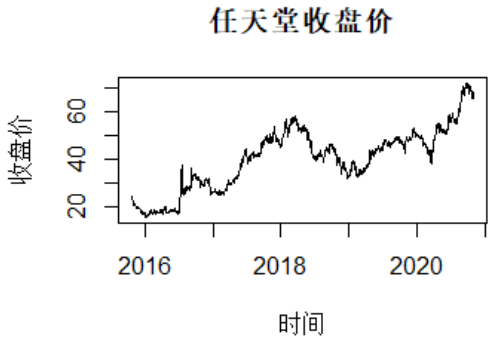


图 4 收盘价时序图

观察任天堂股票时间序列的基本统计数据，发现数据整体呈现上升趋势，但是在 2018 年出现了较大的下降，在 2020 年后半段出现的快速的增长，然后在 2021 年末由出现了下降。我发现数据均值不是零，并且可以从图中看出具有很高的方差。这表明时间序列是非平稳的，均值和方差都在变化。为了验证时间序列不平稳，绘制时间序列的 ACF 图, 如图56所示。

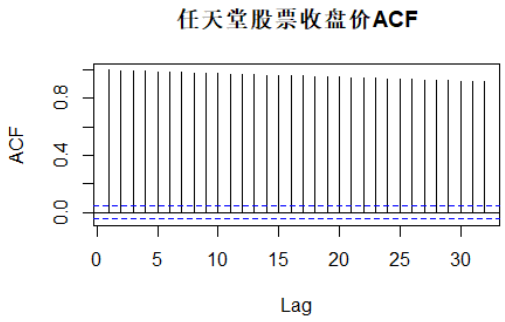


图 5 任天堂股票收盘价 ACF

在数据分析中，相关图是相关统计的图表。例如，在时间序列分析中，样本自相关图 r_h 相对 h（时间滞后）是一个自相关图。如果绘制了互相关，则结果称为互相关图。相关图是检查数据集中随机性的常用工具。如果是随机的，对于任何和所有时滞分离，自相关应该接近于零。如果非随机，则一个或多个自相关将显着非零。可以从图中看出，自相关图中一直位于两倍标差之外，可以认为该自相关系数很大，显著非零，说明该序列自相关系数与有长期相关性。另外画出每日收盘价格的偏自相关图，如图6所示。

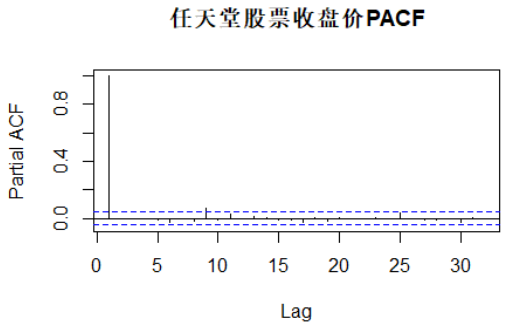


图 6 任天堂股票收盘价 PACF

在时间序列分析中，偏自相关函数 (PACF) 给出了平稳时间序列与其自身滞后值的部分相关性，回归了所有较短滞后时间序列的值。它与不控制其他滞后的自相关函数形成对比。从偏自相关图看出一阶偏自相关系数很明显地大于倍标准差范围，自一阶偏自相关系数后，其余偏自相关系数都在 2 倍标准差范围以内，且一阶后偏自相关系数衰减为在零附近小值波动的过程非常突然，可以判断偏自相关图为截尾。

3.2.2 处理后的序列

可以从 ACF 图以及 PACF 图中看出该序列不平稳。因此，为了使这个序列稳定，我研究了股票价格的取对数后的一阶差分序列。数据可以表示为 $z_n^*, n = 1, 2 \cdots N$，对数据进行对数处理后进行一阶差分操作，可以得到公式1，处理后的序列可以用来表示任天堂了股票价格的日收益。

$$y_n^* = log(z_n^*) - log(z_{n-1}^*) \tag{1}$$

根据公式处理原始数据之后得到了任天堂了股票价格的对数收益，绘制出该序列图如图7所示。

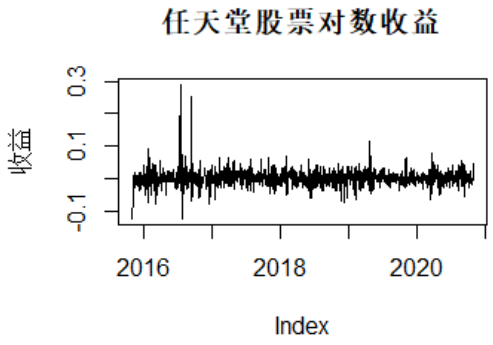


图 7 股票价格的收益

可以从图中看到图中数据基本在 0 附近浮动。在 2016 年后半段时间股票收益发生了剧烈的震荡，在短暂的震动之后任天堂股票的收益基本在一定范围内保持稳定。可以做出该序列的平方收益，如图8所示。

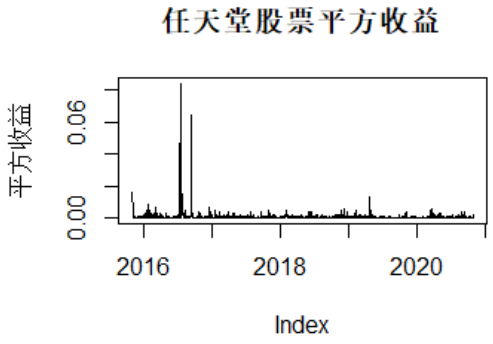


图 8 股票价格的平方收益

从该图也可以同样验证刚才的结论，在 2016 年后半段股票的收益出现了很大的浮动，其中最高的平方数值超过了 0.06，而全年其他时段的平方数据接近 0。另外可以做出该序列的绝对值收益，如图9。

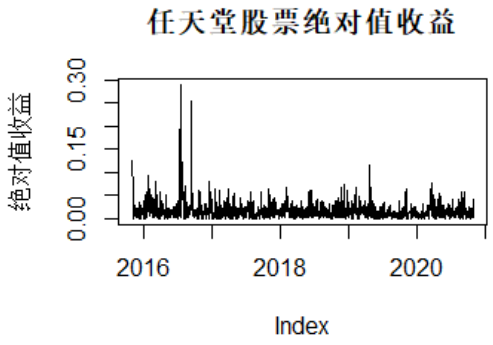


图 9 股票价格的绝对值收益

股票价格的绝对值序列同样描述了股票收益的稳定性，可以看出任天堂股票收益除 2016 年后半段有较大浮动外，其他时段较为稳定。

分别绘制出三个序列的自相关图如如10所示。

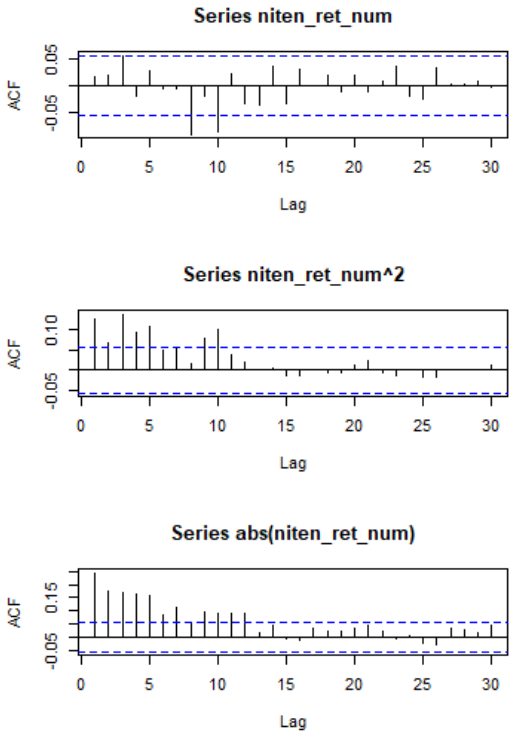


图 10 ACF 对比

从自相关图中可以看出，三个序列都有明显的拖尾，可以初步认定处理后的序列具有稳定性。另外，分别绘制出三个序列的偏自相关图如如10所示。三个序列的偏自相关图也同样具有明显的拖尾性，随着结束的增加偏自相关系数逐渐趋于 0，综合自相关图，可以初步认定这三个序列具有稳定性。

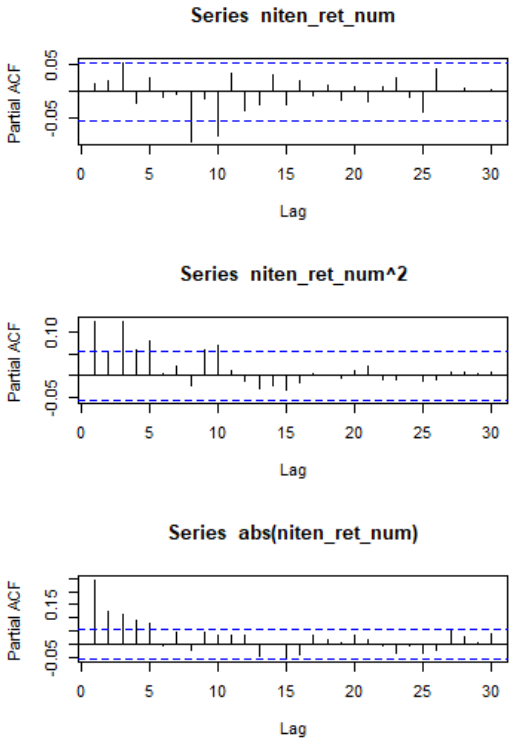


图 11 PACF 对比

下面主要对股票价格的收益序列进行分析，首先分析其基本统计，如表3，可以从表中也可以观察到平均值为 0, 其最大值和最小值差距并不大，其平均值大于 0，所以该股票长期来看是增值的。另外，该数据的分布有很大的峰度（肥尾）。

表 3 日收益的描述性统计

nobs	1657.000000
NAs	0.000000
Minimum	-0.124425
Maximum	0.289851
1. Quartile	-0.011452
3. Quartile	0.012458
Mean	0.000503
Median	0.000157
Sum	0.834015
SE Mean	0.000600
LCL Mean	-0.000673
UCL Mean	0.001680
Variance	0.000596
Stdev	0.024417
Skewness	1.713123
Kurtosis	20.787955

首先绘制出收益序列的直方图，如图12。

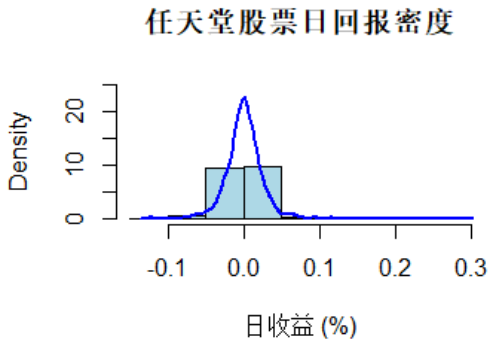


图 12 密度图

从直方图中可以看出，其基本分布在 −0.1 与 0.1 之间, 需要注意的是，由于序列经过了对数处理，所以该数值并非真实的收益，但是可以反应收益的大小以及趋势。可以看出任天堂股票整体还是具有不错的收益的。

QQ 图，在统计学中是通过比较两个概率分布的分位数，来比较这两个概率分布的概率图方法。首先选定分位数的对应概率区间集合，在此概率区间上，点 (x,y) 对应于第一个分布的一个分位数 x 和第二个分布在和 x 相同概率区间上相同的分位数。因此画出的是一条含参数的曲线，参数为概率区间的分割数。绘制出 QQ 图如图13所示。可以看出数据近似于正态分布,QQ 图上的点近似地在一条直线附近, 说明两分布线性相关，该直线的斜率为标准差, 截距为均值。

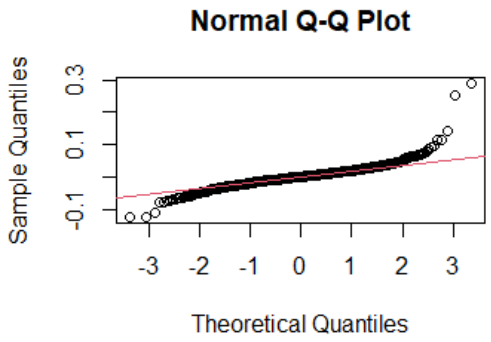


图 13 QQ 图

3.3 年度数据模式分析

3.3.1 箱线图

首先绘制出收益序列的箱线图，如图14所示。可以从图中看出任天堂股票在 16 年有比较大的波动，15 年有比较明显的下降，其他年份比较稳定。

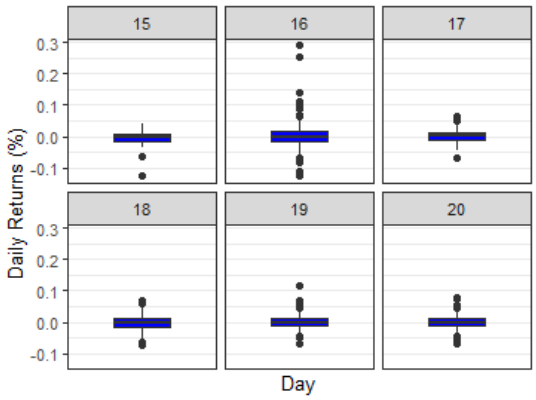


图 14 箱线图

3.3.2 散布图

散布图是用笛卡尔坐标系上的点表示资料中二个或多个变数分布方式的图。多半是在平面笛卡尔坐标上，表示二个变数的分布，若点有区分不同的颜色／形状／大小，可以用此特性表示另一个变数。三维的散布图可以呈现三个变数之间的关系，图中又用不同的颜色及表示另外一个变数散布图中的资料会用许多的点来表示，每个点表示一个资料，而其在水平座标轴及垂直座标轴上的座标，分别对应该资料的变数。绘制收益的出散布图如15所示。

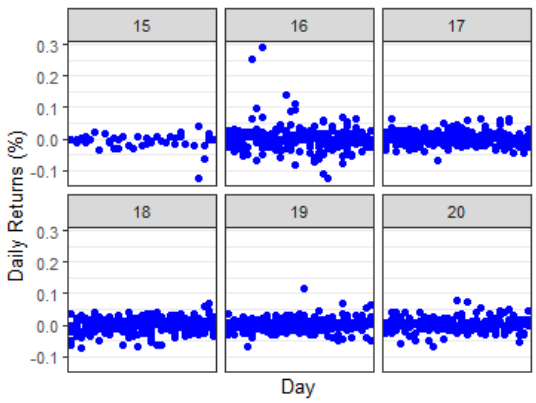


图 15 散布图

3.3.3 密度图

绘制出密度图如16所示。可以看到每年的收益都呈现正态分布，其中 15 年、17 年、19 年、20 年相对较窄，说明收益相较其他年份更稳定。

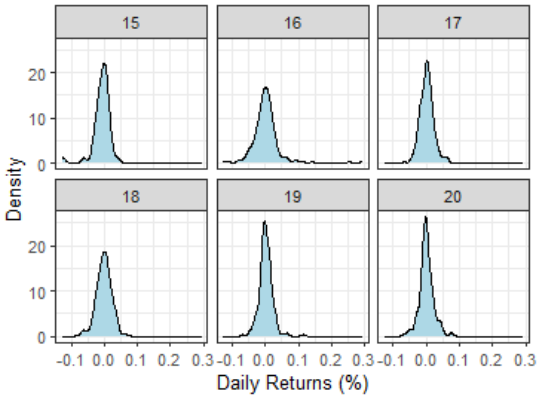


图 16 密度图

4 时间序列模型建模

4.1 线性模型

如果一个观察值序列通过预处理之后可以判定为平稳的非白噪声蓄序列，那么就可以使用 ARMA 模型对序列进行拟合，建模的步骤如图17所示。

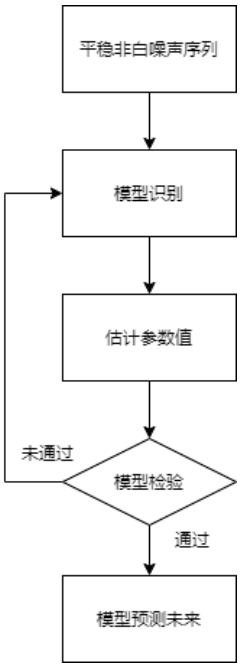


图 17 线性模型建立流程

4.1.1 平稳性检验

对平稳序列建模首先就要确认序列是平稳的，而图检验主要适用趋势或者周期比较明显的序列，最好使用统计方法，而其中 ADF 检验和 KPSS 检验都有广泛的应用，下面就使用这两种检验方法对序列进行稳定性检验。

Augmented Dicky-Fuller(ADF) 检验就是判断序列是否存在单位根：如果序列平稳，就不存在单位根；否则，就会存在单位根。所以，ADF 检验的 H_0 假设就是存在单位根，如果得到的显著性检验统计量小于置信度，则对应有一定的概率的把握来拒绝原假设。

首先对原序列做 ADF 检验，检验结果如表4。

表 4 原序列 ADF 检验

name	value
Dickey-Fuller	-1.915
Lag order	10
p-value	0.6143

可以从表中看出其 p 值大于 0.05，说明该序列并不平稳，而这与之前所得到的结论一致。下面对处理后的对数一阶差分序列做 ADF 检验，检验结果如表5所示。

表 5 对数一阶差分序列 ADF 检验

name	value
Dickey-Fuller	-11.851
Lag order	10
p-value	0.01

可以从表中看出，p 值小于 0.05，说明该序列平稳。

Kwiatkowski–Phillips–Schmidt–Shin (KPSS) 检验用于检验零假设，即可观察的时间序列在确定性趋势（即趋势平稳）周围是平稳的，而不是单位根的替代。由于 KPSS 是单边检验，故大于临界值拒绝原假设，也即原序列为非平稳。

首先对原序列做 KPSS 检验，得到的结果如表6所示。

表 6 原序列 KPSS 检验

name	value
Dickey-Fuller	10.608
Truncation lag parameter	7
p-value	0.01

原序列的 p 值小于 0.05，认为该序列不平稳，此结论与 ADF 检验得到的结果一致。同样用 KPSS 检验处理后的序列，得到的结果如表7所示。

表 7 对数一阶差分序列 KPSS 检验

name	value
Dickey-Fuller	0.055229
Truncation lag parameter	7
p-value	0.1

处理后的序列的 p 值大于 0.05，所以拒绝原假设，认为该序列平稳，此结论与 ADF 检验一致。最终可以得到结论，原序列不平稳，但是处理后的对数一阶差分序列平稳，所以可以对该序列使用 ARMA 建模。

4.1.2 纯随机性检验

平稳序列通常具有短期相关性，如果序列之间存在显著的相关关系，通常只存在于延迟时期比较短的序列值之间。如果一个平稳序列短期延迟的序列值之间都不存在显著的相关关系，通常长期延迟之间就更不会存在显著的相关关系了[9]。

分别进行处理后序列的延迟 6 阶、延迟 12 阶、延迟 24 阶的白噪声检验，如表8、表9、表10所示。

表 8 延迟 6 阶

name	value
X-squared	6.1248
df	6
p-value	0.4094

表 9 延迟 12 阶

name	value
X-squared	28.068
df	12
p-value	0.005407

表 10 延迟 24 阶

name	value
X-squared	37.596
df	24
p-value	0.03812

白噪声检验显示，延迟 12 阶、24 阶的 LB 统计量的 P 值不到 0.05 显著水平，所以显著拒绝序

列为随机序列的原假设，认为该序列为非白噪声序列。该对数收益率序列并非纯随机性序列，具有研究意义。

4.1.3 ARMA 模型

在时间序列的统计分析中，自回归移动平均 (ARMA) 模型根据两个多项式提供了对（弱）平稳随机过程的简约描述，一个用于自回归 (AR)，第二个用于移动平均（MA）。通过多次对模型的阶数进行尝试，按照 AIC 最小以及参数显著的原则，首先采用 ARMA(2,1) 梳系数模型对训练数据进行拟合。得到的参数如表。

表 11 ARMA(2,1) 参数

ar1	ar2	ar3	ma1
-0.7964	0.0340	0.0658	0.8176

4.1.4 ARIMA 模型

在统计学和计量经济学中，特别是在时间序列分析中，自回归综合移动平均 (ARIMA) 模型是自回归移动平均模型的推广。这两个模型都适用于时间序列数据，以便更好地理解数据或预测序列中的未来点（预测）。ARIMA 模型适用于某些情况下，其中数据显示均值意义上的非平稳性证据，其中初始差分步骤可以应用一次或多次以消除均值函数（即趋势）的非平稳性[10]。当季节性以时间序列显示时，可以应用季节性差异[11]来消除季节性成分。由于 ARMA 模型，根据 Wold 分解定理，在理论上足以描述常规（也称为纯非确定性）广义平稳时间序列，因此我们有动力将平稳时间序列设为非平稳时间序列，例如，通过使用差分，我们可以使用 ARMA 模型

采用 ARIMA(2,1,1) 模型对训练数据进行拟合，得到的参数如表12所示。

表 12 ARMA(2,1) 参数

ar1	ar2	ma1
0.4879	0.0267	-0.4725

4.1.5 模型检验

确定了拟合口径之后，还需要对该拟合口径进行必要的检验。模型的显著性检验主要是检验模型的有效性，一个模型是否显著有效主要看他提取的信息是否充分，一个好的模型能提取观察序列中素有的样本信息，拟合参数项中不再蕴含任何相关信息，即认为残差序列应该为白噪声序列。

首先对 ARMA(2,1) 模型进行检验，检验结果如图18所示。

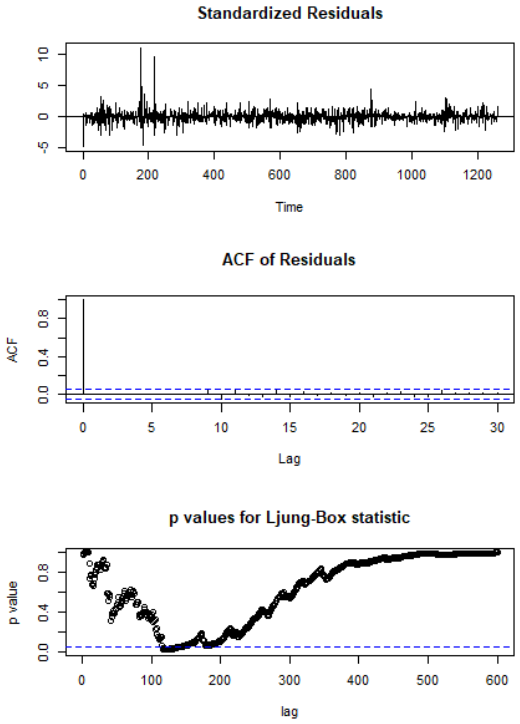


图 18 ARMA(2,1) 模型显著性检验

对拟合模型后得到的残差序列进行白噪声检验，由 LB 检验统计量的 P 值均大于 0.05（图中蓝线）得出残差为白噪声序列。因此模型可以很好地提取对数收益率序列中的有效信息。对 ARIMA(2,1,1) 模型进行模型显著性检验，如图19所示。

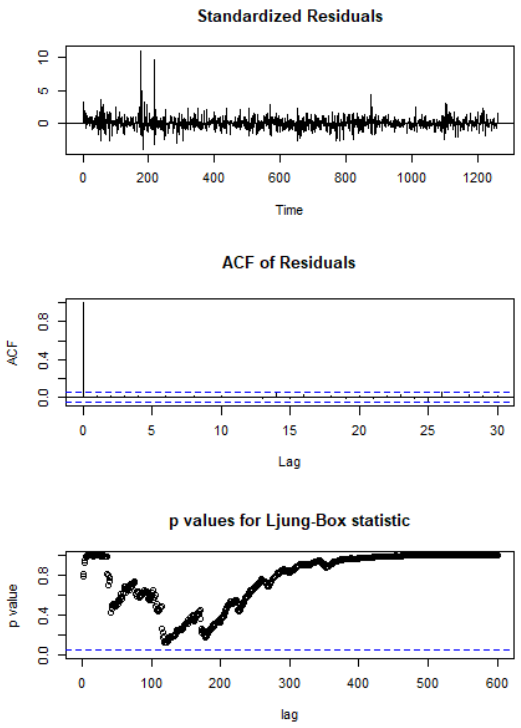


图 19 ARMA(2,1,1) 模型显著性检验

对拟合模型后得到的残差序列进行白噪声检验，由 LB 检验统计量的 P 值均大于 0.05（图中蓝线）得出残差为白噪声序列。因此模型同样可以很好地提取对数收益率序列中的有效信息

自回归模型和移动平均模型的系数应落在单位根内，$|z| <= 1$。做出因果检验和不可逆性检验如图20和图21所示。

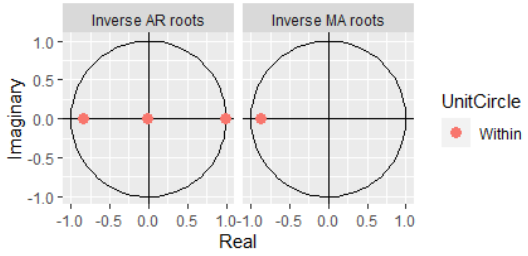


图 20 ARMA(2,1) 因果检验和不可逆性检验

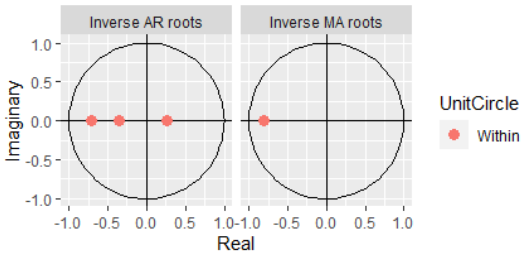


图 21 ARMA(2,1,1) 因果检验和不可逆性检验

自回归模型和移动平均模型的系数落在单位根内，$|z| <= 1$。

4.1.6 模型预测与对比

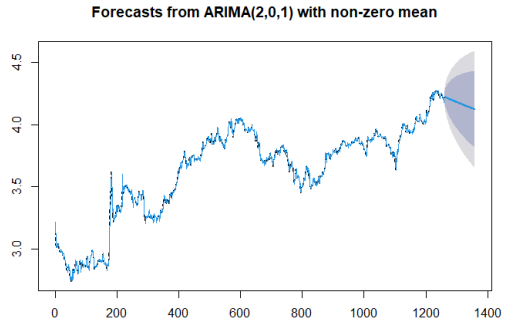


图 22 ARMA(2,1)

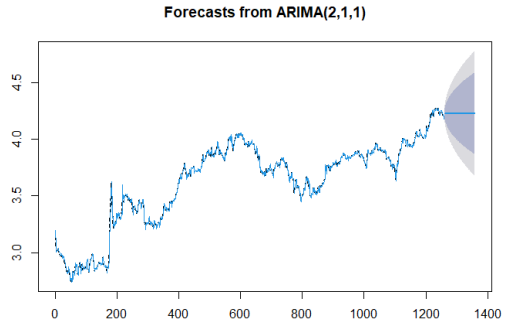


图 23 ARIMA(2,1,1)

4.2 季节模型

4.2.1 季节效应检验

对序列进行三参数指数平滑模型以及 SARIMA 模型拟合的前提均是序列存在季节效应。因此，先对序列进行因素分解查看序列是否存在

季节效应。由于对数收益率序列季节效应并不明显，且随着趋势的递增，每个季节的振幅维持相对稳定，因此本次实验对序列采用加法模型进行因素分解，得到结果如图24。

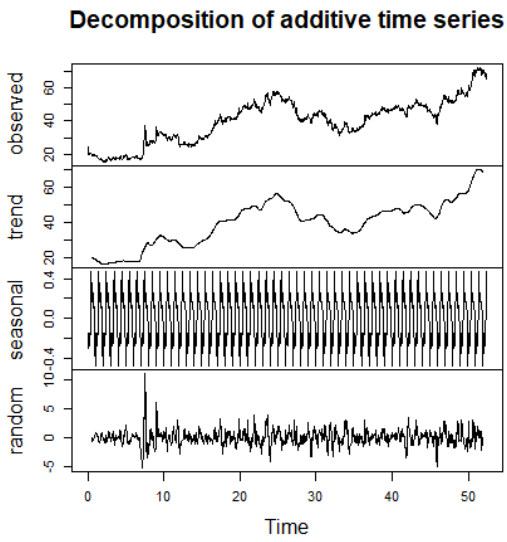


图 24 分解

4.2.2 三参数指数平滑模型

指数平滑是使用指数窗口函数平滑时间序列数据的经验法则。而在简单移动平均线中，过去的观察值被平均加权，指数函数用于分配随时间呈指数下降的权重。这是一个易于学习且易于应用的过程，用于根据用户的先前假设（例如季节性）进行某些确定。指数平滑通常用于时间序列数据的分析。

本次实验利用 Holt-Winters 三参数指数平滑加法模型对对数收益率进行拟合。考虑到一个月大概有 24 个交易日，因此三参数指数模型设置季节周期为 24。得到参数估计结果如表13。

平滑系数 $\alpha = 0.852884, \beta = 0.003188443, \gamma = 0.2986104$. 得到参数的最后迭代值为 $\alpha(t) = 68.24413538, \beta(t) = 0.07010694$ 参数的最后 24 个估计值对应的是一个月 24 天的季节指数。

表 13 三参数指数平滑参数

a	68.24413538
b	0.07010694
s1	-0.37930073
s2	-0.23970298
s3	-0.07558510
s4	-0.12639039
s5	-0.20089564
s6	-0.35747631
s7	-0.26554111
s8	-0.28555106
s9	-0.05263149
s10	-0.02551259
s11	-0.05369471
s12	-0.01193106
s13	-0.12148531
s14	0.28257625
s15	0.31372954
s16	0.53169924
s17	0.52564413
s18	0.26866455
s19	0.22590276
s20	0.24428399
s21	0.30076787
s22	0.27682238
s23	0.24368593
s24	0.06280663

4.2.3 SARIMA 模型

季节性自回归综合移动平均线，SARIMA 或季节性 ARIMA，是 ARIMA 的扩展，它明确支持具有季节性分量的单变量时间序列数据。它添加了三个新的超参数来指定序列季节性分量的自回归 (AR)、差分 (I) 和移动平均 (MA)，以及季节性周期的附加参数。

对对数序列的一阶二十四步差分进行纯随机性检验，得出 p 值小于 0.05，因此可以得出对数序列的一阶二十四步差分序列为非纯随机序列，具有研究意义。

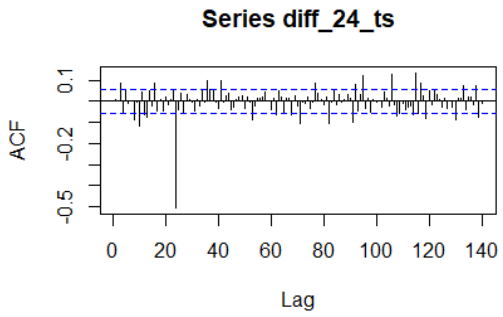


图 25 一阶 24 步差分 ACF

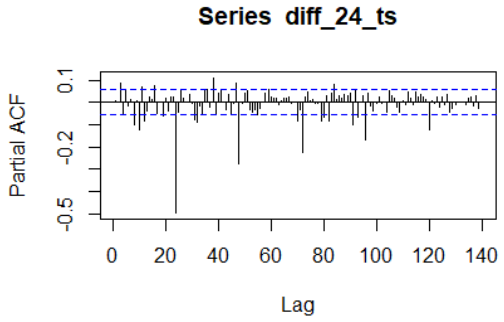


图 26 一阶 24 步差分 PACF

对数序列的一阶二十四步差分序列的自相关图和偏自相关图如图2526。发现其自相关图滞后 24 阶大于零，而其偏自相关图滞后 24 的倍数阶时大于零。因此可以初步判断，季节效应采用 $ARIMA(1,1,0)_{24}$ 模型拟合，即采用 $ARIMA(2,1,1) \times (1,1,0)_{24}$ 模型对股票收盘价格序列进行拟合。

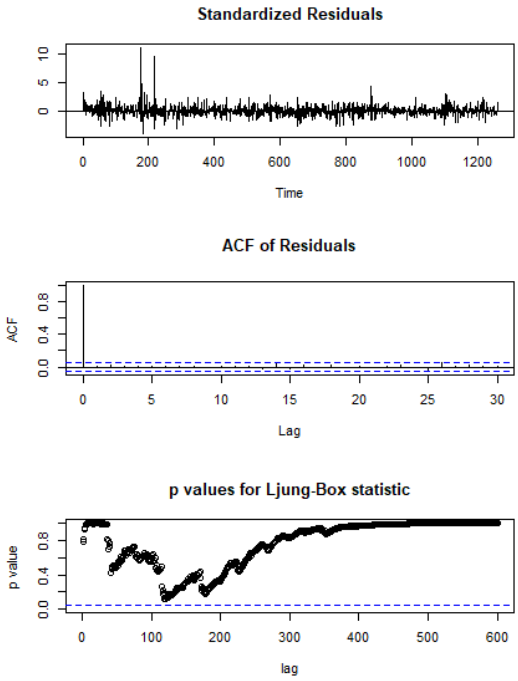


图 27 模型显著性

对拟合模型后得到的残差序列进行白噪声检验，由 LB 检验统计量的 P 值均大于 0.05 得出残差为白噪声序列。因此 $ARIMA(2,1,1)X(1,1,0)_{24}$ 模型同样可以很好地提取对数收益率序列中的有效信息。

4.2.4 模型预测与对比

由于数据采用的比较新，只有 9 天的数据可以用于测试，这 9 天的数据如表14。

表 14 最新数据

Date	Open	High	Low	Close	Adj C
2022-6-10	55	55.05	54	54.11	54.11
2022-6-09	55.89	55.92	55.09	55.25	55.25
2022-6-08	54.23	55.31	54.23	54.64	54.64
2022-6-07	54.34	55.42	54.34	55.12	55.12
2022-6-06	55	55.9	54.79	55.22	55.22
2022-6-03	54.2	55.16	54.2	54.42	54.42
2022-6-02	55.06	55.62	55.06	55.56	55.56
2022-6-01	56.08	56.64	55.71	55.71	55.71
2022-5-31	56.36	56.36	55.5	55.67	55.6

使用三参数指数平滑模型的预测结果如图28。

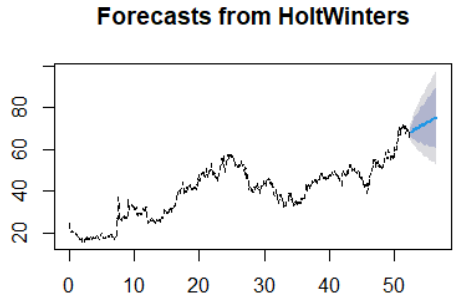


图 28 三参数指数平滑

使用 SARIMA 模型预测结果如图29所示。由于 SARIMA 模型是对数据的对数一阶差分数据拟合的，所以在误差分析阶段需要对数据进行复原。

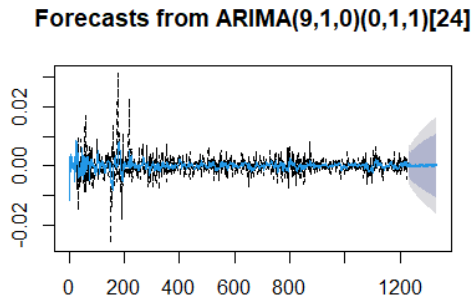


图 29 SARIMA

分别是用两个模型对数据进行预测，预测结

果如15。相比较来说三参数指数平缓模型更好一些。

表 15 季节模型的预测

日期	真实	Holt	SARIMA
2022-6-10	54.11	55.43494	56.09892683
2022-6-09	55.25	55.64465	56.10997878
2022-6-08	54.64	55.87887	56.09632556
2022-6-07	55.12	55.89817	56.10781914
2022-6-06	55.22	55.89377	56.0905186
2022-6-03	54.42	55.8073	56.08607824
2022-6-02	55.56	55.96934	56.07562754
2022-6-01	55.71	56.01944	56.07199548
2022-5-31	55.6	56.32247	56.05230707

4.3 异方差模型

4.3.1 异方差检验

由于上述已用 ARIMA 模型、三参数指数平滑模型以及 SARIMA 模型对序列进行拟合，且数据预处理时判断序列可能存在异方差性，因此，本实验先对模型拟合的残差进行 ARCH 检验。其中 Q 统计检验量的 p 值均小于显著性水平 0.05，因此认为以 AMIMA 模型以及 SARIMA 模型作为均值模型进行序列拟合得到的残差序列方差非齐且有自相关关系。

由图30可以看出，残差序列基本上围绕一个值波动，而残差平方序列存在扰动。同时，残差序列的 Q 统计量以及 LM 统计量均小于 0.05（图中虚线），SARMA 的残差图有同样的结论，说明 ARIMA 模型以及 SARIMA 模型得到的残差序列均存在异方差性。

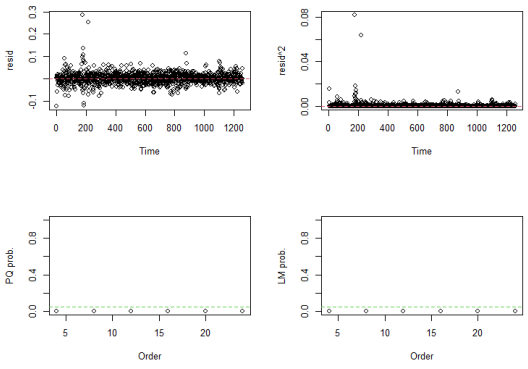


图 30 ARIMA

4.3.2 ARCH 模型

ARCH 模型（Autoregressive conditional heteroskedasticity model）全称“自回归条件异方差模型”，解决了传统的计量经济学对时间序列变量的第二个假设（方差恒定）所引起的问题。以 ε_t 表示收益或者收益残差，假设 $\varepsilon_t = \sigma_t z_t$ ，此处 $z_t \sim iid\ N(0,1)$（即独立同分布，均符合期望为 0，方差为 1 的正态分布）此处序列 σ_t^2 建模为

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2 \tag{2}$$

下面首先使用 ARCH(1) 模型对数据进行拟合。模拟结果如表16所示。

表 16 ARCH(1)

	Estimate	Std. Error	t value	Pr(>|t|)
mu	3.83845	0.00545	704.3384	0.00000
omega	0.00030	0.00005	5.4091	0.00000
alpha1	0.99900	0.04310	23.1737	0.00000
shape	99.999973	30.70663	3.2566	0.00112

4.3.3 GARCH 模型

如果方差用 ARMA 模型来表示，则 ARCH 模型的变形为 GARCH 模型,GARCH(p,q) 如式3。

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_q \varepsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 \tag{3}$$

GARCH(p , q) 过程的滞后长度 p 由三个步骤确定：

1. 估计最佳拟合 AR(q) 模型

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_q y_{tq} + \epsilon_t = a_0 + \sum_{i=1}^{q} a_i y_{ti} + \epsilon_t \tag{4}$$

2. 计算并绘制自相关

$$\rho = \frac{\sum_{t=i+1}^{T} (\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)(\hat{\epsilon}_{t-1}^2 - \hat{\sigma}_{t-1}^2)}{\sum_{t=1}^{T} (\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)^2} \tag{5}$$

3. 渐近，即对于大样本，标准差 $\rho(i)$ 是 $1/\sqrt{T}$. 大于此值的单个值表示 GARCH 错误。要估计滞后的总数，请使用 Ljung-Box 检验，直到这些值小于 10% 显着。Ljung–Box Q 统计量如下 χ^2 具有 n 个自由度的分布，如果平方残差 $epsilon_t^2$ 是不相关

的。建议最多考虑 n 的 T/4 值。原假设表明不存在 ARCH 或 GARCH 错误。因此，拒绝 null 意味着条件方差中存在此类错误。

下面使用 GARCH(1,1) 对模型进行拟合。模拟结果如表17所示。

表 17 GARCH(1,1)

	Estimate	Std. Error	t value	Pr(>|t|)
mu	3.83764	0.00513	746.7375	0.00000
omega	0.00017	0.00005	3.5886	0.00033
alpha1	0.85409	0.07013	12.1771	0.00000
beta1	0.14490	0.06173	2.3474	0.01890
shape	99.99999	29.96015	3.3378	0.00084

4.3.4 ARMA-GARCH 模型

使用 ARIMA(2,1,1)-GARCH(1,1) 模拟并绘制图像。

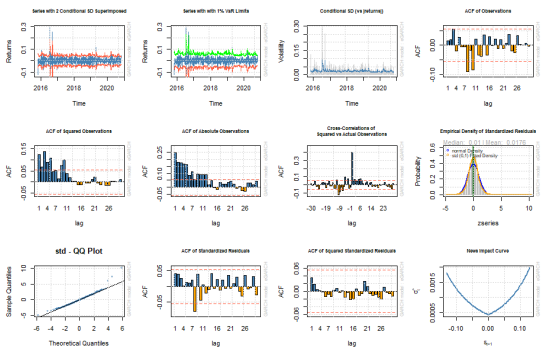


图 31 ARIMA(2,1,1)-GARCH(1,1)

画出残差序列的直方图和 QQ 图如图31，图像均可说明残差并不服从正态分布。但是由于模型拟合精度在可忍受的范围内，因此不对残差序列的分布做出修改。

表 18 GARCH(1,1)

	statistic	p-value
$Lag[1]$	1.538	0.2149
$Lag[2*(p+q)+(p+q)-1][5]$	1.874	0.6486
$Lag[4*(p+q)+(p+q)-1][9]$	2.113	0.8911

从表18看出，标准化残差序列检验结果延迟 1 阶时 LB 检验统计量的 P 值等于 0.2149，明显大于显著性水平 0.05，此后 LB 检验统计量的 P 值均大于 0.05，说明均值模型对水平信息的提取非常充分。标准化残差平方序列的检验结果显示，LB 检验统计量的 P 值同样均大于 0.05，说明 GARCH(1,1) 模型对波动信息的提取非常充分。模型显著成立。如图31残差序列及其 95% 的置信区间。可以看出，当序列小幅波动时置信区间更窄，当序列大幅波动时置信区间更宽。说明模型对序列波动风险的拟合和预测十分准确。

使用该模型模拟的结果如表19。

表 19 季节模型的预测

日期	真实	ARIMA-GARCH
2022-6-10	54.11	54.69683
2022-6-09	55.25	55.34465
2022-6-08	54.64	55.09656
2022-6-07	55.12	55.37810
2022-6-06	55.22	55.47377
2022-6-03	54.42	55.07824
2022-6-02	55.56	55.96934
2022-6-01	55.71	55.81944
2022-5-31	55.6	55.72470

4.4 机器学习模型

时间序列预测可以被构造为一个监督学习问题。通过对时间序列数据的重构，可以针对不同问题使用相关的机器学习算法。下面通过几种机器学习算法对任天堂股票数据进行建模。

4.4.1 MLP

多层感知器（Multilayer Perceptron,MLP）是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP 可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）。一种被称为反向传播算法的监督学习方法常被用来训练 MLP[12-13]。多层感知器遵循人类神经系统原理，学习并进行数据预测。它首先学习，然后使用权重存储数据，并使用算法来调整权重并减少训练过程中的偏差，即实际值和预测值之间的误差。主要优势在于其快速解决复杂问题的能力。多层感知的基本结构由三层组成：第一输入层，中间隐藏层和最后输出层，输入元素和权重的乘积被馈给具有神经元偏差的求和结点，主要优势在于其快速解决复杂问题的能力。[14] MLP 是感知器的推广，克服了感知器不能对线性不可分数据进行识别的弱点

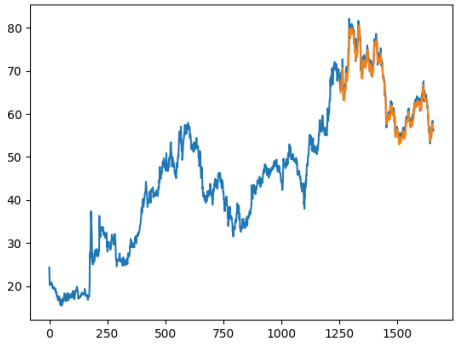


图 32 MLP

经过多次训练之后，可以计算出模型的误差箱线图如图33所示，可以看到整体的误差在 2.3 左右。

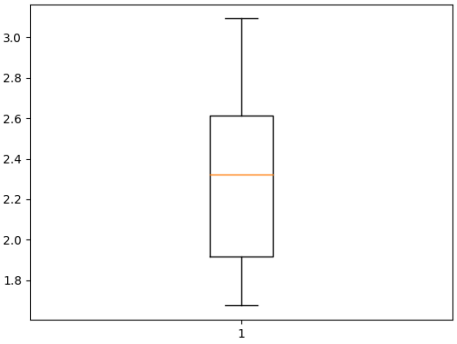


图 33 模型误差

4.4.2 LSTM

长短期记忆（Long Short-Term Memory，LSTM）是一种时间循环神经网络（RNN），论文首次发表于 1997 年。由于独特的设计结构，LSTM 适合于处理和预测时间序列中间隔和延迟非常长的重要事件。LSTM 的表现通常比时间循环神经网络及隐马尔科夫模型（HMM）更好，比如用在不分段连续手写识别上。2009 年，用 LSTM 构建的人工神经网络模型赢得过 ICDAR 手写识别比赛冠军。LSTM 还普遍用于自主语音识别，2013 年运用 TIMIT 自然演讲资料库达成 17.7% 错误率的纪录。作为非线性模型，LSTM 可作为复杂的非线性单元用于构造更大型深度神经网络。

可以将问题表述为回归问题。也就是说，给定今日的股票价格，下一天的股票价格是多少？可以编写一个简单的函数将单列数据转换为两列数据集：第一列包含今日的价格，第二列包含下一天的价格，以进行预测。LSTM 对输入数据的规模很敏感，特别是在使用 sigmoid 或 tanh 激活函数时。将数据重新缩放到 0 到 1 的范围是一个很好的做法，也称为规范化。在这里我使用了 scikit-learn 库中的 MinMaxScaler 预处理类标准化数据集。经过标准化的数据如图34所示。

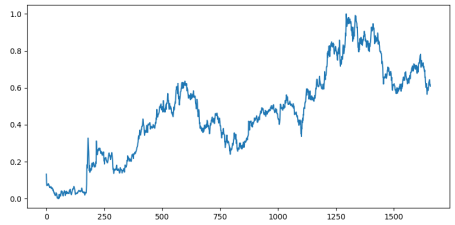


图 34 标准化后的数据

对于时间序列数据，我们将有序数据集拆分为训练数据集和测试数据集。由于数据采集的比较新，如果将所有数据都作为训练集，剩余可用于测试的部分并不多，所以需要重新划分。下面将数据分成训练数据集，其中 67% 的观察可以用来训练模型，剩下的 33% 用于测试模型。该网络有一个带有 1 个输入的可见层，一个带有 4 个 LSTM 块或神经元的隐藏层，以及一个进行单值预测的输出层。默认的 sigmoid 激活函数用于 LSTM 块。网络训练了 100 个 epoch。最后，使用模型为训练和测试数据集生成预测，以获得模型效果。由于数据集的方式不一致，必须改变预测，使它们在 x 轴上与原始数据集对齐。准备好后，绘制数据如图35，以蓝色显示原始数据集，以绿色显示训练数据集的预测，以及以红色显示未见过的测试数据集的预测。

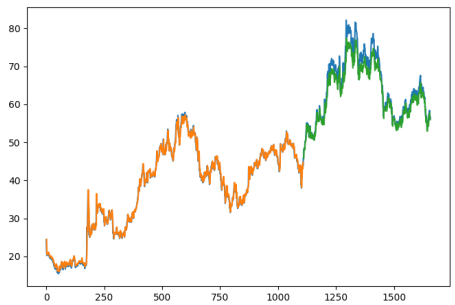


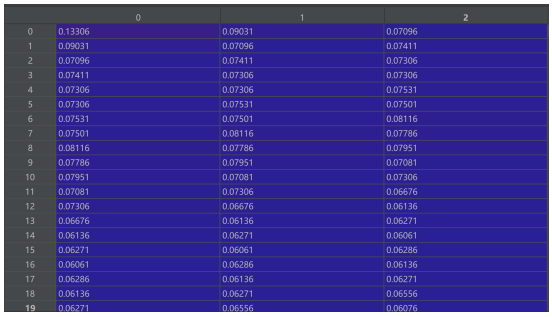
图 35 LSTM 模型

可以从图中看出模型在测试集部分预测非常准确，在测试集部分也有一定的准确性，但是还是存在误差，定量研究的误差如表21所示。

表 20 LSTM 模型误差

Train Score	0.94 RMSE
Test Score	2.29 RMSE

还可以对问题进行表述，以便可以使用多个最近的时间步来预测下一个时间步。这称为窗口，窗口的大小是可以针对每个问题进行调整的参数。例如，给定当前时间 (t)，我们想要预测序列 $(t+1)$ 中下一个时间的值，我们可以使用当前时间 (t) 以及之前的两个时间 $(t-1$ 和 $t-2)$ 作为输入变量。当表述为回归问题时，输入变量为 $t-2$、$t-1$、t，输出变量为 $t+1$。我们通过将 look_back 参数从 1 增加到 3 来创建时间序列问题的公式。具有此公式的数据集示如图36示：



	0	1	2
0	0.13306	0.09031	0.07096
1	0.09031	0.07096	0.07411
2	0.07096	0.07411	0.07306
3	0.07411	0.07306	0.07306
4	0.07306	0.07306	0.07531
5	0.07306	0.07531	0.07501
6	0.07531	0.07501	0.08116
7	0.07501	0.08116	0.07786
8	0.08116	0.07786	0.07951
9	0.07786	0.07951	0.07081
10	0.07951	0.07081	0.07306
11	0.07081	0.07306	0.06676
12	0.07306	0.06676	0.06136
13	0.06676	0.06136	0.06271
14	0.06136	0.06271	0.06061
15	0.06271	0.06061	0.06286
16	0.06061	0.06286	0.06136
17	0.06286	0.06136	0.06271
18	0.06136	0.06271	0.06556
19	0.06271	0.06556	0.06076

图 36 窗口型 LSTM 输入

做出的图像如图37。

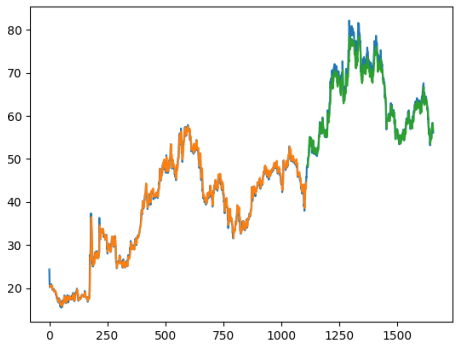


图 37 窗口型 LSTM 预测

LSTM 网络具有记忆，能够跨长序列记忆。在拟合模型时，网络中的状态会在每个训练批次后重置，以及每次调用 model.predict() 或 model.evaluate()。通过使 LSTM 层 “有状态”，可以更好地控制何时在 Keras 中清除 LSTM 网络的内部状态。它可以在整个训练序列上建立状态，甚至在需要进行预测时保持该状态。它要求在拟合网络时不打乱训练数据。它还需要在每次暴露于训练数据（epoch）后通过调用 model.reset_states() 显式重置网络状态。必须创建自己的 epoch 外循环，并在每个 epoch 中调用 model.fit() 和 model.reset_states()。该模型的模拟结果如图38所示。

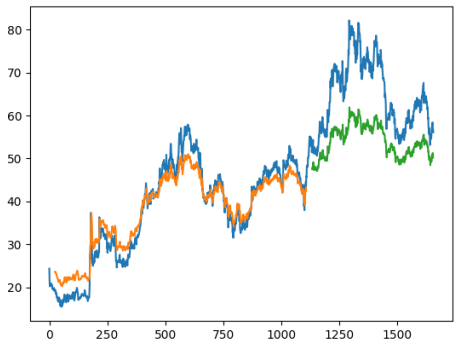


图 38 批次之间具有记忆的 LSTM 预测

LSTM 还具有一大优势：当它们堆叠到深度网络架构中时，它们可以被成功训练。LSTM 网络可以在 Keras 中堆叠，就像其他层类型可以堆叠一样。所需配置的一个补充是每个后续 LSTM 层之前的 LSTM 层必须返回序列。这可以通过将图层上的 return_sequences 参数设置为 True 来完成。我们将上一个模型中的有状态 LSTM 扩展为具有两层，训练结果如图39。

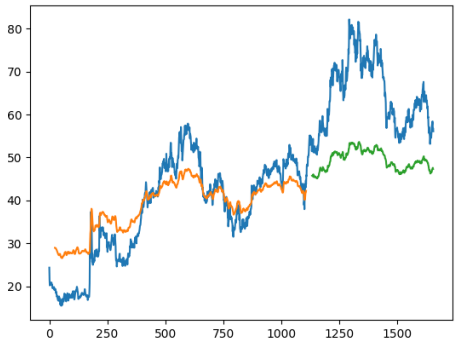


图 39 批次之间具有堆叠的 LSTM 预测

在该部分我使用了四种不同的 LSTM 模型对数据进行建模，四种 LSTM 模型的误差对比。

表 21 四种 LSTM 模型比较

模型	Train Score	Test Score
普通回归网络	0.94RMSE	2.24RMSE
移动窗口回归	0.96RMSE	1.61RMSE
批次之间具有记忆	3.02RMSE	11.62RMSE
批次之间具有堆叠	5.66RMSE	16.39RMSE

4.4.3 CNN

卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元 |[15]，对于大型图像处理有出色表现。

卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络需要考量的参数更少，使之成为一种颇具吸引力的深度学习结构[16]。

同样的将数据集合的 66% 作为训练集，剩余的 33% 作为测试集。设置 CNN 网络循环次数以及批次，定义具有两个一维卷积层的 CNN 模型用于从输入序列中提取特征，在卷积层之后使用最大池化层，将加权的输入提取特征，从而将输入大小减小 1/4，合并的输入在被解释并用于进行单步预测之前，将其展平为一个长向量。最终预测如图27所示。

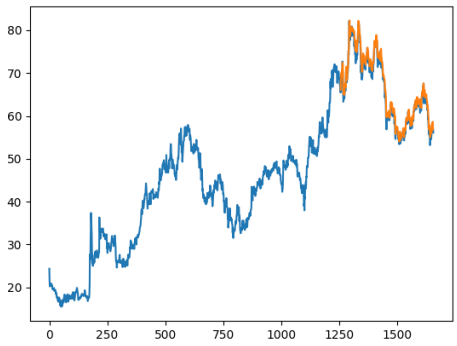


图 40 CNN 预测

4.4.4 模型比较

三种机器学习方法的比较如表22，可创意从表中看出整体三个模型在测试集的预测都算比较准取，其中 LSTM 的误差最小，对数据的预测效果最好，而 CNN 的预测效果只比 LSTM 略差，而 MLP 的预测效果是三种机器学习算法最差的，说明其不适合本数据集。

表 22 三种机器学习方法比较

模型	Test Score
MLP	2.24RMSE
LSTM	1.61RMSE
CNN	1.75RMSE

5 总结与展望

本文首先通过线性模型 ARMA 和 ARIMA 对数据建模，发现模型的效果并不理想。然后考虑了数据可能具有的季节性，使用了三参数指数平滑模型和 ARIMA 季节模型对数据建模。考虑到数据可能异方差，所以还使用了异方差模型对股票数据建模。其中季节模型和异方差模型通过了模型显著性检验和参数显著性检验且对数据的预测准确率都较高。另外，本文还使用了现在流行的机器学习方法对数据建模，包括 MLP、LST 和 CNN，经过计算其中 LSTM 的预测效果最好。

新冠肺炎疫情促使日本电玩大厂任天堂的业绩大幅成长，并带动股价攀涨至近 12 年来的高点，但任天堂股价持续上涨的空间有限。根据任天堂的投资者关系页面，该公司已在全球售出约 1.08 亿台 Switch。这个庞大的安装基础带来了大量的软件/游戏销售，这推动了任天堂的大部分利润。但最大的问题是任天堂何时会发布其下一代设备。对于任天堂的业务来说，一个成功的新硬件设备是必不可少的，因为它是让人们购买游戏的引擎。如果任天堂能够在原始 Switch 的成功基础上再接再厉，并继续每年销售数亿款游戏，那么其股票在 2027 年可能会处于良好状态。

参考文献

[1] KHAN U, AADIL F, GHAZANFAR M A, et al. A robust regression-based stock exchange forecasting and determination of correlation between stock markets[J]. Sustainability, 2018, 10(10): 3702.

[2] GUPTA R, CHEN M. Sentiment analysis for stock price prediction [C]//2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE, 2020: 213-218.

[3] DIAS R, ALEXANDRE P, HELIODORO P. Contagion in the lac financial markets: The impact of stock crises of 2008 and 2010[J]. Littera Scripta, 2020, 13(1): 32-45.

[4] RAJESH N, GANDY L. Cashtagnn: Exploiting the use of cashtags to predict stock market prices using convolutional networks [C]//Proceedings of the 2020 4th International Conference on Algorithms, Computing and Systems. 2020: 1-5.

[5] EBADATI O, MORTAZAVI M. An efficient hybrid machine learning method for time series stock market forecasting[J]. Neural Network World, 2018, 28(1): 41-55.

[6] GANDHMAL D P, KUMAR K. Systematic analysis and review of stock market prediction techniques[J]. Computer Science Review, 2019, 34: 100190.

[7] VOCHOZKA M, HORAK J, KRULICKY T. Innovations in management forecast: Time development of stock prices with neural networks[J]. 2020.

[8] HORÁK J, KRULICKỲ T. Comparison of exponential time series alignment and time series alignment using artificial neural networks by example of prediction of future development of stock prices of a specific company[C]//SHS Web of Conferences: volume 61. EDP Sciences, 2019: 01006.

[9] 张树京, 齐立心. 时间序列分析简明教程[M]. 清华大学出版社有限公司, 2003.

[10] CRAIGMILE P F. Simulating a class of stationary gaussian processes using the davies–harte algorithm, with application to long memory processes[J]. Journal of Time Series Analysis, 2003, 24 (5): 505-511.

[11] HYNDMAN R J, ATHANASOPOULOS G. 8.9 seasonal arima models[J]. Forecasting: principles and practice. oTexts. Retrieved, 2015, 19.

[12] ROSENBLATT F. Principles of neurodynamics. perceptrons and the theory of brain mechanisms[R]. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[13] ㊣本健司. David E. Rumelhart, James L. McClelland and the PDP Research Group 著," Parallel Distributed Processing Explorations in the Microstructure of Cognition", Vol. 1, Vol. 2, MIT Press, 約\7,480, 1986[J]. 情報㊣理, 1988, 29(9): 1035-1036.

[14] LI R Y M, TANG B, CHAU K W. Sustainable construction safety knowledge sharing: a partial least square-structural equation modeling and a feedforward neural network approach[J]. Sustainability, 2019, 11(20): 5831.

[15] DHEIR I M, METTLEQ A S A, ELSHARIF A A, et al. Classifying nuts types using convolutional neural network[J]. International Journal of Academic Information Systems Research (IJAISR), 2020, 3(12).

[16] O'SHEA K, NASH R. An introduction to convolutional neural networks[J]. arXiv preprint arXiv:1511.08458, 2015.

附录：代码

全代码

```
# R语言部分
library(readxl)
library(TSA)
library(tseries)
library(forecast)
library(lattice)
library(ggplot2)
library(lattice)
library(StatDA)
library(rugarch)
library(zoo)
# 读取数据
Data<- read_excel("D:/Study/大三下/5 时间序列分析/时间序列大作业/NTDOY.xlsx")
View(Data)
Stock_data<-Data[,c(1,6)]
View(Stock_data)
# 直方图
lines(density(Stock_data$`Adj Close`))
 hist(x=Stock_data$`Adj Close`,breaks = 12,main="直方图",
 xlab="收盘价",ylab="密度",freq=F)
# 收盘价时序图
nitenlets <- zoo(Stock_data$`Adj Close`, as.Date(as.character(Stock_data$Date),
format = c("%Y-%m-%d")))
#log return time series
niten_rets <- log(nitenlets/lag(nitenlets,-1))
niten_ret_num <- coredata(niten_rets)
plot(Stock_data, type = "l",xlab="时间",ylab="收盘价",main="任天堂收盘价")
# ACF 和 PACF
acf(coredata(Stock_data$`Adj Close`), main="任天堂股票收盘价ACF")
pacf(coredata(Stock_data$`Adj Close`), main="任天堂股票收盘价PACF")
```

```
# 对数收益
returns <- diff(log(Stock_data$`Adj Close`))
basicStats(returns)
plot(returns, type='l', ylab = "时间", main="任天堂股票对数收益")
#收益密度
hist(returns, col = "lightblue", border = "black", prob = TRUE,
ylim = c(0, 25),
xlab = "Daily Returns (%)", main = "Density of Daily NINTENDO Stock Returns")
lines(density(returns), lwd = 2, col = "red")
#年度模式
Stock_data$Date <- as.Date (Stock_data$Date, formal = "%d/%m/%y")
str(Stock_data)
Year <- format(Stock_data$Date, "%y")
Day <- format(Stock_data$Date, "%d")
Date <- data.frame(Day, Year, c(returns,NA))
#箱线图
Date %>%
ggplot(aes(x = frequency(c(returns,NA)),y = c(returns,NA))) +
geom_boxplot(fill = "Blue") +
    facet_wrap( ~ Year) +
    labs(y = "Daily Returns (%)", x = "Day") +
    theme_bw(base_size = 10) +
    scale_x_discrete(breaks = 0)
#散布图
Date %>%
ggplot(aes(x = Day, y = c(returns,NA))) + geom_point(color = "blue") +
    facet_wrap( ~ Year) +
    labs(y = "Daily Returns (%)", x = "Day") +
    theme_bw(base_size = 10) +
    scale_x_discrete(breaks = c(0,50,100,150,200,250,300))
#密度图
Date %>%
```

```
ggplot(aes(x = c(returns,NA)), y = frequency(c(returns,NA))) +
geom_density(fill = "lightblue") +
    facet_wrap( ~ Year) +
    labs(x = "Daily Returns (%)", y = "Density") +
    theme_bw(base_size = 10)
#QQ-Plot of NINTENDO Daily Returns
# 平方收益
plot(apple_rets^2,type='l', ylab = "square of stock price return",
main="Plot of 2002-2017 daily apple stock price squared return")
#绝对值收益
plot(abs(apple_rets),type='l', ylab = "abs value of stock price return",
main="Plot of 2002-2017 daily apple stock price abs return")
# acf
par(mfrow=c(3,1))
acf(niten_ret_num)
acf(niten_ret_num^2)
acf(abs(niten_ret_num))
# pacf
par(mfrow=c(3,1))
pacf(niten_ret_num)
pacf(niten_ret_num^2)
pacf(abs(niten_ret_num))
# 线性模型
# 平稳性检验
# 原始序列
adf.test(log(Stock_data$`Adj Close`))
kpss.test(log(Stock_data$`Adj Close`))
# 新序列
diff.data <- diff(log(Stock_data$`Adj Close`))
adf.test(diff.data)
kpss.test(diff.data)
#纯随机检验
```

```
Box.test(niten_ret_num^2, lag=2, type="Ljung")
Box.test(niten_ret_num^2, lag=4, type="Ljung")
Box.test(niten_ret_num^2, lag=6, type="Ljung")
#ARMA
ModelX <- Arima(log(Stock_data$`Adj Close`), order = c(2,0,1), method = "ML")
summary(ModelX)
#ARIMA
ModelY <- Arima(log(Stock_data$`Adj Close`), order = c(2,1,1), method = "ML")
summary(ModelY)
tsdiag(ModelX)
tsdiag(ModelY)
autoplot(ModelX)
autoplot(ModelY)
# 分解
ns_ts <- ts(ns_df,frequency = 365,start = 2016-04-03 )
ns_de <- decompose(ns_ts)
plot(ns_de)
# 季节
 ns_ts <- ts(Stock_data$`Adj Close`,frequency = 24,start =0)
 ns_de <- decompose(ns_ts)
 plot(ns_de)
 ModelH<-HoltWinters(ns_ts)
 ModelH
PP.test(diff(diff(log(2.7,Stock_data$`Adj Close`),24)))
ns_ts=diff(diff(log(2.7,Stock_data$`Adj Close`),24))
fit<-arima(x=ns_ts,order=c(3,0,1),seasonal = list(order=c(0,1,1),period=24),transform.
foreY<-forecast::forecast(ModelY,h=100)
plot(foreY,lty=2)
lines(foreY$fitted,col=4)
# 多层感知机
from math import sqrt
import numpy
```

```
from numpy import array
from numpy import mean
from numpy import std
from pandas import DataFrame
from pandas import concat
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from matplotlib import pyplot
def train_test_split(data, n_test):
    return data[:-n_test], data[-n_test:]
def series_to_supervised(data, n_in, n_out=0):
    df = DataFrame(data)
    cols = list()
    for i in range(n_in, -1, -1):
        cols.append(df.shift(i))
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    agg = concat(cols, axis=1)
    agg.dropna(inplace=True)
    return agg.values[:, :-1], agg.values[:, -1]
def measure_rmse(actual, predicted):
    return sqrt(mean_squared_error(actual, predicted))
def model_fit(train, config):
    n_input, n_nodes, n_epochs, n_batch = config
    train_x, train_y = series_to_supervised(train, n_input)
    model = Sequential()
    model.add(Dense(n_nodes, activation='relu', input_dim=n_input))
    model.add(Dense(1))
    model.compile(loss='mse', optimizer='adam')
    model.fit(train_x, train_y, epochs=n_epochs, batch_size=n_batch, verbose=0)
```

```
    return model
def model_predict(model, history, config):
    n_input, _, _, _ = config
    x_input = array(history[-n_input:]).reshape(1, n_input)
    yhat = model.predict(x_input, verbose=0)
    return yhat[0]
def walk_forward_validation(data, n_test, cfg):
    predictions = list()
    train, test = data[:-n_test], data[-n_test:]
    model = model_fit(train, cfg)
    # seed history with training dataset
    history = [x for x in train]
    # 开始单步循环预测数据
    for i in range(len(test)):
        yhat = model_predict(model, history, cfg)
        predictions.append(yhat)
        history.append(test[i])
    # 评估预测结果的误差
    error = measure_rmse(test, predictions)
    print(' > %.3f' % error)
    return error
def repeat_evaluate(data, config, n_test, n_repeats=5):
    # fit and evaluate the model n times
    scores = [walk_forward_validation(data, n_test, config) for _ in range(n_repeats)]
    return scores
def summarize_scores(name, scores):
    scores_m, score_std = mean(scores), std(scores)
    print('%s: %.3f RMSE (+/- %.3f)' % (name, scores_m, score_std))
    # 画出箱线图
    pyplot.boxplot(scores)
    pyplot.show()
series = read_csv('NTDOY2.csv', header=0, index_col=0)
```

```
data = series.values
# data split
n_test = 400
# 配置信息，[输入长度n_input， 节点数n_nodes， 周期n_epochs， 批次n_batch]
config = [24, 500, 10, 10]
# scores=walk_forward_validation(data,n_test,config)
# pyplot.boxplot(score)
# pyplot.show()
# grid search
scores = repeat_evaluate(data, config, n_test)
# 汇总模型均方差
summarize_scores('mlp', scores)

# CNN
from math import sqrt
from numpy import array
from numpy import mean
from numpy import std
from pandas import DataFrame
from pandas import concat
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from matplotlib import pyplot
def train_test_split(data, n_test):
        return data[:-n_test], data[-n_test:]
def series_to_supervised(data, n_in=1, n_out=1):
        df = DataFrame(data)
```

```
        cols = list()
        for i in range(n_in, 0, -1):
                cols.append(df.shift(i))
        for i in range(0, n_out):
                cols.append(df.shift(-i))
        agg = concat(cols, axis=1)
        agg.dropna(inplace=True)
        return agg.values
def measure_rmse(actual, predicted):
        return sqrt(mean_squared_error(actual, predicted))
def model_fit(train, config):
        n_input, n_filters, n_kernel, n_epochs, n_batch = config
        data = series_to_supervised(train, n_in=n_input)
        train_x, train_y = data[:, :-1], data[:, -1]
        train_x = train_x.reshape((train_x.shape[0], train_x.shape[1], 1))
        model = Sequential()
        model.add(Conv1D(filters=n_filters,
        kernel_size=n_kernel, activation='relu', input_shape=(n_input, 1)))
        model.add(Conv1D(filters=n_filters,
        kernel_size=n_kernel, activation='relu'))
        model.add(MaxPooling1D(pool_size=2))
        model.add(Flatten())
        model.add(Dense(1))
        model.compile(loss='mse', optimizer='adam')
        model.fit(train_x, train_y, epochs=n_epochs,
        batch_size=n_batch, verbose=0)
        return model
def model_predict(model, history, config):
        n_input, _, _, _, _ = config
        x_input = array(history[-n_input:]).reshape((1, n_input, 1))
        # forecast
        yhat = model.predict(x_input, verbose=0)
```

```
        return yhat[0]
def walk_forward_validation(data, n_test, cfg):
    predictions = list()
    train, test = data[:-n_test], data[-n_test:]
    model = model_fit(train, cfg)
    history = [x for x in train]
    # 开始单步循环预测数据
    for i in range(len(test)):
        yhat = model_predict(model, history, cfg)
        predictions.append(yhat)
        history.append(test[i])
    # pyplot.show()
    error = measure_rmse(test, predictions)
    print(' > %.3f' % error)
    return error
def repeat_evaluate(data, config, n_test, n_repeats=5):
        scores = [walk_forward_validation(data, n_test, config) for _ in range(n_repea
        return scores
def summarize_scores(name, scores):
        # print a summary
        scores_m, score_std = mean(scores), std(scores)
        print('%s: %.3f RMSE (+/- %.3f)' % (name, scores_m, score_std))
        # box and whisker plot
        pyplot.boxplot(scores)
        pyplot.show()
series = read_csv('NTDOY2.csv', header=0, index_col=0)
data = series.values
# data split
n_test = 400
# 配置信息，[输入长度n_input, 节点数n_nodes, 周期n_epochs, 批次n_batch]
config = [24, 256, 3, 5, 5]
# walk_forward_validation(data,n_test,config)
```

```
scores=walk_forward_validation(data,n_test,config)
scores = repeat_evaluate(data, config, n_test)
# summarize scores
summarize_scores('cnn', scores)
# LSTM
import numpy
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import pandas as pd
pd.set_option('display.max_columns',1000)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth',1000)
def create_dataset(dataset, look_back=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-look_back-1):
                a = dataset[i:(i+look_back), 0]
                dataX.append(a)
                dataY.append(dataset[i + look_back, 0])
        return numpy.array(dataX), numpy.array(dataY)
# 定义随机种子，以便重现结果
numpy.random.seed(7)
# 加载数据
dataframe = read_csv('NTDOY2.csv', usecols=[1], engine='python')
dataset = dataframe.values
dataset = dataset.astype('float32')
# 缩放数据
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:],
dataset[train_size:len(dataset),:]
look_back = 3
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
# 重构输入数据格式 [samples, time steps, features] = [93,1,3]
trainX = numpy.reshape(trainX, (
trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (
testX.shape[0], 1, testX.shape[1]))
# 构建 LSTM 网络
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=30, batch_size=1, verbose=2)
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
trainScore = math.sqrt(mean_squared_error(
trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(
testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :]
= trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+
(look_back*2)+1:len(dataset)-1, :] = testPredict
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```