西南大学研究生课程考试

答　卷　纸

考试科目：	时间序列分析
院、所、中心：	数学与统计学院
专业或专业领域：	应用统计
研究方向：	不区分方向
级　　别：	2020
学　　年：	2020-2021 学年
学　　期：	第二学期
姓　　名：	杜兴兴
学　　号：	112020314201385
类　　别：	②全日制硕士

（①全日制博士 ②全日制硕士 ③教育硕士 ④高师硕士
⑤工程硕士 ⑥农推硕士 ⑦兽医硕士 ⑧进修)

2021 年 5 月 4 日

研 究 生 院 (筹)制

课 程 类 别		②选修课
课程考试方式		③课程论文
题号	得分	教 师 评 价
一		
二		
三		
四		
五		
六		
七		
八		
九		
十		
总分		
任课教师签名：		

备注：成绩评定以百分制或等级制评分，每份试卷均应标明课程类别(①必修课②选修课③同等学力补修课)与考核方式（①闭卷考试②口试③开卷考试④课程论文)。课程论文应给出评语。

正文目录

图目录

表目录

ARIMA 模型对医药 300 股票指数的实证分析及预测

摘要

股票是市场经济中必不可少的部分，随着股票市场的不断发展，人们已经习惯将购买股票作为一种重要的透支手段。股票的价格预测能够为投资者进行投资时给与一定的意见参考。股票价格预测是对股票市场进行量化分析的一个过程，在各种量化分析方法中，时间序列分析是最经典的方式。在建立时间序列模型时，首先提取股票开盘价最为序列，然后将数据代入模型中，最后训练模型，将训练完成的模型应用与未来股票价格预测。

本报告选取了 2015 年 5 月至 2021 年 5 月的医药 300 指数股票的开盘价，通过频率变化之后，取每周的最低开盘价最为分析序列，建立 ARIMA 模型对医药 300 指数的总体股票趋势做出预测，来研究股票的短期变化规律。本报告使用“金点子”软件进行股票数据收集，使用 python3.8 软件进行统计分析，相关代码见附录。

一、 数据预处理

(一) 平稳性检验

数据的平稳性检验是建立 ARIMA 条件的前提，如果不满足，那么需要对时间序列数据的平稳性进行修正和调整。常用的检验方法有如下两种：

对序列的平稳性有两种检验方法，一种是根据时序图和自相关图显示的特征做出判断的图检验法；一种是构造检验统计量进行假设检验的方法。

1. 图示法

根据平稳时间序列均值、方差为常数的性质，平稳序列的时序图应该显示出该序列始终在一个常数值附近随机波动。以周作为单位长度，对选取的时间序列数据绘制初始的时序走势图，如图 1 所示。

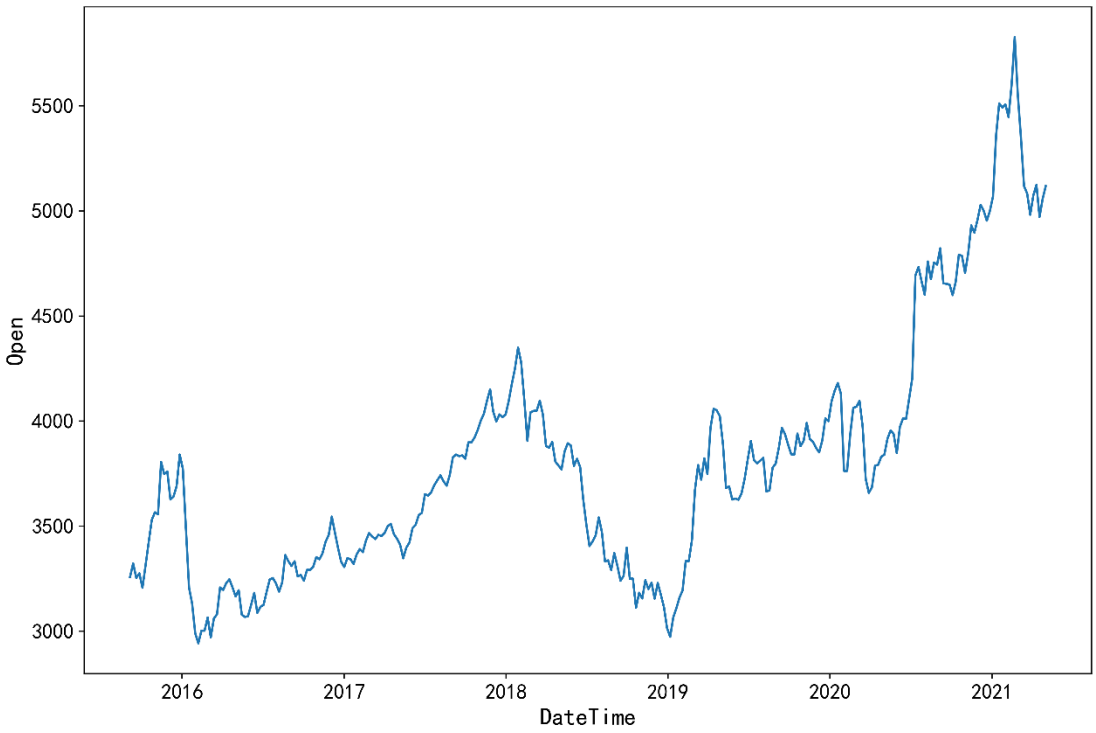

图 1　医药 300 股票指数时间序列图

观察时序图可以发现，医药 300 指数的开盘价时序图呈现出剧烈的上下波动，且总体上有一个上升的趋势，所以初步判断该序列存在某种线性趋势。为非平稳序列。但真实结论还需要通过假设检验进行验证，即如下的单位根检验。

2. 单位根检验

单位根检验是对序列是否存在单位根进行判断：如果序列平稳，就不存在单位根；否则就会存在单位根。对原序列检验结果如下所示：

表 1　医药 300 指数原序列单位根检验结果

指标	值
Test Statistic	0.232
p-value	0.974
Lags Used	3
Number of Observations Used	278
Critical Value (1%)	-3.454
Critical Value (5%)	-2.872
Critical Value (10%)	-2.572
是否平稳(是/否):	否

原假设为存在单位根，根据表 1 可以得到，在检验水平为 0.01 的条件下，接受原假设，原序列存在单位根，序列不平稳。故需要对原序列进行平稳化调整。

3. 差分调整

差分是调整平稳序列的重要方式，本报告对原序列采用了一阶差分。差分之后数据的时序图如下所示：

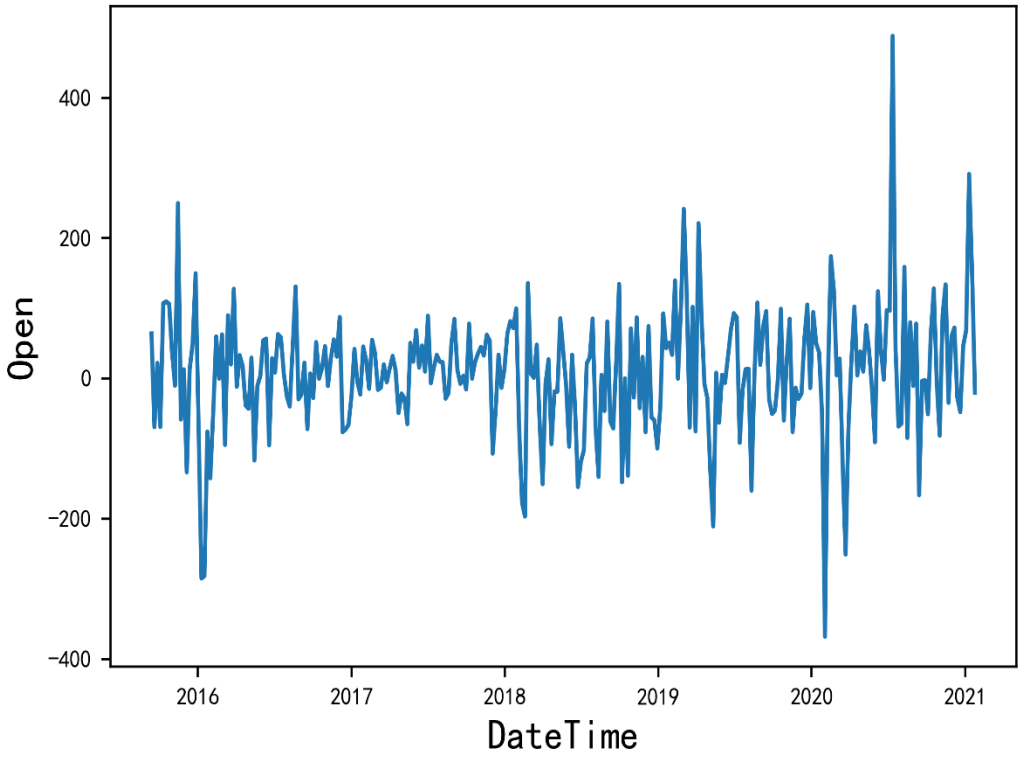

图 2　医药 300 股票指数异界差分后时间序列图

从时序图中初步可以发现，经过差分调整之后序列变得平稳有界。进一步通过单位根检验，结果如下：

表 2　医药 300 股票指数一阶差分后单位根检验结果

指标	值
Test Statistic	-13.676
p-value	0.000
Lags Used	0
Number of Observations Used	280
Critical Value (1%)	-3.454
Critical Value (5%)	-2.872
Critical Value (10%)	-2.572
是否平稳(是/否):	是

从表中可以看出，经过一阶差分之后，单位根检验结果表面，拒绝原假设。差分后序列平稳。可以进行下一步的时间序列分析。

(二) 纯随机检验

纯随机检验也称为白噪声检验,是专门用来检验序列是否为纯随机序列的一种方法。本报告中计算了差分后序列的 Q 统计量和 Ljung-Box 统计量[1]，结果如下:

表 3 医药 300 股票指数差分序列的纯随机检验结果

滞后阶数	AC 统计量	Q 统计量	P 值(>Q)
1	0.196	10.922	0.001
2	0.013	10.973	0.004
3	-0.127	15.595	0.001
4	0.003	15.597	0.004
5	-0.053	16.415	0.006
6	0.068	17.738	0.007
7	0.019	17.847	0.013
8	0.037	18.235	0.02
9	-0.014	18.29	0.032
10	-0.008	18.311	0.05
11	-0.007	18.324	0.074
12	-0.11	21.92	0.038
13	-0.007	21.932	0.056
14	-0.067	23.274	0.056
15	0.024	23.452	0.075
16	-0.062	24.614	0.077
17	0.062	25.788	0.078
18	0.037	26.206	0.095
19	0.08	28.173	0.08
20	0.077	29.971	0.07
21	0.039	30.439	0.084
22	-0.031	30.742	0.102
23	-0.174	40.052	0.015
24	-0.08	42.015	0.013

由于在各阶延迟下 Q 统计量的 P 值都非常小（<0.05)，所以差分后序列显著拒绝纯随机的原假设。因而可以认为医药 300 股票指数的变动不属于纯随机波动，也即序列属于非白噪声序列。

二、 ARIMA 建模

(一) 图示定阶

通过两图观察得到，自相关图显示滞后 1 阶超出了置信边界；偏相关图显示在滞后 1 阶时的偏自相关系数超出了置信边界，但是自相关图和偏自相关图在滞后 3 阶的时候也有略微的超出置信边界[2]。

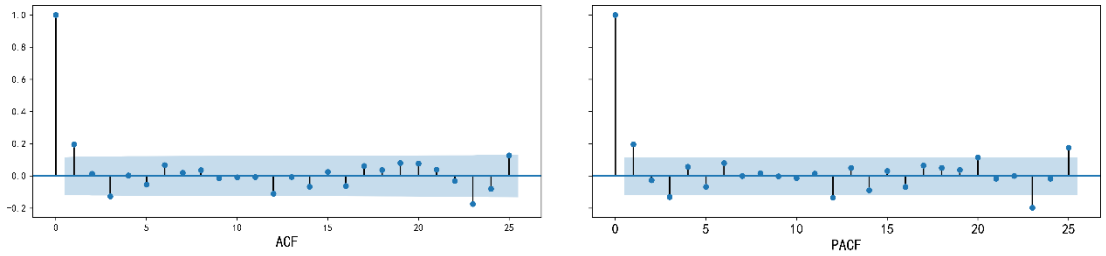

图 3 差分序列的自相关图和偏自相关图

注意在滞后阶数较高时也存在较高相关性，此次可能的原因是因为序列中存在季节周期性，本报告暂不考虑其实的季节周期性。则有以下模型可以供选择：

- ARMA(1,0)模型：即自相关图在滞后 1 阶之后缩小为 0，且偏自相关缩小至 0，则是一个阶层 p=1 的自回归模型；
- ARMA(0,1)模型：即偏自相关图在滞后 1 阶之后缩小为 0，且自相关缩小至 0，则是一个阶层 q=1 的移动平均模型；
- ARMA(1,1)模型：即自相关图在滞后 1 阶之后缩小为 0，且偏自相关在滞后 1 阶滞后缩小至 0，则是一个阶数 p=1、q=1 的自回归移动平均混合模型；
- ARMA(3,3)模型：即自相关图在滞后 3 阶之后缩小为 0，且偏自相关在滞后 3 阶滞后缩小至 0，则是一个阶数 p=3、q=3 的自回归移动平均混合模型。

(二) 自动定阶

由于自相关图和偏自相关图的定阶方法存在一定的主观性,故可以采用客观的模型选择准则来确定阶数。本报告主要采用三种准则：AIC、BIC、HQIC。

AIC=-2 ln(L) + 2 k：赤池信息量；BIC=-2 ln(L) + ln(n)*k：贝叶斯信息量；HQIC=-2 ln(L) + ln(ln(n))*k。构造这些统计量所遵循的统计思想是一致的，就是在考虑拟合残差的同时，依自变量个数施加“惩罚”。但要注意的是，这些准则不能说明某一个模型的精确度，也即是说，对于三个模型A，B，C，我们能够判断出C模型是最好的，但不能保证C模型能够很好地刻画数据，因为有可能三个模型都是糟糕的[3]。

计算最大 p、q 为 3 以内的所有组合情况的模型。得到各准则信息量结果如下：

表 4 自动定阶各准则结果

模型	AIC	BIC	HQIC
ARMA (0,0)	3319.623	3326.900	3322.541
ARMA (1,0)	3310.622	**3321.537**	3315.000
ARMA (2,0)	3312.417	3326.970	3318.254
ARMA (3,0)	3309.394	3327.585	3316.690
ARMA (0,1)	3311.018	3321.933	**3315.396**
ARMA (1,1)	3312.536	3327.089	3318.373
ARMA (2,1)	3313.126	3331.318	3320.422
ARMA (3,1)	3308.681	3330.511	3317.436
ARMA (0,2)	3311.818	3326.371	3317.655
ARMA (1,2)	3308.073	3326.264	3315.368
ARMA (2,2)	3313.222	3335.053	3321.978
ARMA (3,2)	3310.680	3336.149	3320.895
ARMA (0,3)	3308.923	3327.115	3316.219
ARMA (1,3)	3308.540	3330.370	3317.295
ARMA (2,3)	3310.289	3335.758	3320.504
ARMA (3,3)	**3304.805**	3333.912	3316.479

由 AIC 准则可以得到，最优阶数为 ARMA（3，3），AIC 信息量为：3304.805；

由 BIC 准则确定的最优模型为 ARMA（1，0），BIC 信息量为：3321.537；由 HQIC 信息量确定的最有模型为 ARMA（0，1），HQIC 信息量为：3315.396。综合考虑之后，本报告选择了模型 ARMA（3，3）。

(三) 模型结果

由于在数据处理中进行了一阶差分，故实际模型为 ARIMA（3，1，3）。通过 python 软件得到模型的拟合结果如下：

表 5 医药 300 股票指数 ARIMA（3，1，1）模型结果

模型	系数	标准误	Z 统计量	P>|z|	[0.025	0.975]
const	6.8140	3.004	2.268	0.023	0.926	12.702
ar.L1.D.Open	-0.6203	0.068	-9.127	0.000	-0.753	-0.487
ar.L2.D.Open	0.8540	0.034	25.034	0.000	0.787	0.921
ar.L3.D.Open	0.7213	0.065	11.062	0.000	0.594	0.849
ma.L1.D.Open	0.7755	0.039	19.735	0.000	0.699	0.853
ma.L2.D.Open	-0.8312	0.022	-38.480	0.000	-0.874	-0.789
ma.L3.D.Open	-0.9444	0.035	-26.851	0.000	-1.013	-0.875

由上表可以发现，ARIMA(3,1,3)模型的各系数均通过显著性检验(0.05)，故模型结果为：

$$x_t = 6.814 - 0.6203x_{t-1} + 0.854x_{t-2} + 0.7213x_{t-3} + \epsilon_t + 0.7755\epsilon_{t-1} - 0.8312\epsilon_{t-2} - 0.9444\epsilon_{t-3}$$

$$\{\epsilon_t\} \sim WN(0, \sigma^2)$$

三、 模型诊断

(一) 残差正态性检验

观察 ARIMA 模型的残差是否是平均值为 0 且方差为常数的正态分布（服从零均值、方差不变的正态分布）。

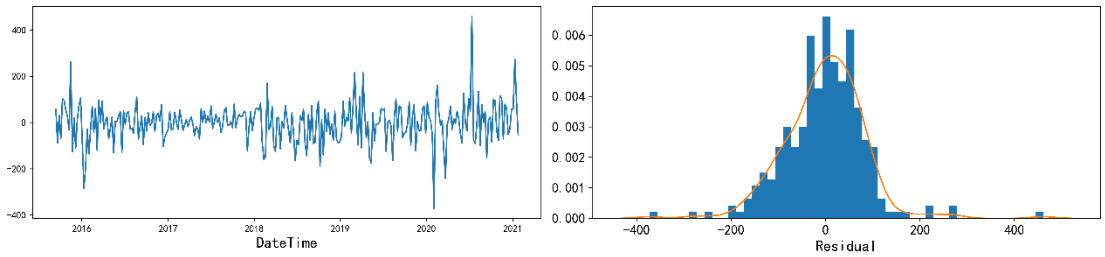

图 4　模型残差的序列图和核密度估计图

绘制模型残差的序列图，可以发现残差序列在 0 附近波动，且波动无异常，说明残差序列服从 0 均值同方差的分布。另绘制了残差序列的核密度估计，可以发现残差序列的直方图近似服从正态分布。

(二)　残差自相关检验

观察连续残差是否（自）相关。德宾-沃森检验,简称 D-W 检验，是目前检验自相关性最常用的方法，但它只使用于检验一阶自相关性。因为自相关系数 ρ 的值介于-1 和 1 之间，所以 0≤DW≤ 4 。并且 DW=0 时，则 ρ＝ 1 ，即存在正自相关性；DW＝ 4 时，则 ρ＝－ 1 ，即存在负自相关性；DW＝ 2 时，则 ρ＝ 0 ，即不存在（一阶）自相关性[4]。

表 6　D-W 检验参考标准

统计量	相关系数
DW=0	ρ=-1
DW=2	ρ=0
DW=4	ρ=1

当 DW 值显著的接近于 0 或 4 时，则存在自相关性，而接近于 2 时，则不存在（一阶）自相关性。这样只要知道DW统计量的概率分布，在给定的显著水平下，根据临界值的位置就可以对原假设进行检验。在本案例中对残差序列进行 D-W 检验得到 D-W 统计量为：

$$DW = 1.7946$$

故可以认为在一阶情况下，残差序列近似不存在自相关性。进一步绘制残差序列的自相关图和偏自相关图如下：

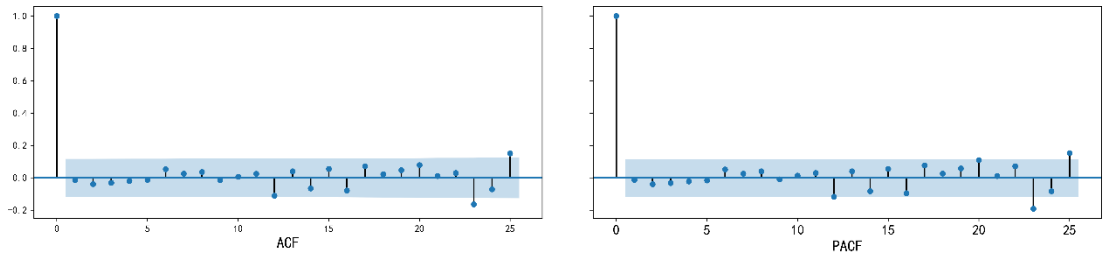

图 5　模型残差自相关图和偏自相关图

从图中可以发现，残差序列的自相关图和偏自相关图在滞后各阶的情况下都没有超过两倍标准差的参考界限。所以可以认为模型残差序列不存在自相关性。

(三)　残差纯随机检验

对于 ARIMA 模型，其残差被假定为高斯白噪声序列，所以当用 ARIMA 模型去拟合数据时，拟合后我们要对残差的估计序列进行 Ljung-Box 检验，判断其是否是高斯白噪声，如果不是，那么就说明 ARIMA 模型也许并不是一个适合样本的模型。对本报告得到的模型残差进行纯随机检验，如果模型建立可行，则残差应为一个白噪声序列[4]。进行的纯随机检验结果如下所示：

表 7　残差序列的纯随机检验结果

滞后阶数	AC 统计量	Q 统计量	P 值(>Q)
1	-0.014	0.054	0.816
2	-0.039	0.484	0.785
3	-0.031	0.757	0.86
4	-0.019	0.862	0.93
5	-0.013	0.909	0.97
6	0.053	1.722	0.943
7	0.026	1.914	0.964
8	0.036	2.283	0.971
9	-0.015	2.347	0.985
10	0.007	2.36	0.993

11	0.025	2.537	0.996
12	-0.111	6.185	0.906
13	0.04	6.667	0.918
14	-0.066	7.957	0.892
15	0.055	8.854	0.885
16	-0.078	10.689	0.828
17	0.072	12.244	0.785
18	0.022	12.389	0.827
19	0.048	13.074	0.835
20	0.078	14.938	0.78
21	0.011	14.974	0.824
22	0.03	15.256	0.851
23	-0.164	23.498	0.432
24	-0.071	25.037	0.404

由于在各阶延迟下 Q 统计量的 P 值都非常大（>0.05），所以模型残差序列显著不拒绝纯随机的原假设。因而可以认为残差序列的波动属于纯随机波动，也即建立的 ARIMA 模型可行。

四、 短期预测

至此医药 300 股票指数的 ARIMA 模型已经建立完成，为了进一步评估模型的效果，需要对原序列进行预测评价。首先在原序列的基础上进行回代预测，然后对 2021 年 1 月份之后的 4 个月进行预测。预测效果如下表所示：

表 8　ARIMA（3，1，3）模型对医药 300 股票指数预测结果

日期时间	真实值	预测值	预测误差 1
2021-01-24 00:00:00	5491.876	5453.090	-38.7859
2021-01-31 00:00:00	5507.026	5443.206	-63.8204
2021-02-07 00:00:00	5446.906	5427.800	-19.1063
2021-02-14 00:00:00	5599.997	5417.557	-182.439
2021-02-21 00:00:00	5828.045	5404.187	-423.858
2021-02-28 00:00:00	5551.608	5394.081	-157.527
2021-03-07 00:00:00	5350.066	5382.271	32.2054
2021-03-14 00:00:00	5118.246	5372.581	254.3353
2021-03-21 00:00:00	5083.284	5362.031	278.7466
2021-03-28 00:00:00	4981.776	5352.905	371.1291
2021-04-04 00:00:00	5074.220	5343.416	269.1959

2021-04-11 00:00:00	5124.650	5334.928	210.2779
2021-04-18 00:00:00	4971.858	5326.366	354.5081
2021-04-25 00:00:00	5058.108	5318.544	260.4359
2021-05-02 00:00:00	5118.368	5310.814	192.4458

进一步绘制预测结果的折线图如下，黑色的线代表实际的股票指数数据，红色的线代表对建模所用数据进行回代预测，蓝色的线代表对未来 4 个月进行预测。

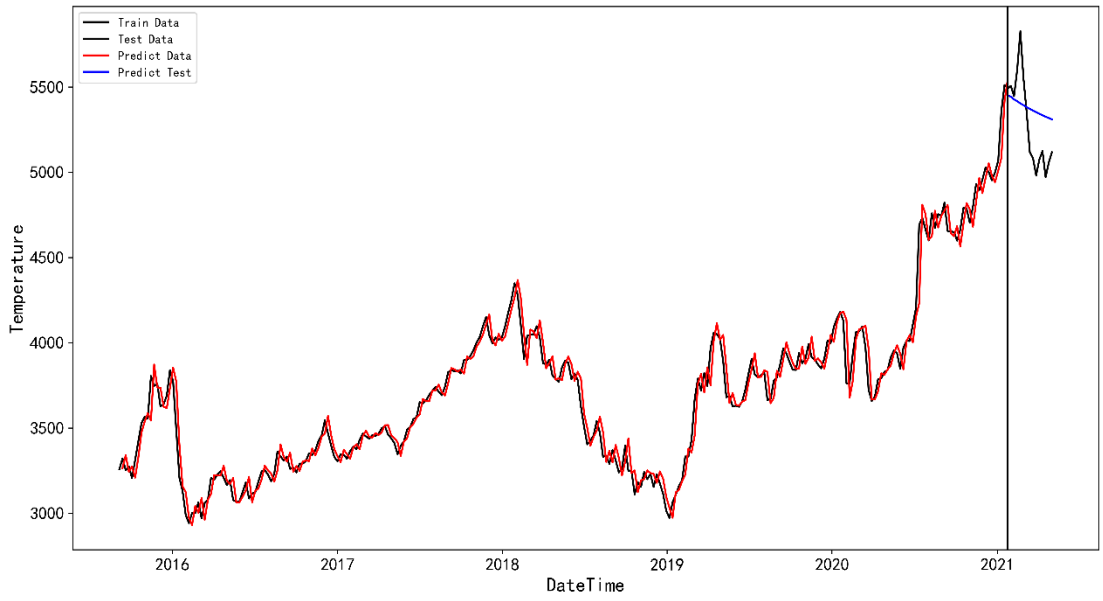

图 6　医药 300 股票指数时间序列预测效果图

从图形来看，在测试集数据上预测效果拟合较好，但是在未来数据的测试集预测效果较为一般，但是也能对大体的走势进行一个拟合。这里的主要原因是由于股票数据的波动较大，采用普通的 ARIMA 模型进行预测往往可能出现差异较大的情况。且在本案例报告中未考虑季节因素的影响，故建立的 ARIMA 模型对序列的季节波动难以很好的拟合。另外通过计算预测效果的均方误差得：

表 9　ARIMA 模型预测效果的均方误差

数据	均方误差
2015 年 5 月-2021 年 1 月	84.121
2021 年 1 月-2021 年 5 月	310.289

随着对未来预测的期数越长，总体来说预测效果越差。但是在对建模数据的

回代拟合上来看，效果较好。即 2015 年 5 月-2021 年 1 月上拟合的均方误差仅为 84.121；而在 2021 年 1 月-2021 年 5 月预测的均方误差为 310.289。

五、 结果评价

至此本报告针对医药 300 股票指数的时间序列建模及预测已全部完成。总体上看，所建立的 ARIMA（3，1，3）模型对数据的拟合效果较好，对预测未来整体股票走向上有一定的效果。但是由于没有考虑股票指数存在的季节性因素，且股票指数自身存在巨大的波动变化，这种波动难以用具体的统计模型拟合，所以模型对股票指数序列的具体数据预测上效果较差。

六、 参考文献

1. 赵康银 and 薛亚楠, *基于 ARIMA 模型的 TCL 集团股票预测及评价*. 广西质量监督导报, 2020(09): p. 144-145.

2. 闫宇 and 吴海涛, *基于 ARIMA 模型的纳斯达克指数短期预测*. 信息与电脑(理论版), 2020. **32**(20): p. 155-158.

3. 黄莉霞, *基于ARIMA模型的股价分析与预测——以中国平安为例*. 科技经济市场, 2020(10): p. 62-63.

4. 石琳枫, *基于 ARIMA 模型的万科公司股票收盘价研究*. 商展经济, 2021(03): p. 50-52.

七、 附录：Python 代码

```
import re
import os
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
import statsmodels.api as sm
import statsmodels.tsa.stattools as ts
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.arima_model import ARIMA
import warnings
plt.style.use("default")
plt.rcParams['font.sans-serif'] = ['SimHei']     # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False       # 用来正常显示负号
warnings.filterwarnings("ignore")                  # 忽略警告
os.makedirs("./output",exist_ok=True)              # 文件配置

# 导入数据
series=pd.read_csv("./husheng300.csv",encoding="UTF8",index_col="Date",parse_dates=["Date"]).O
pen
data = series.resample("W").mean().bfill()
data.isna().sum()

# 绘制时序图
fig = plt.figure(figsize=(12,8))
plt.plot(data)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.xlabel("DateTime", fontsize = 16)
plt.ylabel("Open", fontsize = 16)
plt.savefig("output/TimeSeries.png", dpi = 500, format = "png", bbox_inches = "tight")
plt.show()

# 重采样
#划分训练集
index = data.index[int(len(data)*0.95)]
```

```
train_data = data[:int(len(data)*0.95)+1]
test_data = data[int(len(data)*0.95):]

#可视化
# 绘制时序图
fig = plt.figure(figsize=(12,8))
plt.plot(train_data,"r",label="Train Data")
plt.plot(test_data,"g",label="Test Data")
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.xlabel("DateTime", fontsize = 16)
plt.ylabel("Open", fontsize = 16)
plt.axvline(x=index,ls="-",c="black")#添加垂直直线
plt.legend(loc="best")
plt.savefig("output/TimeSeries2.png", dpi = 500, format = "png", bbox_inches = "tight")
plt.show()

# 季节性分解
decomposition = seasonal_decompose(train_data)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

fig,axes = plt.subplots(4,1,sharex=True,figsize=(12,8))
axes[0].plot(train_data)
axes[0].set_ylabel("original")
axes[1].plot(trend)
axes[1].set_ylabel("trend")
axes[2].plot(seasonal)
axes[2].set_ylabel("seasonal")
axes[3].plot(residual)
axes[3].set_ylabel("residual")
plt.xlabel("DateTime",fontsize=14)
plt.subplots_adjust(hspace=0.2)
plt.savefig("output/decomposition.png", dpi = 500, format = "png", bbox_inches = "tight")
plt.show()

#数据平稳性检测
# 用 Dickey-Fuller test 检验序列是否平稳
def judge_stationarity(data_sanya_one):
    dftest = ts.adfuller(data_sanya_one)
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of
Observations Used'])
    stationarity = "是"
```

```
        for key, value in dftest[4].items():
            dfoutput['Critical Value (%s)'%key] = value
            if dftest[0] > value:
                    stationarity = "否"
        print(dfoutput)
        print("是否平稳(是/否): %s" %(stationarity))
        return stationarity
stationarity = judge_stationarity(train_data)

# 季节差分
differenced = train_data.diff(1).dropna()
plt.plot(differenced)
plt.xticks(fontsize = 8)
plt.yticks(fontsize = 8)
plt.xlabel("DateTime", fontsize = 14)
plt.ylabel("Open", fontsize = 14)
plt.savefig("output/SeasonDiff.png", dpi = 500, format = "png", bbox_inches = "tight")
plt.show()
stationarity = judge_stationarity(differenced)

# 季节性分解
decomposition = seasonal_decompose(differenced)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
fig,axes = plt.subplots(4,1,sharex=True,figsize=(12,8))
axes[0].plot(train_data)
axes[0].set_ylabel("origon")
axes[1].plot(trend)
axes[1].set_ylabel("trend")
axes[2].plot(seasonal)
axes[2].set_ylabel("seasonal")
axes[3].plot(residual)
axes[3].set_ylabel("residual")
plt.xlabel("Date",fontsize=14)
plt.subplots_adjust(hspace=0.2)
plt.savefig("output/decomposition2.png", dpi = 500, format = "png", bbox_inches = "tight")
plt.show()

r,q,p = sm.tsa.acf(differenced.squeeze(), qstat=True)
data = np.c_[range(1,41), r[1:], q, p]
table = pd.DataFrame(data, columns=['lag', "AC", "Q", "Prob(>Q)"])
test_stochastic1 = round(table.set_index('lag'),3)
test_stochastic1
```

```
#查看 acf 与 pacf 确定 q 和 p
fig, axes = plt.subplots(1,2, sharey=True,figsize = (20,4))
plot_acf(differenced, ax=axes[0])
plot_pacf(differenced, ax=axes[1])
axes[0].set_xlabel("ACF",fontsize=16)
axes[0].set_title("")
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
axes[1].set_xlabel("PACF",fontsize=16)
axes[1].set_title("")
plt.yticks(fontsize=14)
plt.xticks(fontsize=14)
plt.subplots_adjust(wspace=0.1)
plt.savefig("./output/acf_pacf.png",dpi=500, bbox_inches = "tight")
plt.show()

# 定阶
#在 statsmodels 包里还有更直接的函数：
order = ts.arma_order_select_ic(differenced,max_ar=3,max_ma=3,ic=['aic', 'bic', 'hqic'])
orderAIC = pd.melt(order["aic"].reset_index(), id_vars=["index"]).rename(columns = {"index":"AR","variable":"MA","value":"AIC"})
orderBIC = pd.melt(order["bic"].reset_index(), id_vars=["index"]).rename(columns = {"index":"AR","variable":"MA","value":"BIC"})
orderHQIC = pd.melt(order["hqic"].reset_index(), id_vars=["index"]).rename(columns = {"index":"AR","variable":"MA","value":"HQIC")
orderAIC["order"] = "ARMA(" + orderAIC["AR"].astype(str) +"," + orderBIC["MA"].astype(str) + ")"
orderBIC["order"] = "ARMA(" + orderAIC["AR"].astype(str) +"," + orderBIC["MA"].astype(str) + ")"
orderHQIC["order"] = "ARMA(" + orderAIC["AR"].astype(str) +"," + orderBIC["MA"].astype(str) + ")"
orderAIC.drop(["AR","MA"],axis=1,inplace=True)
orderBIC.drop(["AR","MA"],axis=1,inplace=True)
orderHQIC.drop(["AR","MA"],axis=1,inplace=True)
print(order.aic_min_order)
print(order.bic_min_order)
print(order.hqic_min_order)
orderdf = orderAIC.merge(orderBIC,on="order").merge(orderHQIC,on="order")
order_result = round(orderdf[["order","AIC","BIC","HQIC"]],3)
mod = sm.tsa.ARIMA(train_data,order=(2, 1, 3))
model = mod.fit()
result_df = model.summary()
table1 = pd.DataFrame(result_df.tables[0])
table2 = pd.DataFrame(result_df.tables[1])
table3 = pd.DataFrame(result_df.tables[2])
result_df
```

```
# # 自动定阶
# pdq = list(itertools.product(range(0,3), range(0,2), range(0,3)))
# seasonal_pdq = [(x[0], x[1], x[2], 52) for x in list(pdq)]

# results = {}
# modelpara,AIC,BIC,HQIC = [],[],[],[]
# for param_seasonal in seasonal_pdq:
#     try:
#         mod = sm.tsa.statespace.SARIMAX(train_data,
#                                         order=(1,1,2),
#                                         seasonal_order=param_seasonal,
#                                         enforce_stationarity=False,
#                                         enforce_invertibility=False).fit()
#         modelpara.append("ARIMA(1,1,2)x{}".format(param_seasonal))
#         AIC.append(mod.aic)
#         BIC.append(mod.bic)
#         HQIC.append(mod.hqic)
#             print('ARIMA{}x{} - AIC:{} - BIC:{} - HQIC:{}'.format(param, param_seasonal,
mod.aic,mod.bic,mod.hqic))
#     except:
#         continue
# results["ARIMA"] = modelpara
# results["AIC"] = AIC
# results["BIC"] = BIC
# results["HQIC"] = HQIC
# results = pd.DataFrame(results)
# results

#可视化
fig, axes = plt.subplots(1,2,figsize = (20,4))
axes[0].plot(model.resid,label = "Model Residual")
axes[1].hist(model.resid,bins = 50, normed=True)
sns.kdeplot(model.resid,shade=False)
axes[0].set_xlabel("DateTime",fontsize=16)
axes[0].set_title("")
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
axes[1].set_xlabel("Residual",fontsize=16)
axes[1].set_title("")
plt.yticks(fontsize=14)
plt.xticks(fontsize=14)
plt.subplots_adjust(wspace=0.1)
plt.savefig("./output/Residual.png",dpi=500, bbox_inches = "tight")
```

```
plt.show()
x = sm.stats.durbin_watson(model.resid.values)

#查看 acf 与 pacf 确定 q 和 p
fig, axes = plt.subplots(1,2, sharey=True,figsize = (20,4))
plot_acf(model.resid, ax=axes[0])
plot_pacf(model.resid, ax=axes[1])
axes[0].set_xlabel("ACF",fontsize=16)
axes[0].set_title("")
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
axes[1].set_xlabel("PACF",fontsize=16)
axes[1].set_title("")
plt.yticks(fontsize=14)
plt.xticks(fontsize=14)
plt.subplots_adjust(wspace=0.1)
plt.savefig("./output/residual_acf_pacf.png",dpi=500, bbox_inches = "tight")
plt.show()

r,q,p = sm.tsa.acf(model.resid.values.squeeze(), qstat=True)
data = np.c_[range(1,41), r[1:], q, p]
table = pd.DataFrame(data, columns=['lag', "AC", "Q", "Prob(>Q)"])
test_stochastic2 = round(table.set_index('lag'),3)
test_stochastic2

predict_train = model.predict(typ='levels')
# predict_test = model.forecast(int(len(data)*0.15))[0]
predict_test = pd.Series(model.forecast(int(len(data)*0.05+1))[0],index=test_data.index)

#可视化
#  绘制时序图
fig = plt.figure(figsize=(15,8))
plt.plot(train_data,"black",label="Train Data")
plt.plot(test_data,"black",label="Test Data")
plt.plot(predict_train,"red",label = "Predict Data")
plt.plot(predict_test,"blue",label = "Predict Test")
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.xlabel("DateTime", fontsize = 16)
plt.ylabel("Temperature", fontsize = 16)
plt.axvline(x=index,ls="-",c="black")#添加垂直直线
plt.legend(loc="best")
plt.savefig("output/PredictResultAll.png", dpi = 500, format = "png", bbox_inches = "tight")
print('Train RMSE: %.4f'% np.sqrt(np.sum((predict_train-train_data)**2)/train_data.size))
```

```
print('Test RMSE: %.4f'% np.sqrt(np.sum((predict_test-test_data)**2)/test_data.size))
plt.show()


error = pd.DataFrame({"Real Data":test_data,"Predict Data":predict_test,"Predict Residual":predict_test-test_data})
with pd.ExcelWriter("./output/df_result.xls") as writer:
    test_stochastic1.to_excel(writer,sheet_name="stochastic1",index=True)
    table1.to_excel(writer,sheet_name="table1")
    table2.to_excel(writer,sheet_name="table2")
    table3.to_excel(writer,sheet_name="table3")
    order_result.to_excel(writer,sheet_name="order_result",index=False)
    test_stochastic2.to_excel(writer,sheet_name="stochastic2",index=True)
    error.to_excel(writer,sheet_name="error",index=True)
```