西南财经大学
SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS

# 本科课程论文

项目名称：基于突发事件的网络搜索数据对股价影响预测分析

组长姓名：何亦淼

组员姓名：阙志宏、姚真珍、龚小曼、胡琳悦、张子若

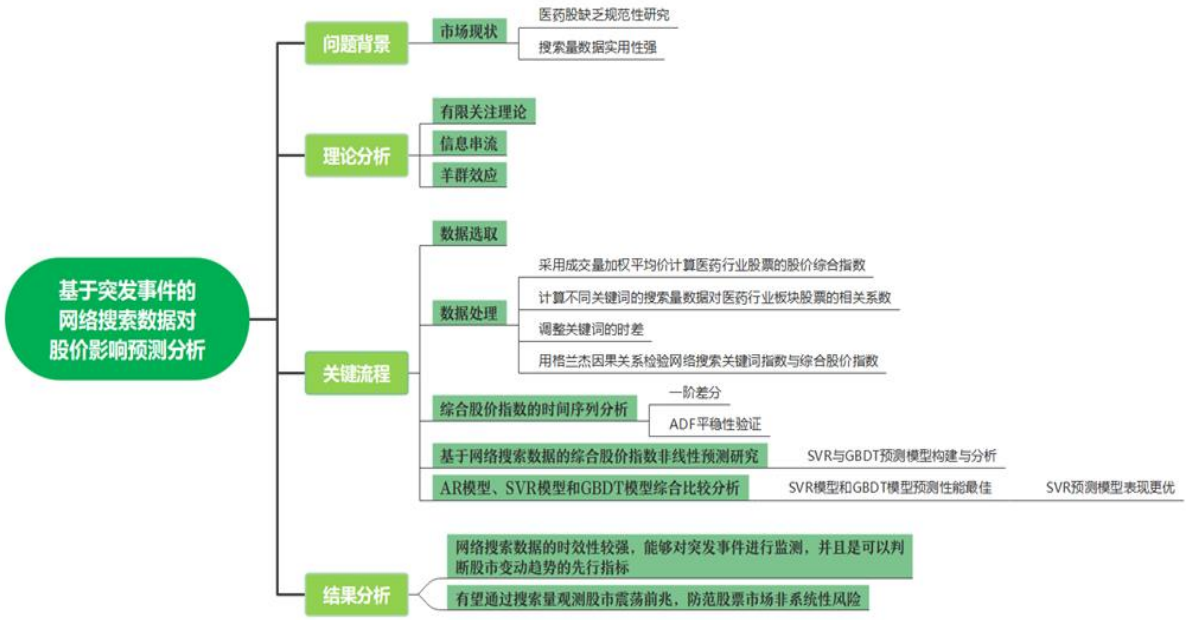
所在学院：金融学院

所在专业：金融工程

指导教师：田正磊

二〇二二年六月

# 一、思维导图




# 二、问题背景

## 2.1 市场现状

### 2.1.1 缺乏规范性研究

我国医药行业正处在黄金阶段，医疗技术水平不断提升，相关政策扶持也促进行业高速地发展。但是于此同时，医药行业也暴露出一系列的问题，药品安全隐患层出不穷，如 2012 年修正集团的“毒胶囊”事件，健康元采购“地沟油”制药事件，2016 年的山东非法疫苗案，再到 2018 年的鸿茅药酒事件、令人震惊的长春长生问题疫苗案、权健事件……这些负面事件的发生，每次都会对医药行业造成巨大地冲击，并在股票市场上得以反映。医药行业股票以其良好的风险抵抗力，被称之为“防御性板块”，即在大盘震荡时其也能保持较为稳定的波动性，因此受到投资者的青睐，但是关于医药行业股市波动性的规范研究相对缺乏。我们必须重视到医药制造业不仅关系到我国人民的切身利益，更是对国家未来发展有着重要的意义。

### 2.1.2 搜索量数据实用性强

一直以来医疗安全保障都是涉及到民生的社会问题,每当这些突发事件产生，随之都会带来巨大地社会关注与舆论。在互联网经济的背景下，网络与信息技术爆炸性地发展，人们获取信息地方式多种多样，但毫无疑问搜索引擎是其中最重要的渠道之一，根据中国互联网络信息中心(CNNIC）统计报告显示截至 2019 年 6 月，中国网民规模达 8.54 亿，互联网普及率为 61.2%，使用搜索引擎用户规模达 6.95 亿;搜索引擎具有能够将用户搜索信息记录储存下来的特点，为股市的相关研究提供如重大突发事件、股票市场行情等网络搜索量数据。近年来在社会经济行为的预测研究中，网络搜索数据受到广泛地应用，因其相对于传统的统计局数据具备成本更低、时效性更强的优势，且能够得到较好的预测效果，可行性被充分证实。

因此，我们将搜索量作为预测股价的一个因子，并研究其对于股价估值准确性的影响。

# 三、理论分析

在具体操作之前，我们首先分析搜索量作为股价因子的可行性。在行为金融学领域进行分析，突发事件会对人的认知、情绪造成偏误，从而造成非理性行为，导致股价偏离真实价值。

行为金融学发展至今，结合人们的行为心理探寻基金股市现象已经成为许多学者日以继夜研究的目标。以下是我们展开研究主要基于的几个行为金融学理论：

## 3.1 有限关注理论：

人类在处理信息和同时进行多项任务的时候,存在着能力上的局限性,这表现为关注现象。目前,关注现象对于决策制定过程的影响,广泛存在于经济和心理研究领域。心理学家认为关注是指一种稀有的认知资源。

Simon（1955）认为人们在进行经济决策时具有有限的处理能力。Kahneman(1973)指出有限关注是在一个具有大量信息环境下的必然结果，当个体需要同时处理多个任务时，注意力会受到扰乱，关注的有限性会导致其对信息的处理效率降低。在股票市场上，有限关注具体表现为由于时间和精力的有限性，使得投资者不可能考虑所有的股票投资，对信息的分析能力会受到一定的约束（Aboody 等,2010)；投资者因不具备充分的处理和吸收所有可得信息的精力和能

力，会导致对联系股票基本面的相关信息反应不足（Engelhergetal.2009)。

以搜索量对投资者的关注进行替代，加入搜索量因子，使用新的模型对股票进行定价，使得定价更加合理。

### 3.2 信息串流

信息串流是指人们在决策是都会参考其他人的选择，而忽略自己已有的信息或可获得的信息。信息串流理论刻画了大量信息在传播与评估中的丢失现象。

由于人们注意力的限制，只能关注那些热点信息，并形成相似的信念，而人们的交流以及媒体的宣传使得这些信念得到进一步加强。

在信息串流过程中，集体信念得以逐步形成。信息串流模型还解释了为什么非常少的信息就可以引起社会潮流或时尚。

当突发事件出现后，人们在百度上对该事件进行搜索，获得的绝大部分信息均来自于媒体报道或相关官方信息披露。在此信息串流的过程中，集体信念得以逐步形成，绝大多数人对该板块股票的价格预测逐渐趋于一致，由此引发股价的波动。

### 3.3 羊群效应

羊群效应也就是从众行为，它是指人们在决策时把他人行为作为最重要的决策信息的心理，这种心理通常会导致市场行为的趋同，例如故事暴涨或暴跌现象。

即投资者在对相关事件进行搜索时，会受到相关事件的大众情绪的影响，大规模买入或卖出股票。通过搜索量对投资所接受的大众情绪进行预测，将使得对股价的预测更为精准。

## 四、关键流程

### 4.1 数据选取

#### 4.1.1 样本数据（搜索关键词数据）

谷歌趋势和百度指数是目前国内外用户群体基数最大的搜索引擎产品。由于谷歌广泛地运用于国际，国内网民更多地使用百度搜索引擎，所以此处使用百度指数提供的关键词网络搜索量数据进行分析。百度指数时间范围选取如下表，为了保持股票价格数据的一致性，对于关键词网络搜索量只保留交易日的数据，该

数据是绝对值的形式，所以已有的数据不会随着时间而变动。

根据具体事件设定相关联的初始关键词，所选取的事件为“山东非法疫苗事件”、“2017 版医保目录发布”、“CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》”、“药明康德闪崩”。

其次，利用百度指数对相关关键词的推荐功能扩充初始关键词库且仅选择与对应事件相关的关键词，建立相应突发事件的关键词库如下。

| 事件 | 时间点 | 时间区间（135 个交易日） | 关键词库 |
| --- | --- | --- | --- |
| 山东非法疫苗事件（ym） | 2016/3/19 | 2015/11/06-2016/05/24 | 疫苗,二类疫苗,疫苗事件,狂犬疫苗,脊灰疫苗,水痘疫苗,问题疫苗,非法疫苗,山东非法疫苗,山东疫苗事件,山东食品药品监督管理局,沃森生物,沃森生物股票,沃森生物股吧,300142,冷链物流,智飞生物 |
| 2017 版医保目录发布（yb） | 2017/2/23 | 2016/10/13-2017/05/02 | 医药,医药股,医药板块,医保目录,医保目录查询,医药股票有哪些,亿帆医药,益佰制药,益佰制药股票,人福医药,人福医药股票,康缘药业,康缘药业股票,海正药业,众生药业,康弘药业,葵花药业,易明医药 |
| 长春长生问题疫苗（cs） | 2018/7/16 | 2018/02/26-2018/09/07 | 长生生物疫苗,长生生物官网,长生生物股票,长春长生疫苗事件,长生生物股吧,狂犬疫苗造假,长春长生股票,长春长生,长春长生生物,长生生物疫苗事件,2680 |
| CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》（cde） | 2021/7/2 | 2021/02/22-2021/09/03 | 医药,医药股,医药板块,医药基金,cde,国家药监局,cro,cxo,etf,泰格医药,300347,复星医药,复星医药股吧 |
| 药明康德闪崩（ymkd） | 2021/12/15 | 2021/08/03-2022/02/24 | 医药,医药股,医药板块,医药基金,cxo,cxo 医药是什么意思,cro,药明康德,药明生物,药明康德 股票,药明康德股吧,603259,药明康德跌停,康龙化成,泰格医药,中欧医疗健康混合 c,百济神州,美国实体 |

| | | | |
|---|---|---|---|
| | | | 清单,美国制裁 |

表 1 突发事件关键词库

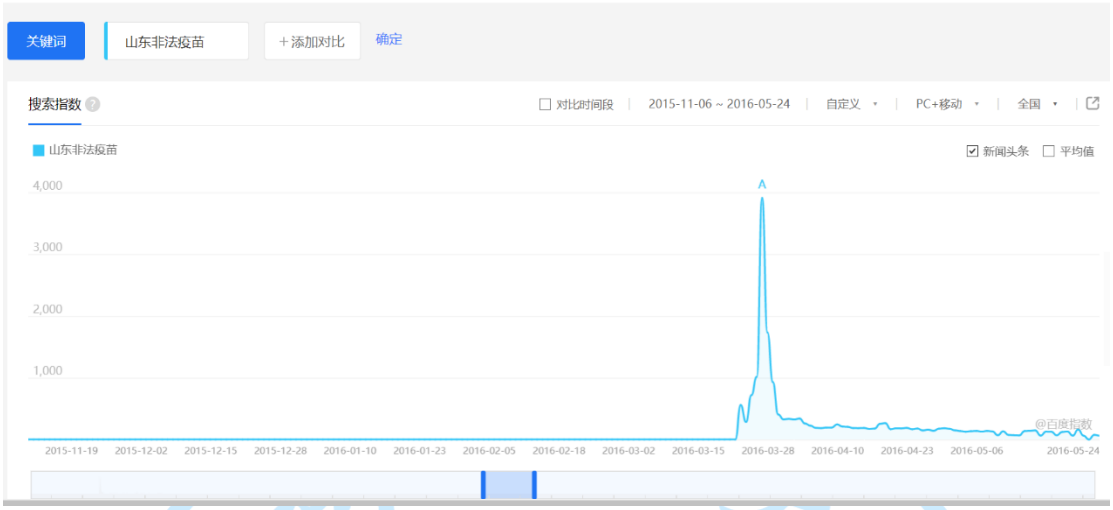



图 1 山东非法疫苗事件（ym）搜索指数

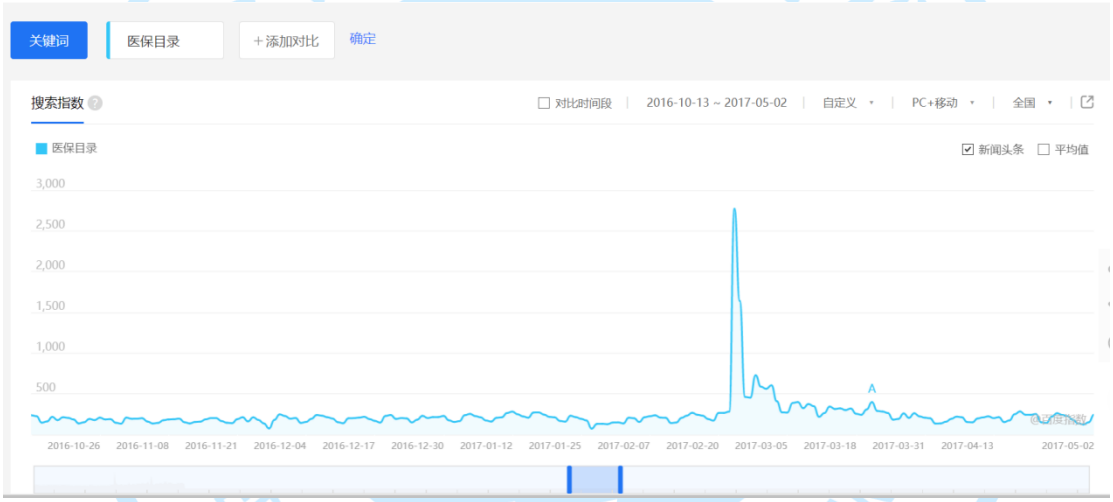



图 2 2017 版医保目录发布（yb）搜索指数

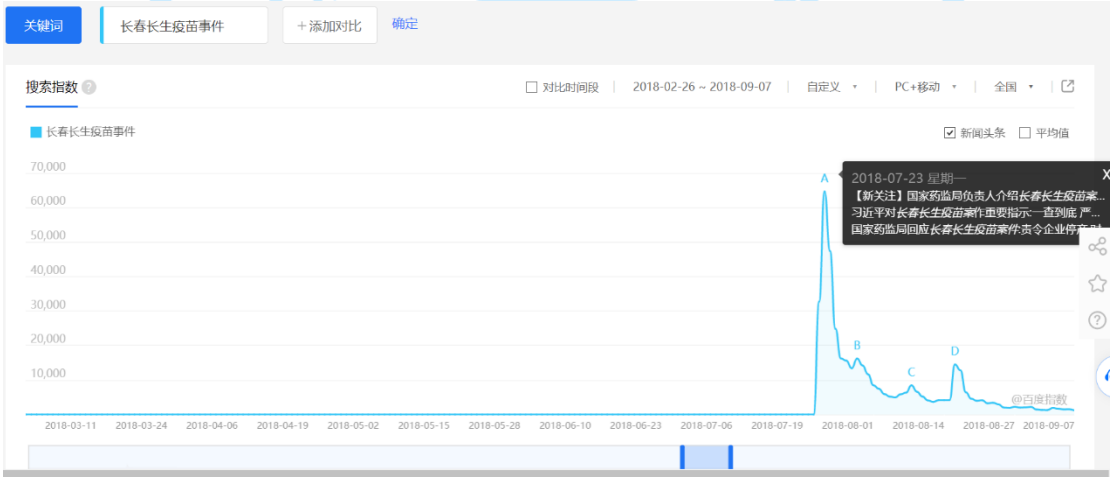



图 3 长春长生问题疫苗（cs）

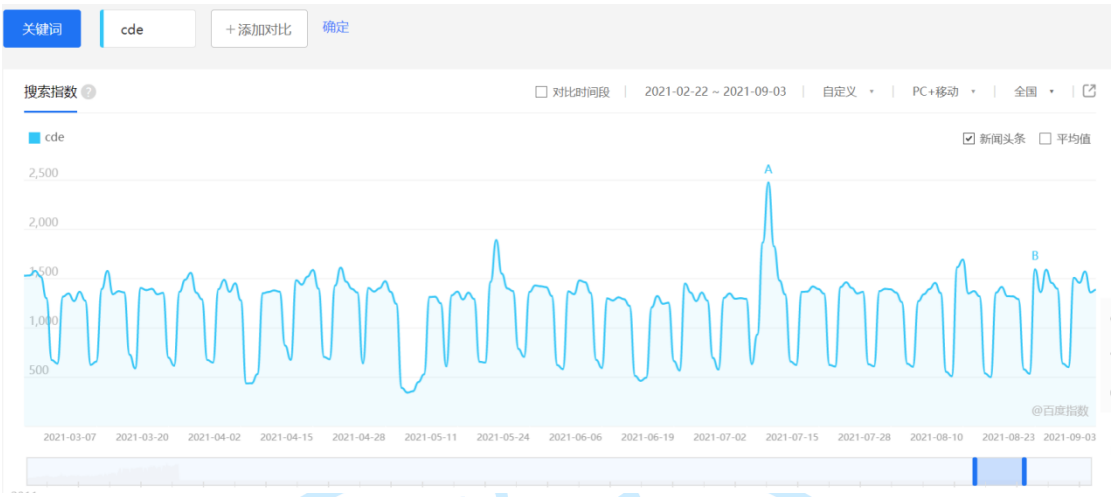



图 4 CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则

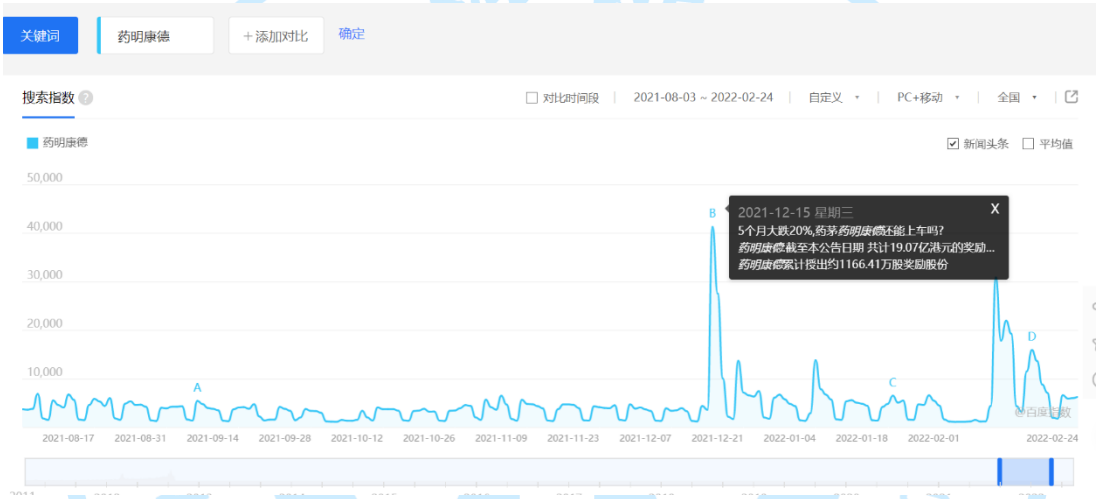



图 5 药明康德闪崩

### 4.1.2 股票数据

从国泰安（CSMAR）数据库选取深沪 A 股中医药行业板块的 224 只股票，对应的时间范围分别为：山东非法疫苗事件：2015 年 11 月 6 日-2016 年 5 月 24 日，2017 版医保目录发布：2016 年 10 月 13 日-2017 年 5 月 2 日，长春长生问题疫苗：2018 年 2 年 26 日-2018 年 9 月 7 日，CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》:2021 年 2 月 22 日-2021 年 9 月 3 日，药明康德闪崩：2021 年 8 月 3 日-2022 年 2 月 24 日，均取 135 个股票交易日数据。

## 4.2 数据处理

### 4.2.1 股价综合指数

采用成交量加权平均价（VWAP，Volume-weighted Average Price）计算医药

行业股票的股价综合指数，计算公式如下，其中$P_i$为样本个股的日收盘价，$V_i$为日个股交易股数，同时对其进行自然数处理，避免异方差问题的影响，记为ln(VWAPt):

$$\ln\left(VWAP_t\right)=\ln\left(\sum_{i=1}^{n}V_{it}P_{it}/\sum_{i=1}^{n}V_{it}\right)$$

### 4.2.2 搜索关键词指数合成

根据选定的关键词搜索量数据对医药行业板块股票 ln(VWAPt)的相关性进行计算检验，选取相关系数较高的关键词，此类关键词与股票收益率趋势变动具有相似性。同时考虑到相关系数的稳定性问题，对每个关键词计算 8 次相关系数，即由当天医药行业板块股票 ln(VWAPt)与每个关键词提前几天的搜索量数据分别得出相关性，相关性计算公式如下所示，其中相关系数 rl 表示 l 为对应阶数，阶数最大为 8，x 为关键词搜索量数据，y 是医药行业板块股票 ln(VWAPt)，$\bar{x}$，$\bar{y}$为均值。

$$r_l=\frac{\sum_{t=l}^{n}(x_{t-l}-\bar{x})(y_t-\bar{y})}{\sqrt{\sum_{t=l}^{n}(x_{t-l}-\bar{x})^2\sum_{t=l}^{n}(y_t-\bar{y})^2}}\quad l=0,\pm1,\ldots,\pm L$$

相关代码如下:

```
import os
import pandas as pd
import numpy as np
import math
#以交易量为权重对样本股票进行加权平均合成行业日均价序列
def lnPrice(fname):
    df = pd.read_csv(fname, names=["stock", "date", "price", "tradeCnt"], header=0)
    dateList = df["date"].tolist()
    dateList = list(set(dateList))
    dateList.sort(reverse=False)
    priceList = []
    lnPriceList = []
    for i in dateList:
        dailyPrice = 0
        dailyTrade = 0
        for j in range(len(df)):
            if df["date"][j] == i:
                dailyPrice += df["price"][j] * df["tradeCnt"][j]
                dailyTrade += df["tradeCnt"][j]
        priceList.append(dailyPrice/dailyTrade)
        lnPriceList.append(math.log(dailyPrice/dailyTrade))
    priceData = {"Date":dateList, "Price":priceList, "lnPrice":lnPriceList}
    dfp = pd.DataFrame(priceData)
    return dfp
#筛选交易日关键词搜索量加入序列
def searchCnt(dfp, fnames):
    dfs = pd.read_excel(fnames, sheet_name=0, names=["keyword", "area", "date", "search", "pc", "mobile"], header=0)
    searchList = []
    dateList = dfp["Date"].tolist()
    for i in dateList:
        for j in range(len(dfs)):
            if dfs["date"][j] == i:
                searchList.append(dfs["search"][j])
    dfp["searchCnt"] = searchList
    return dfp
```

```python
#得到K阶Pearson相关系数的最大值及对应阶数
def correlation(dfp, maxLag):
    corrdata = []
    for i in range(0, maxLag+1):
        xList = dfp["searchCnt"][:len(dfp)-i]
        yList = dfp["Price"][i:]
        x = np.array(xList)
        y = np.array(yList)
        rho = np.corrcoef(x, y)
        corrdata.append([rho[1][0], i])
    corrdata.sort(key=lambda x:math.fabs(x[0]), reverse=True)
    return corrdata[0]
```

### 4.2.3 相关性呈现及最终关键词选取

每个突发事件对应的关键词与股票相关性如下：

山东非法疫苗事件：

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 300142 | 0.629849371 | 1 |
| 二类疫苗 | 0.576620449 | 6 |
| 冷链物流 | 0.573985531 | 1 |
| 山东疫苗事件 | 0.539923246 | 1 |
| 山东非法疫苗 | 0.357178686 | 8 |
| 山东食品药品监督管理局 | -0.20258077 | 0 |
| 智飞生物 | -0.126975753 | 0 |
| 水痘疫苗 | -0.104246818 | 0 |
| 沃森生物 | -0.086112677 | 0 |
| 沃森生物股吧 | -0.076712124 | 0 |
| 沃森生物股票 | -0.071589459 | 0 |
| 狂犬疫苗 | -0.063824335 | 0 |
| 疫苗 | -0.061443756 | 0 |
| 疫苗事件 | -0.058253388 | 0 |
| 脊灰疫苗 | -0.055009736 | 0 |
| 问题疫苗 | -0.053561447 | 0 |
| 非法疫苗 | -0.042854103 | 4 |

2017 版医保目录发布：

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 人福医药 | 0.430645326 | 3 |
| 人福医药股票 | -0.375649858 | 0 |
| 亿帆医药 | 0.337617753 | 8 |
| 众生药业 | 0.293171479 | 8 |
| 医保目录 | 0.263564695 | 8 |

| | | |
|---|---|---|
| 医保目录查询 | 0.256263946 | 7 |
| 医药 | -0.245338167 | 0 |
| 医药板块 | 0.186711434 | 8 |
| 医药股 | 0.162635622 | 5 |
| 医药股票有哪些 | 0.156797442 | 6 |
| 康弘药业 | 0.140594065 | 6 |
| 康缘药业 | 0.138741916 | 7 |
| 康缘药业股票 | -0.136070987 | 0 |
| 易明医药 | -0.135713062 | 3 |
| 海正药业 | 0.124690276 | 6 |
| 益佰制药 | -0.115406718 | 0 |
| 益佰制药股票 | -0.102623052 | 0 |
| 葵花药业 | -0.089971785 | 0 |

长春长生问题疫苗：

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 2680 | -0.394502435 | 3 |
| 狂犬疫苗造假 | -0.29206356 | 3 |
| 长春长生 | -0.280957165 | 4 |
| 长春长生生物 | -0.273041665 | 3 |
| 长春长生疫苗事件 | -0.24365927 | 3 |
| 长春长生股票 | -0.237890742 | 3 |
| 长生生物官网 | -0.224453441 | 3 |
| 长生生物疫苗 | -0.206581197 | 3 |
| 长生生物疫苗事件 | -0.197267602 | 2 |
| 长生生物股吧 | -0.191589646 | 3 |
| 长生生物股票 | -0.185732465 | 3 |

CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》：

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 300347 | 0.602511442 | 2 |
| cde | -0.29358537 | 2 |
| cro | -0.275586211 | 5 |
| cxo | 0.270981846 | 8 |
| etf | 0.226757563 | 8 |
| 医药 | 0.195705036 | 2 |
| 医药基金 | -0.193606353 | 8 |
| 医药板块 | -0.185083959 | 2 |
| 医药股 | 0.174770897 | 3 |
| 国家药监局 | 0.164798709 | 0 |
| 复星医药 | -0.159587369 | 7 |
| 复星医药股吧 | -0.098637595 | 7 |

| | | |
|---|---|---|
| 泰格医药 | -0.096945007 | 6 |

药明康德闪崩:

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 603259 | -0.634677545 | 4 |
| cro | -0.583582409 | 7 |
| cxo | -0.475014286 | 8 |
| cxo 医药是什么意思 | -0.43268216 | 7 |
| 中欧医疗健康混合 c | -0.419100703 | 7 |
| 医药 | -0.378798319 | 7 |
| 医药基金 | -0.374199774 | 7 |
| 医药板块 | -0.365977547 | 7 |
| 医药股 | -0.3497229 | 7 |
| 康龙化成 | -0.339913257 | 7 |
| 泰格医药 | -0.322698031 | 7 |
| 百济神州 | -0.310522952 | 7 |
| 美国制裁 | -0.306732839 | 7 |
| 美国实体清单 | -0.303966296 | 7 |
| 药明康德 股票 | -0.281833263 | 7 |
| 药明康德 | -0.264454992 | 7 |
| 药明康德股吧 | -0.237773588 | 7 |
| 药明康德跌停 | 0.209746523 | 0 |
| 药明生物 | -0.162906108 | 7 |

最终选取 Pearson 相关系数绝对值在 0.4 以上的关键词，选取的关键词如下表:

1、山东非法疫苗事件:

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 300142 | 0.629849371 | 1 |
| 二类疫苗 | 0.576620449 | 6 |
| 冷链物流 | 0.573985531 | 1 |
| 山东疫苗事件 | 0.539923246 | 1 |

2、2017 版医保目录发布:

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 人福医药 | 0.430645326 | 3 |

3、长春长生问题疫苗:

| 关键词 | Corrcoef | Lag |
|---|---|---|

| | | |
|---|---|---|
| 2680 | -0.394502435 | 3 |

4、CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》:

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 300347 | 0.602511442 | 2 |

药明康德闪崩:

| 关键词 | Corrcoef | Lag |
|---|---|---|
| 603259 | -0.634677545 | 4 |
| cro | -0.583582409 | 7 |
| cxo | -0.475014286 | 8 |
| cxo 医药是什么意思 | -0.43268216 | 7 |
| 中欧医疗健康混合 c | -0.419100703 | 7 |

### 4.2.4 调整关键词的时差

各关键词的领先期和权重，是在构建搜索关键词合成指标时需要考虑的重点。通过前面的计算得到各个搜索关键词的领先(滞后或一致)期和对应相关系数，并以其为合成指标的权数，由于每个关键词的领先期的阶数各有不同，还需要调整关键词的时差使其与目标变量达到一致，并取其自然对数形式$\ln(concern_t)$，避免异常值影响，使数据更加平滑。

```
import os
import pandas as pd
import correalation
import math
import copy
#遍历得到每个关键词与对应股价序列的最大相关系数
def keywordCorr():
    for i in os.listdir():
        if os.path.isdir(i) and "关键词" in os.listdir(i):
            corrData = []
            keywordName = []
            for j in os.listdir(i):
                if os.path.isfile(os.path.join(i, j)):
                    dfp = correalation.lnPrice(os.path.join(i, j))
                    filename = j.split("_")[1].split(".")[0] +"Corr.csv"
                    path = os.path.join(i, "关键词")
                    for k in os.listdir(path):
                        df = correalation.searchCnt(dfp, os.path.join(path, k))
                        corrData.append(correalation.correlation(df, 8))
                        keywordName.append(k.split("_")[0])
                    corrData.sort(key=lambda x:math.fabs(x[0]), reverse=True)
                    dfc = pd.DataFrame(corrData, columns=["Corrcoef", "Lag"], index=keywordName)
                    dfc.to_csv(os.path.join("corr", filename))
class weightValue:
    def __init__(self):
        self.keyword = []
        self.corr = []
        self.weight = []
    def addKeyword(self, e):
        self.keyword.append(e)
    def addCorr(self, e):
        self.corr.append(math.fabs(e))
    def isInKeyword(self, e):
        return e in self.keyword
    def setWeight(self):
        s = sum(self.corr)
        l = len(self.corr)
        self.weight = [None] * l
        for i in range(len(self.corr)):
            self.weight[i] = self.corr[i]/s
    def getWeight(self,e):
        assert e in self.keyword
        for i in range(len(self.keyword)):
            if e == self.keyword[i]:
                return self.weight[i]
```

```
#以相关系数绝对值为权重合成搜索量指数
def searchIndex():
    for i in os.listdir("corr"):
        name = i.split("C")[0]
        value = weightValue()
        df = pd.read_csv(os.path.join("corr", i), names=["Name", "Corrcoef", "Lag"], header=0)
        for j in range(len(df)):
            if math.fabs(df["Corrcoef"][j]) > 0.39:
                value.addCorr(df["Corrcoef"][j])
                value.addKeyword(df["Name"][j])
        value.setWeight()
        searchData = []
        for j in os.listdir():
            if os.path.isdir(j) and "price_"+name+".csv" in os.listdir(j):
                dfp = correalation.lnPrice(os.path.join(j, "price_"+name+".csv"))
                df = copy.deepcopy(dfp)
                for k in os.listdir(os.path.join(j, "关键词")):
                    if value.isInKeyword(k.split("_")[0]):
                        weight = value.getWeight(k.split("_")[0])
                        dfs = correalation.searchCnt(dfp, os.path.join(j, "关键词", k))
                        searchList = [item * weight for item in dfs["searchCnt"]]
                        searchData.append(searchList)
                search = []
                for l in range(len(searchData[0])):
                    s = 0
                    for t in range(len(searchData)):
                        s += searchData[t][l]
                    search.append(math.log(s))
                df["searchIndex"] = search
                df.to_csv(os.path.join("learningData", name+".csv"), index=False)
def priceShift():
    for i in os.listdir("learningData"):
        if "Shift" not in i:
            name = i.split(".")[0]
            df = copy.deepcopy(pd.read_csv(os.path.join("learningData", i)))
            df["shiftPrice"] = df["lnPrice"].shift(-1)
            df = df.drop(labels=len(df)-1, axis=0)
            df = df.drop(columns=["Price"])
            df.to_csv(os.path.join("learningData", name+"Shift.csv"), index=False)
```

### 4.2.5 格兰杰因果关系检验

股价预测模型在将网络搜索指标作为解释变量引入之前，我们需要检验其因果关系，即通过格兰杰因果关系检验网络搜索关键词指数与综合股价指数，来明确构建的指标是否可以作为影响医药行业股价的变量之一。检验代码与结果如下所示，即网络搜索关键词指数通过检验：

```
from statsmodels.tsa.stattools import coint
import pandas as pd
fname='ybShift.csv'
df = pd.read_csv(fname, names=['Date','InPrice','searchIndex','shiftPrice'], header=0)
print(coint(df['searchIndex'],df['InPrice'],maxlag=2))
#print(coint(df['InPrice'],df['searchIndex'],maxlag=2))
```

cde,cs,yb,ym,ymkd 检验结果如下：

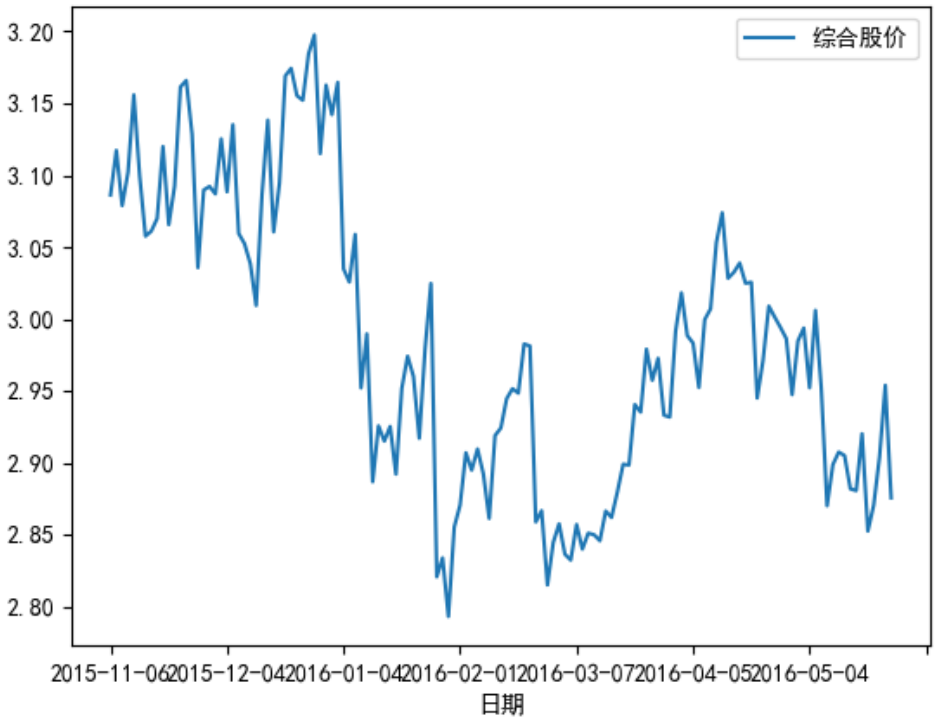
```
(-3.664622379413805, 0.02033432337698851, array([-3.98068047, -3.38245632, -3.07649249]))
(-9.987896298526149, 2.6621694888364203e-16, array([-3.98068047, -3.38245632, -3.07649249]))
(-4.561974065196871, 0.0009668954016023522, array([-3.98068047, -3.38245632, -3.07649249]))
(-4.0288687510017045, 0.00652910656909l701, array([-3.98068047, -3.38245632, -3.07649249]))
(-5.9594377121849, 1.9849020844451363e-06, array([-3.98068047, -3.38245632, -3.07649249]))
```

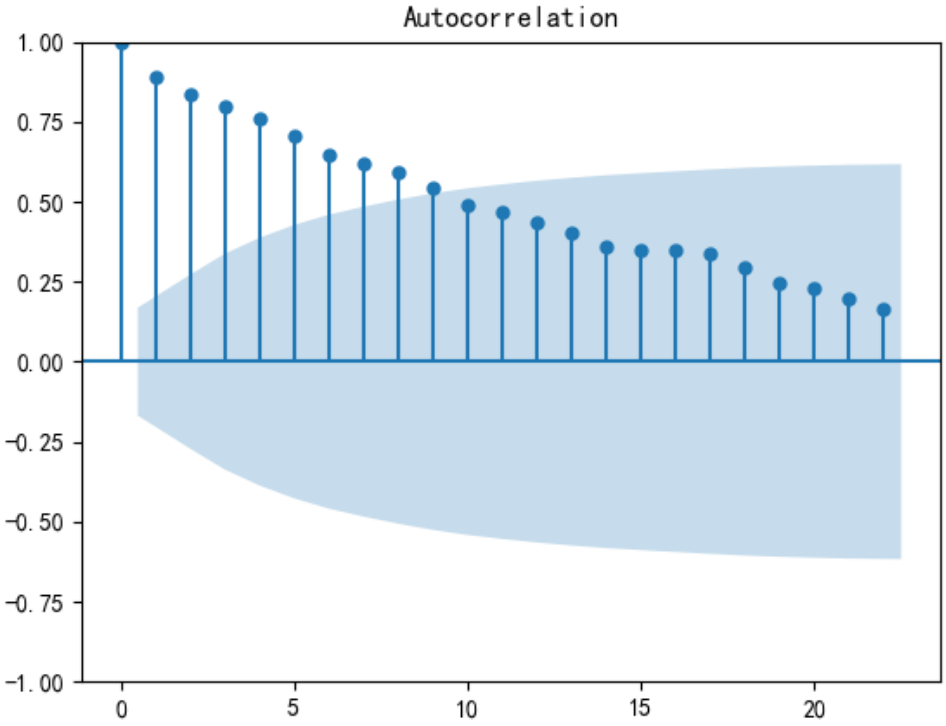
## 4.3 综合股价指数的时间序列分析

### 4.3.1 时间序列建模分析

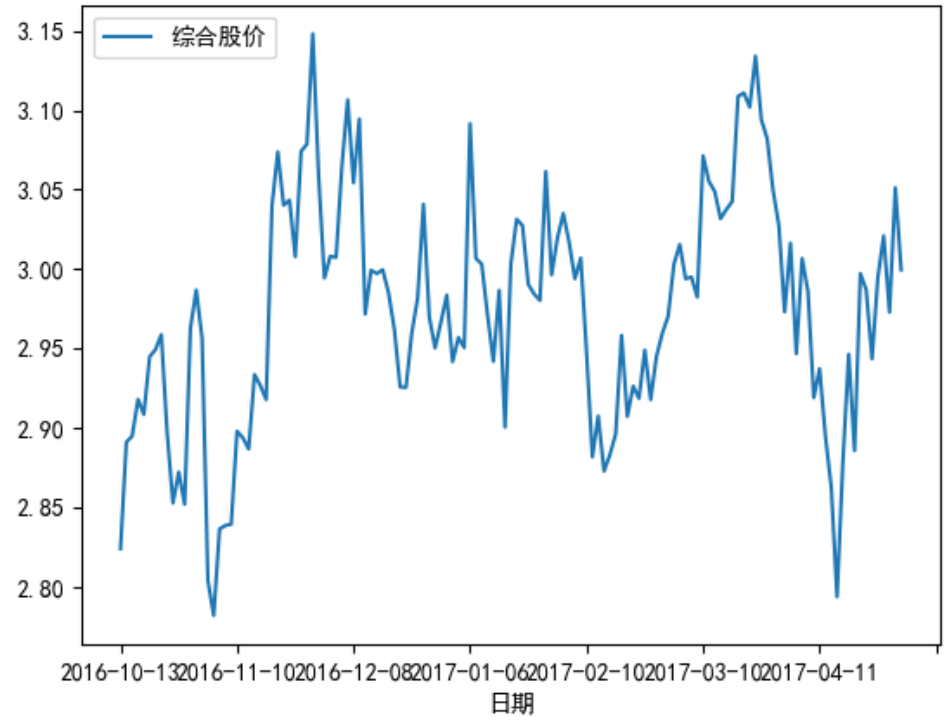
对综合股价指数进行时间序列建模分析，下图为综合股价数据的时序图和自相关图：

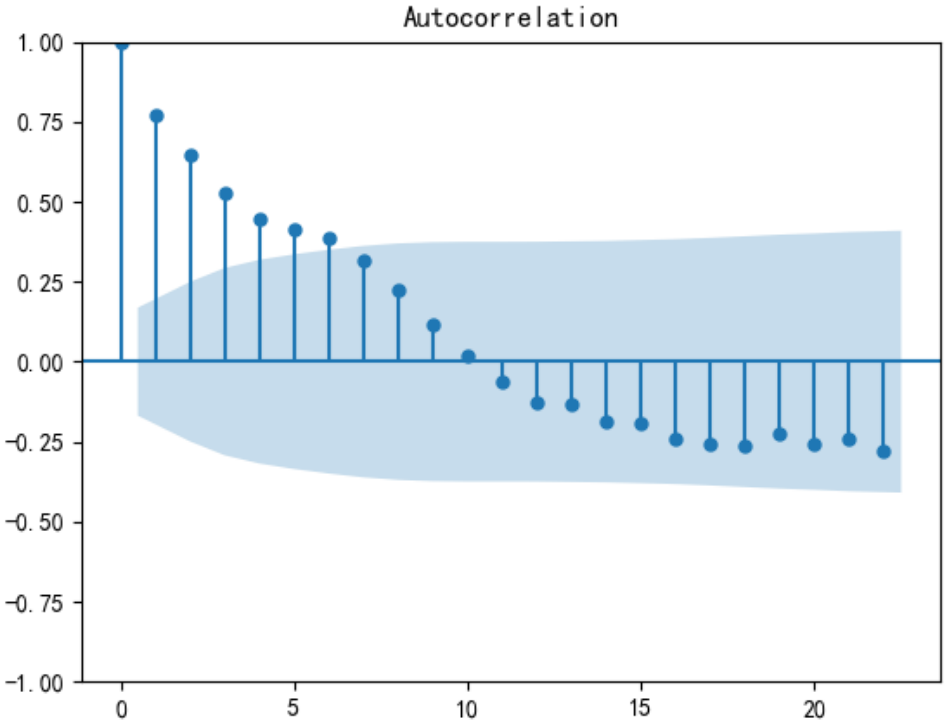
1、山东非法疫苗事件:



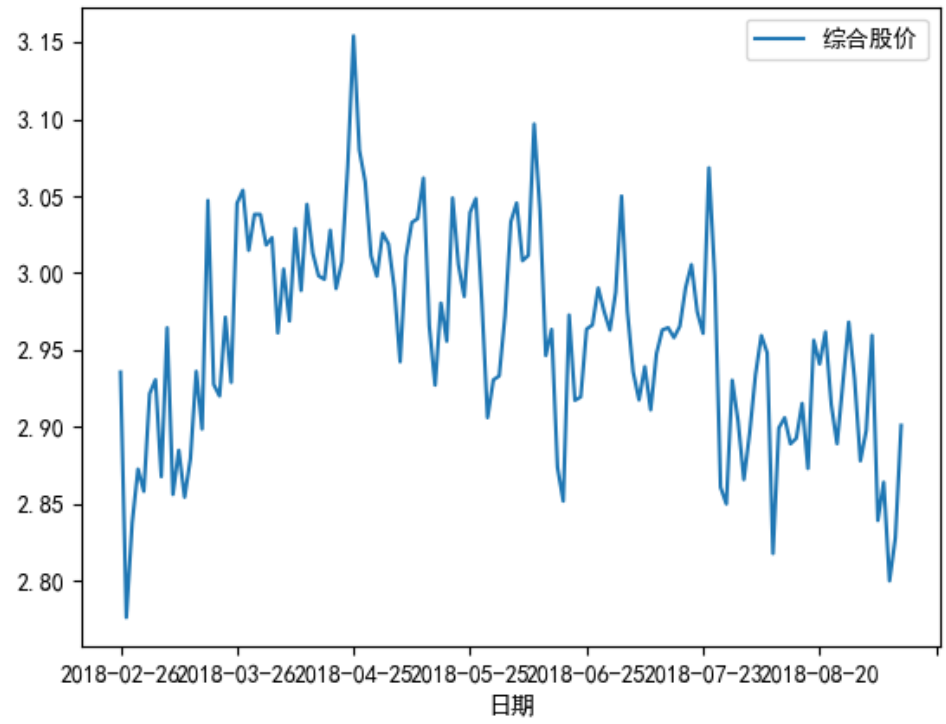


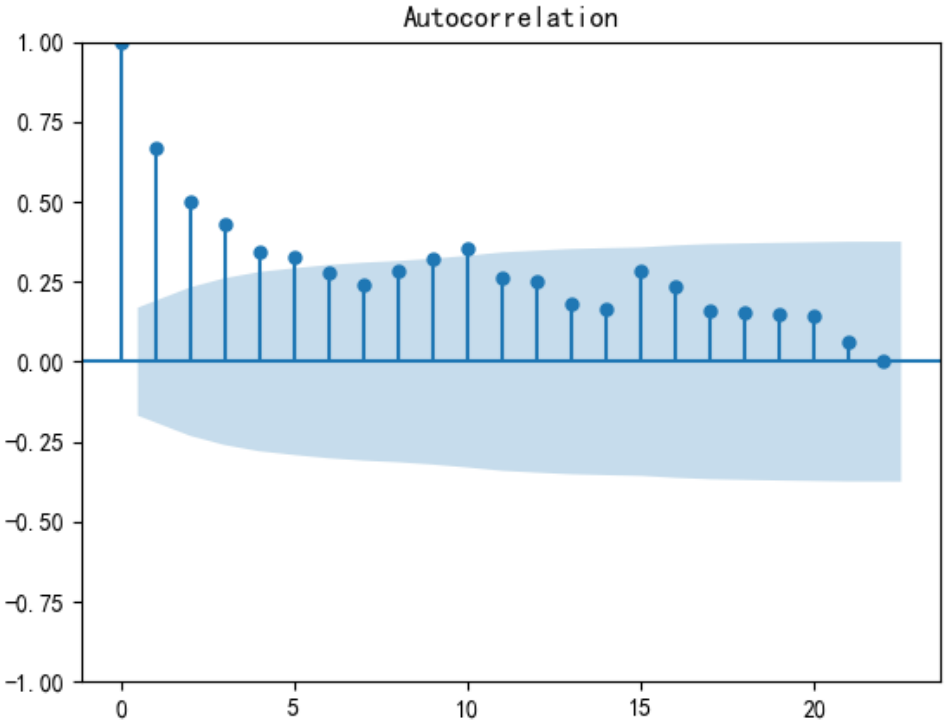

2、2017 版医保目录发布:



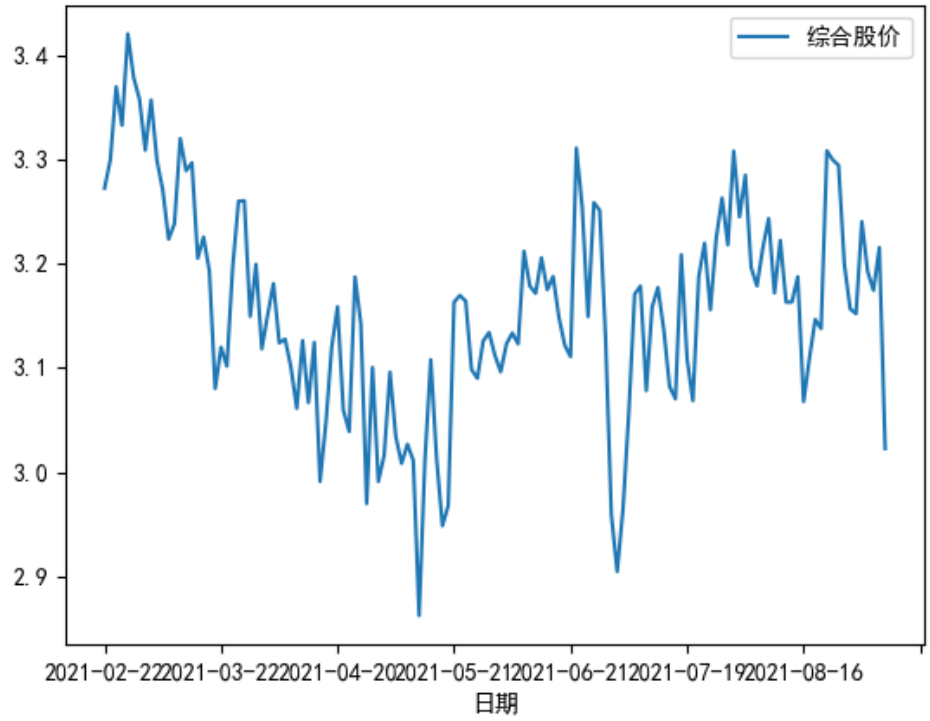

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

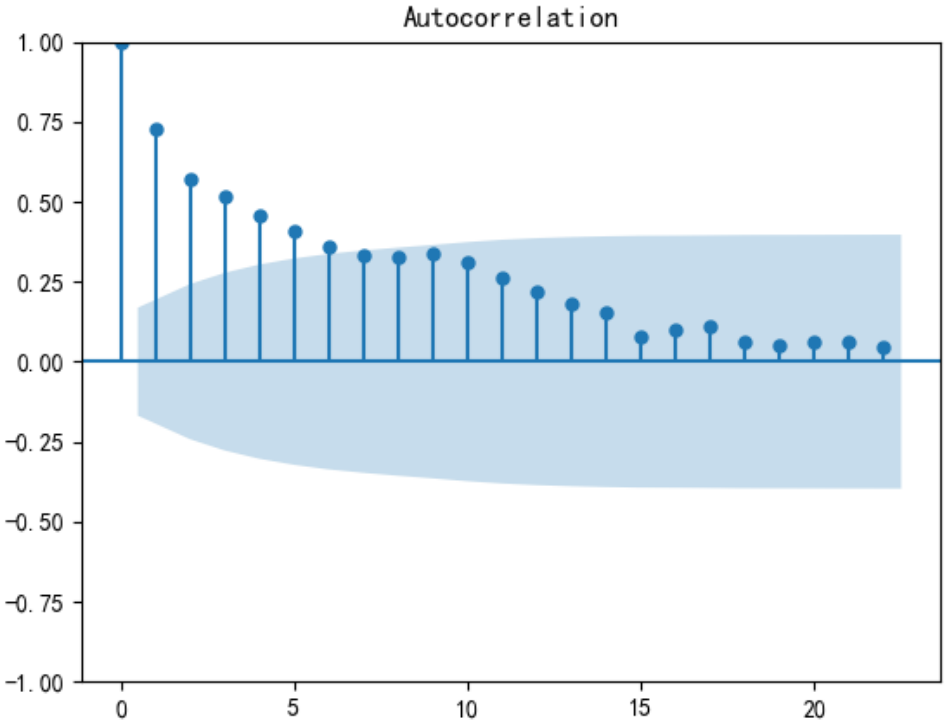
3、长春长生问题疫苗:


综合股价
3.15
3.10
3.05
3.00
2.95
2.90
2.85
2.80
2018-02-26
2018-03-26
2018-04-25
2018-05-25
2018-06-25
2018-07-23
2018-08-20
日期

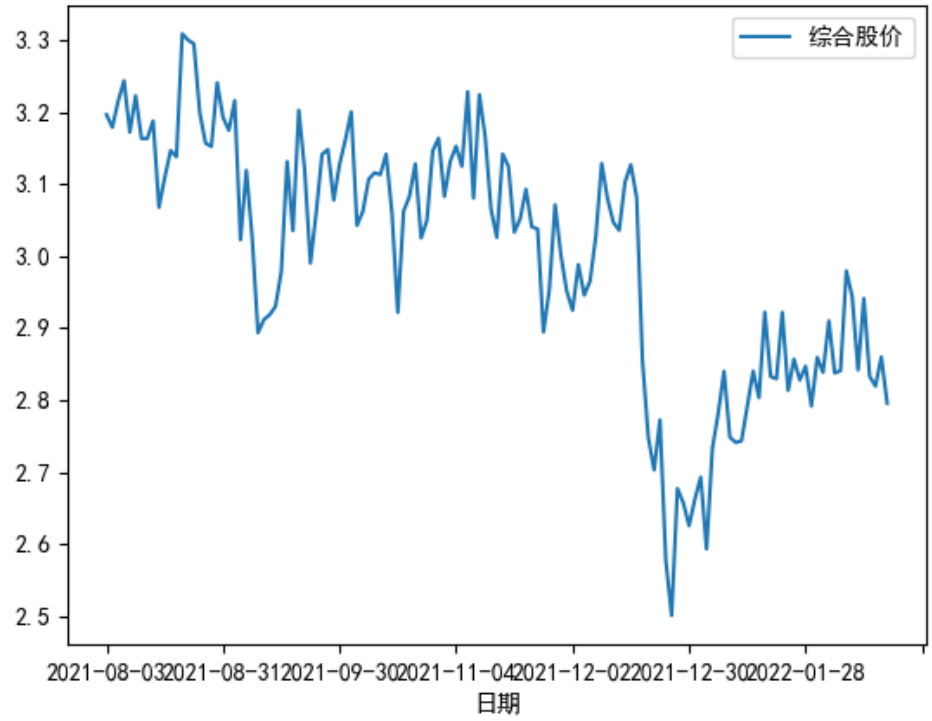


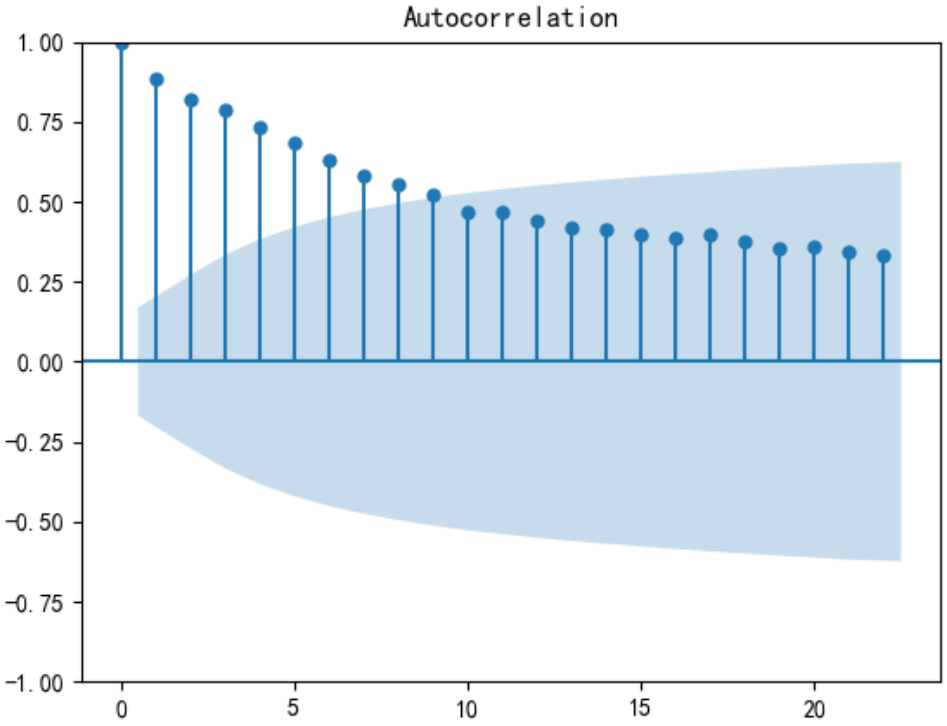

4、CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》:




Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

5、药明康德闪崩:


综合股价
3.3
3.2
3.1
3.0
2.9
2.8
2.7
2.6
2.5
2021-08-03
2021-08-31
2021-09-30
2021-11-04
2021-12-02
2021-12-30
2022-01-28
日期




### 4.3.2 一阶差分

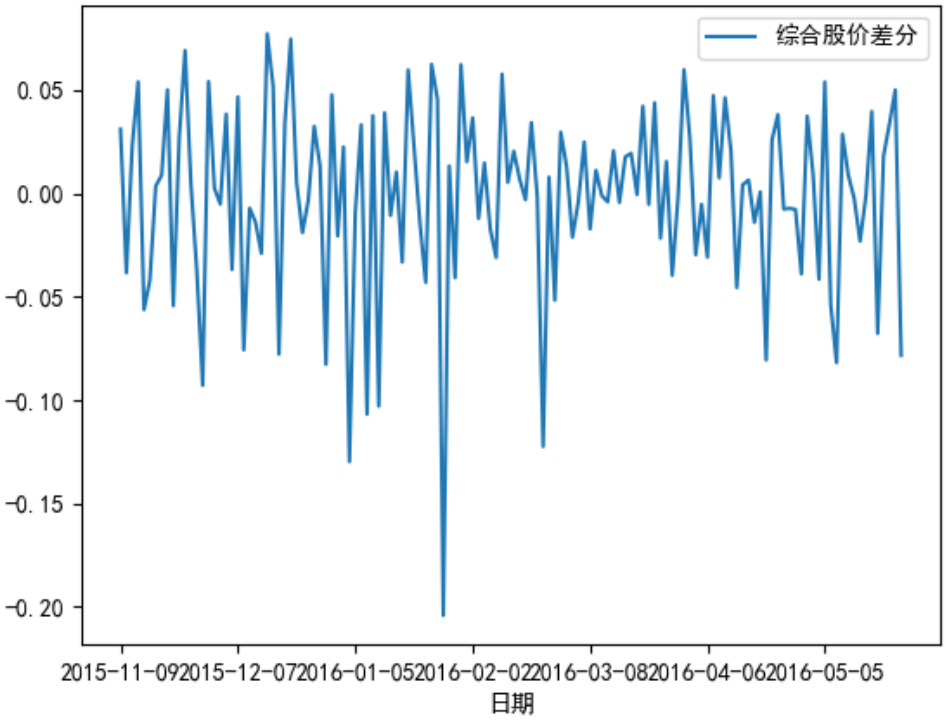
通过对上图的观察，初步判断该序列不是平稳时间序列，故对其进行一阶差分，计算公式如下：

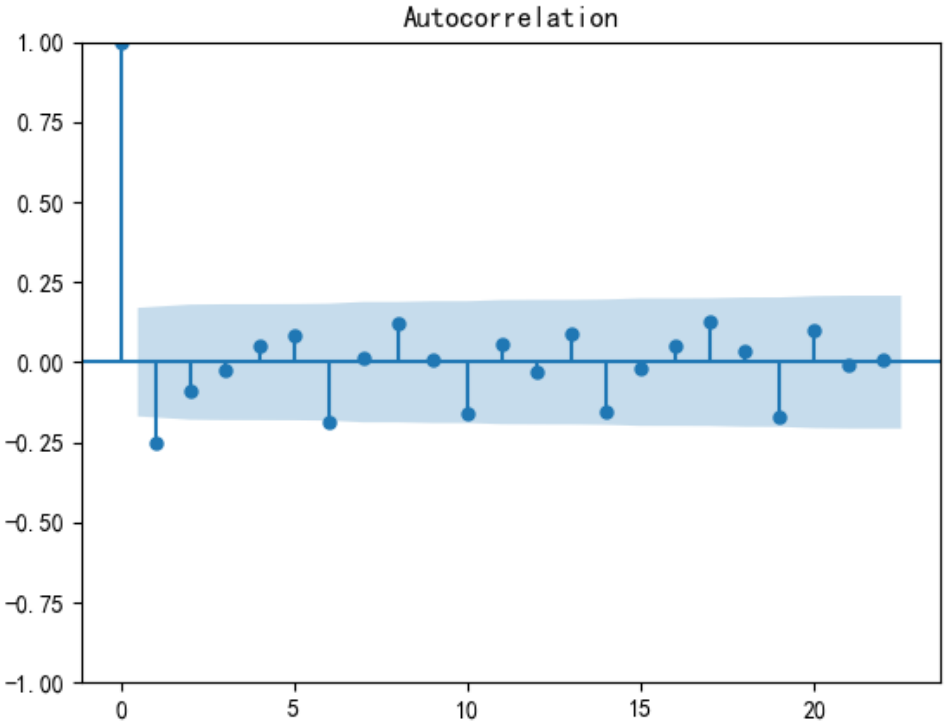
$$\Delta y_x = y(x+1) - y(x) \qquad (x = 0, 1, 2, \ldots)$$

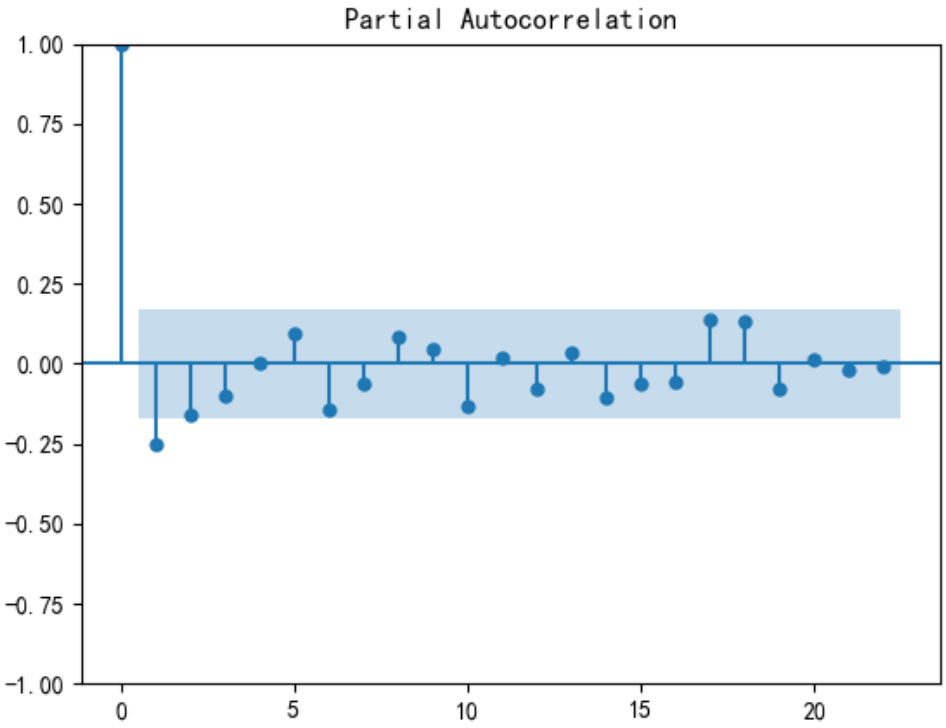
$$\Delta y_x = y_{x+1} - y_x \qquad (x = 0, 1, 2, \ldots).$$

一阶差分后股票的时序图、自相关函数图、偏相关函数图如下：

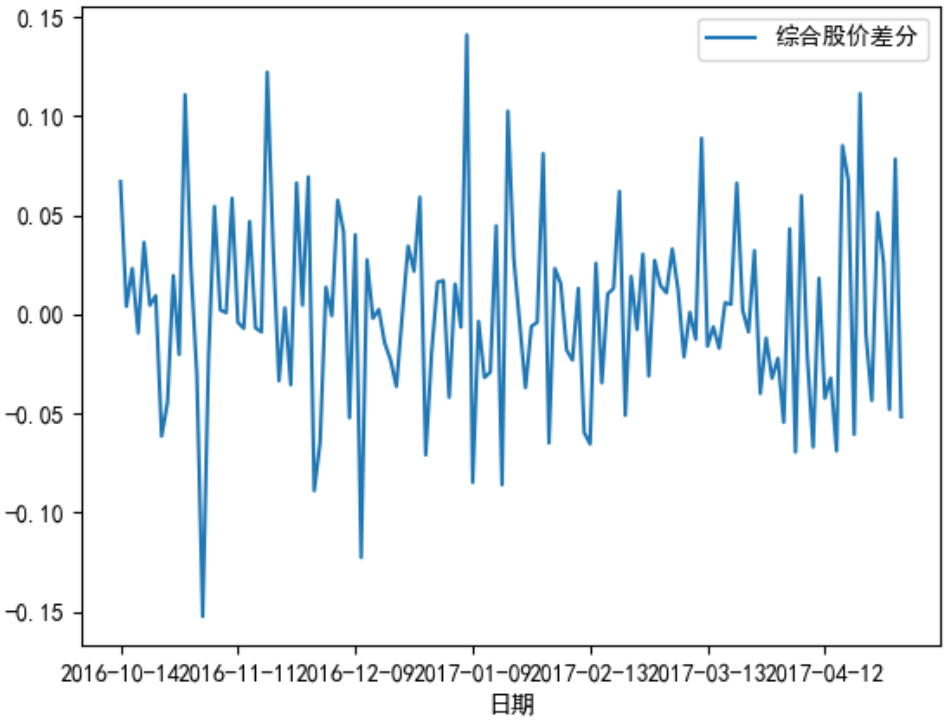
1、山东非法疫苗事件:

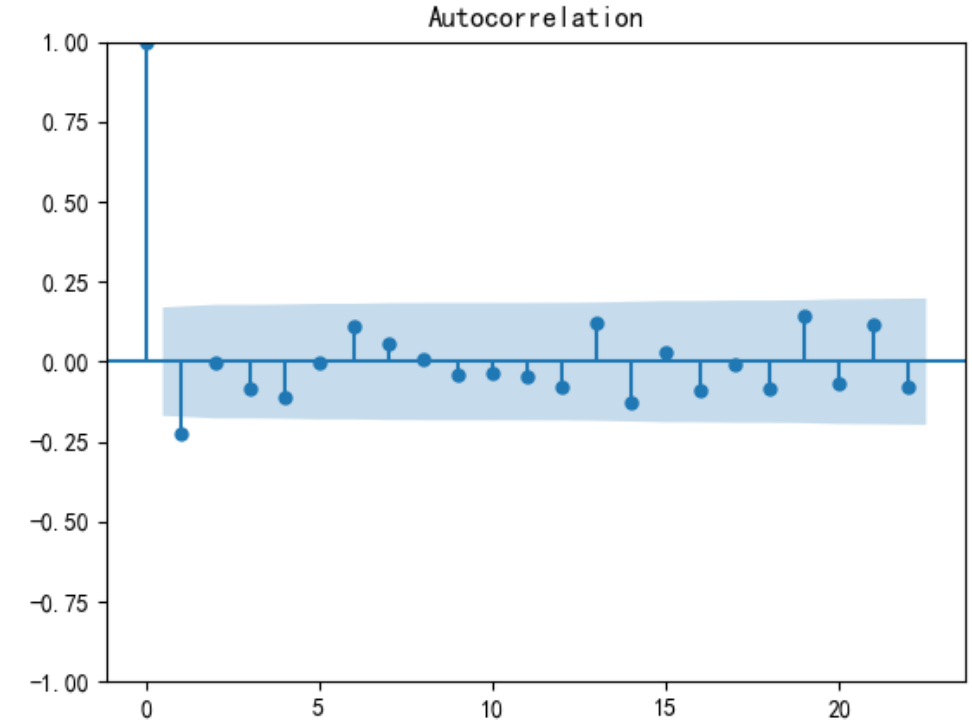

综合股价差分
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2015-11-09
2015-12-07
2016-01-05
2016-02-02
2016-03-08
2016-04-06
2016-05-05
日期

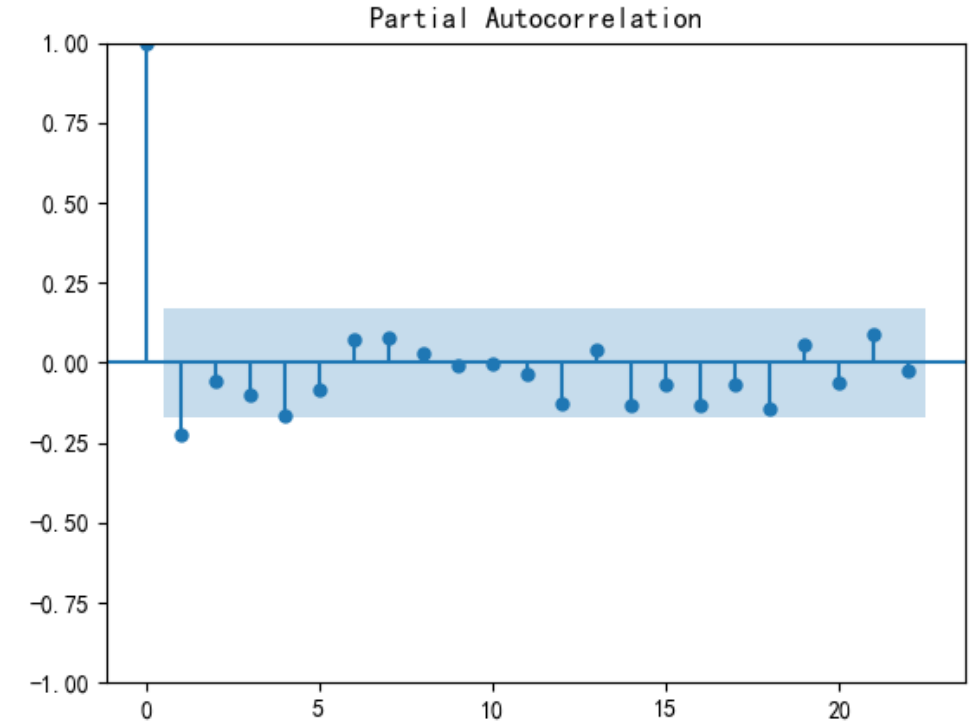

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20



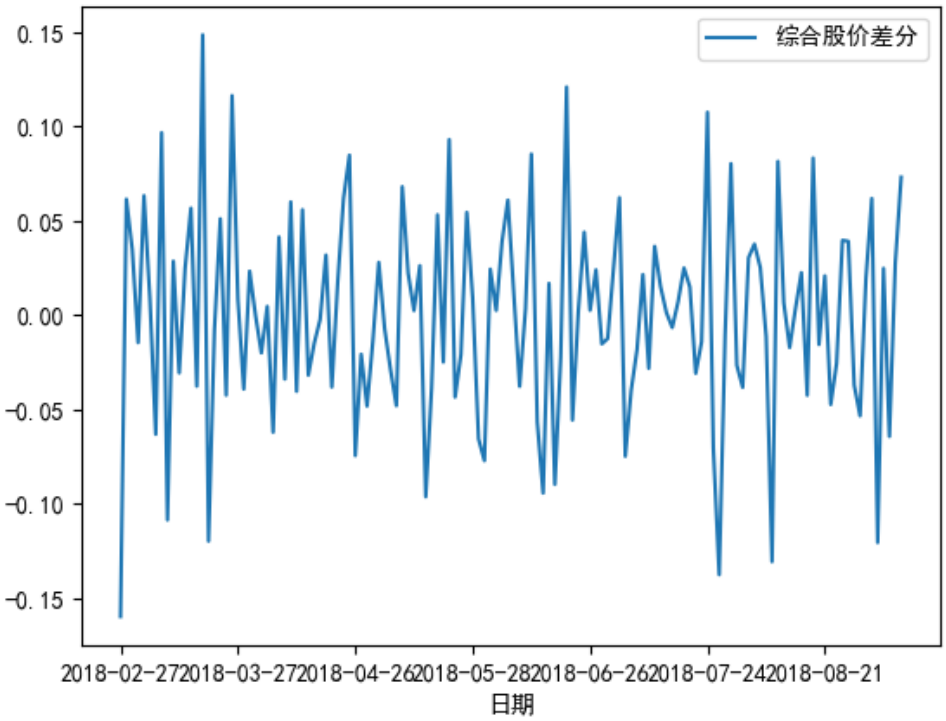

2、2017 版医保目录发布：

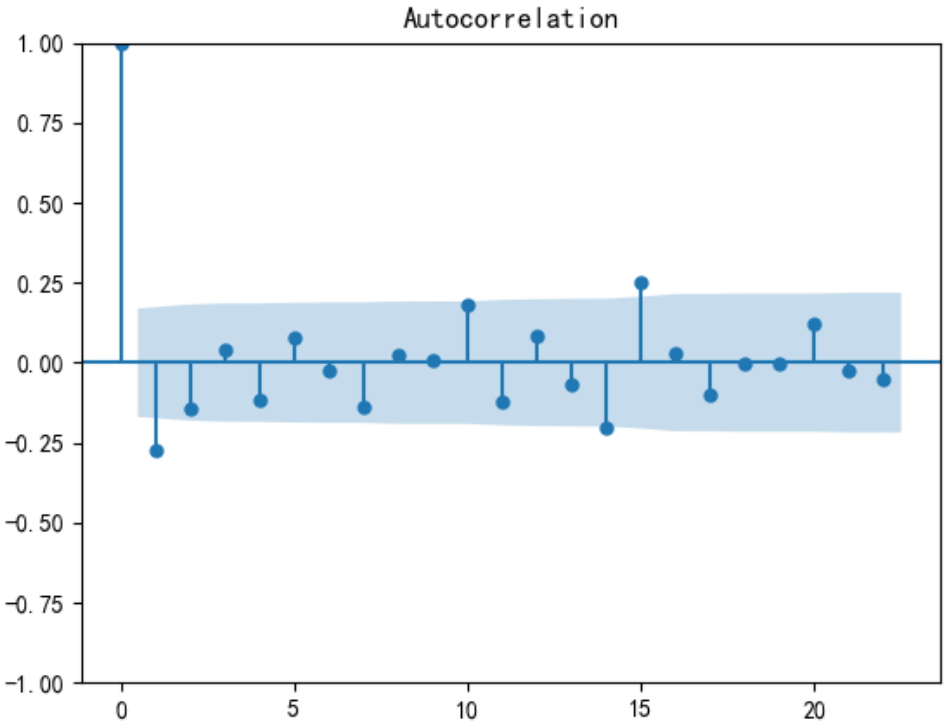


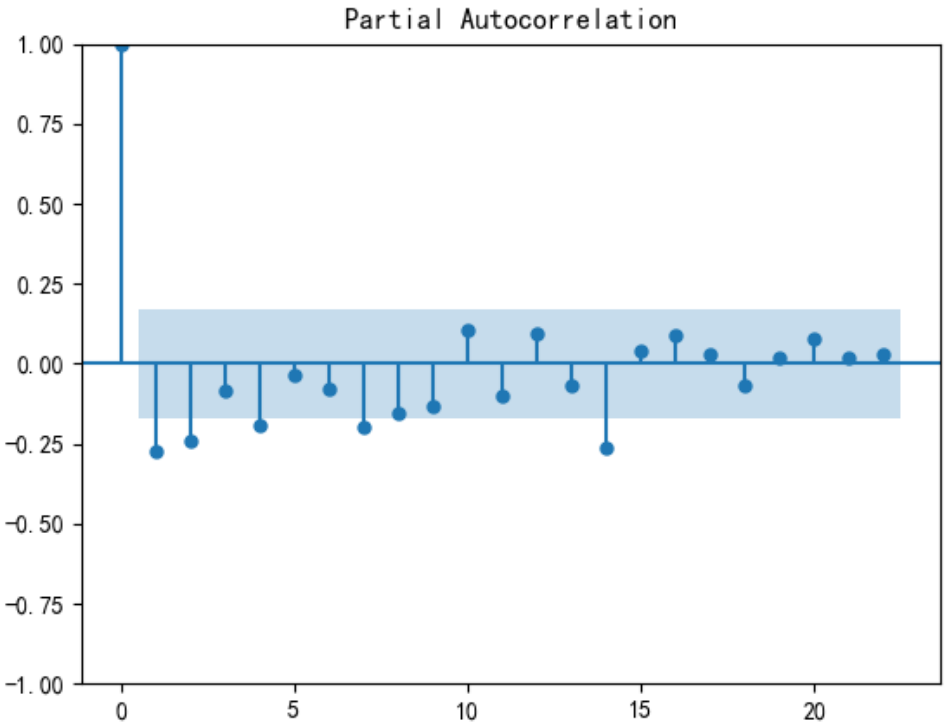

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

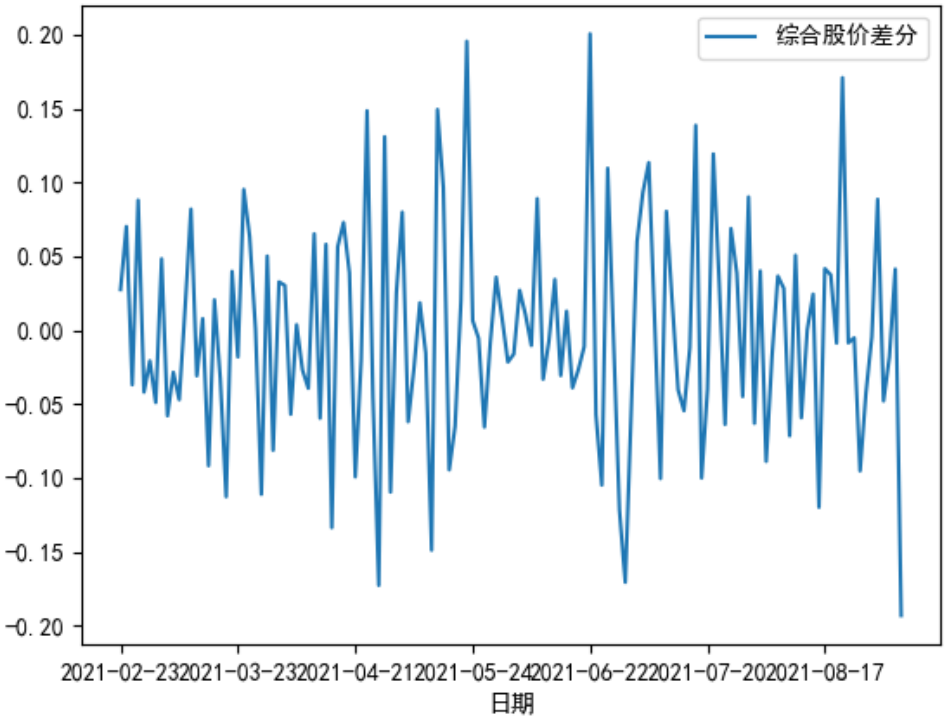
3、长春长生问题疫苗：

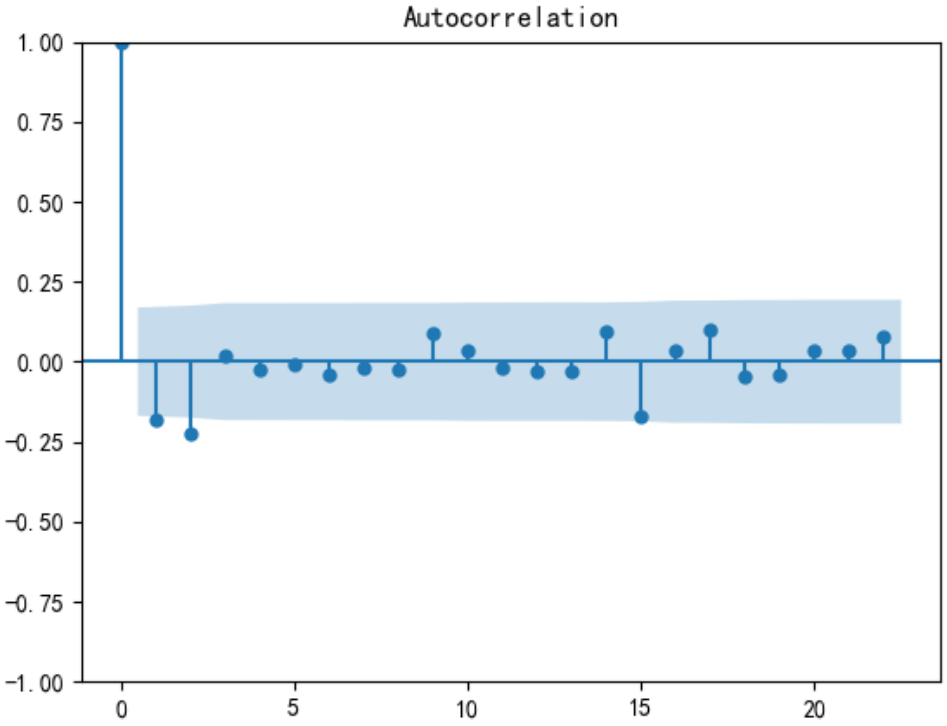

综合股价差分
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
2018-02-27
2018-03-27
2018-04-26
2018-05-28
2018-06-26
2018-07-24
2018-08-21
日期

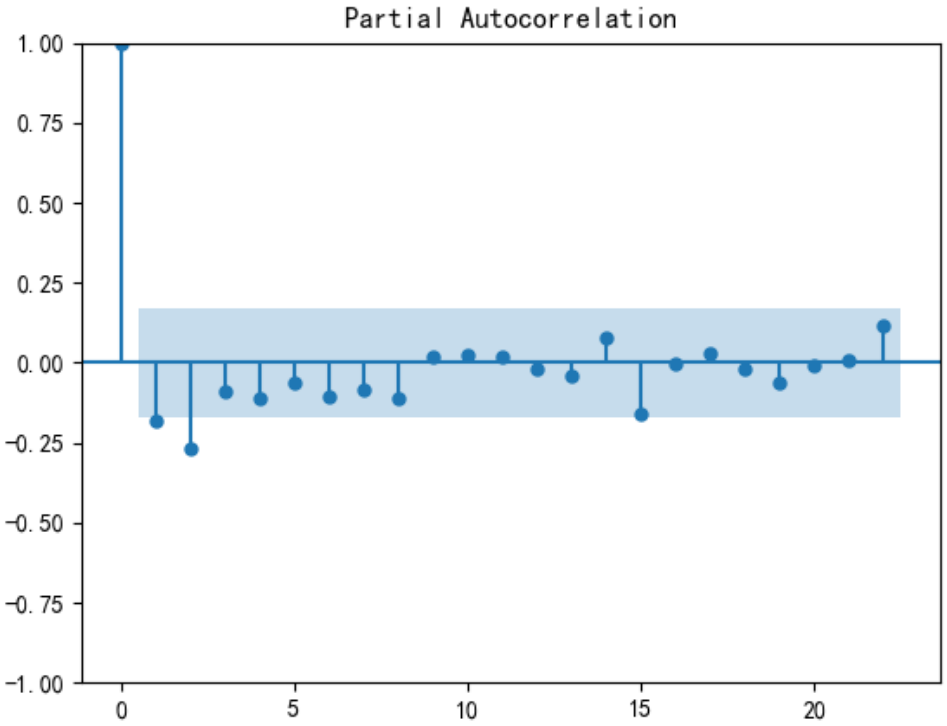

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20



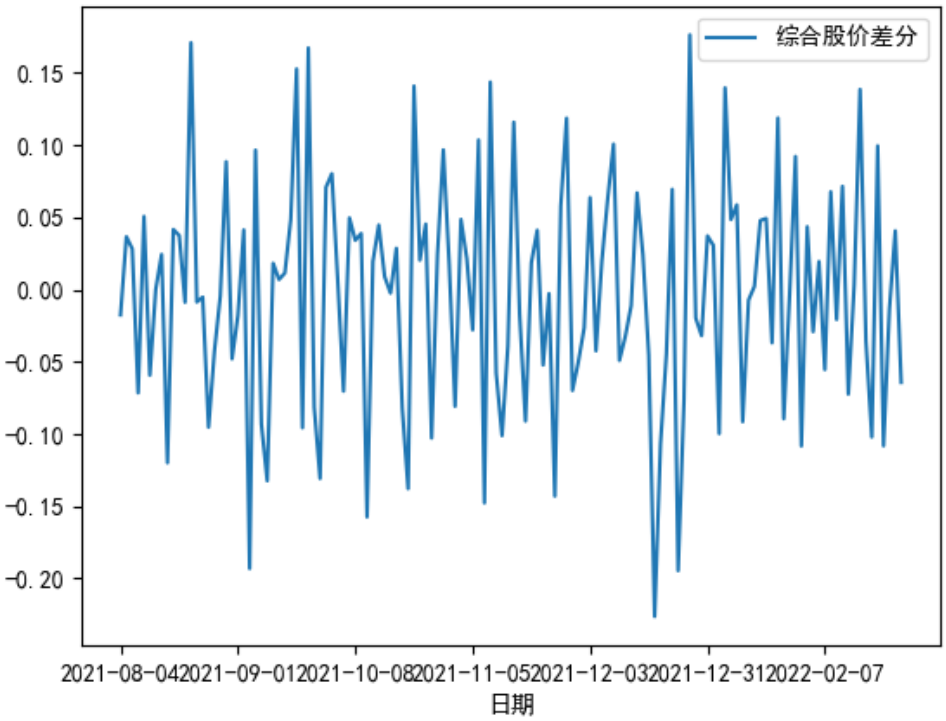

4、CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》：

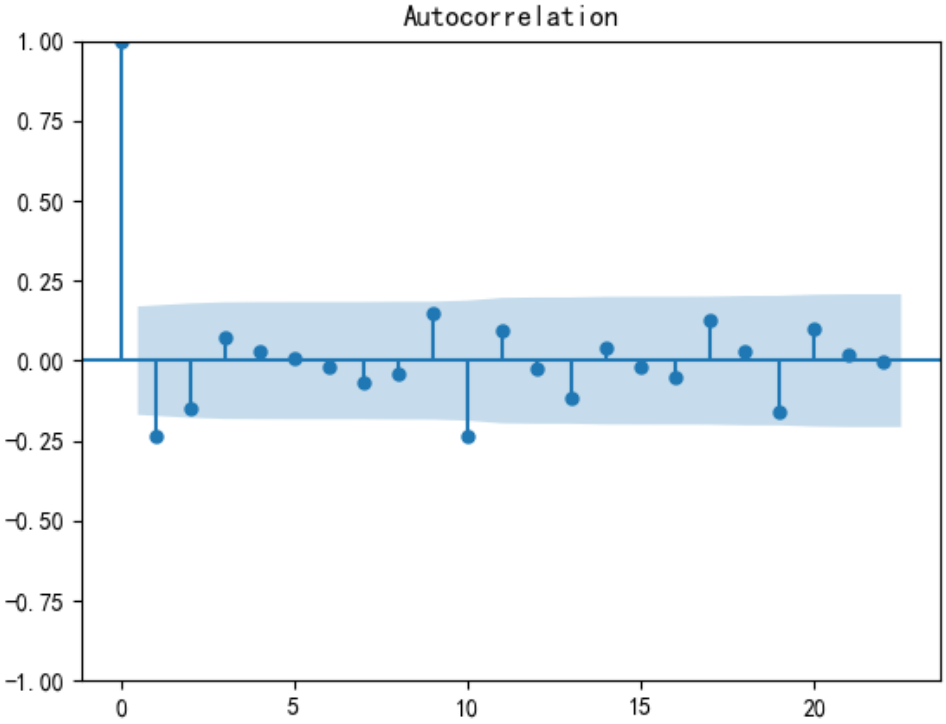


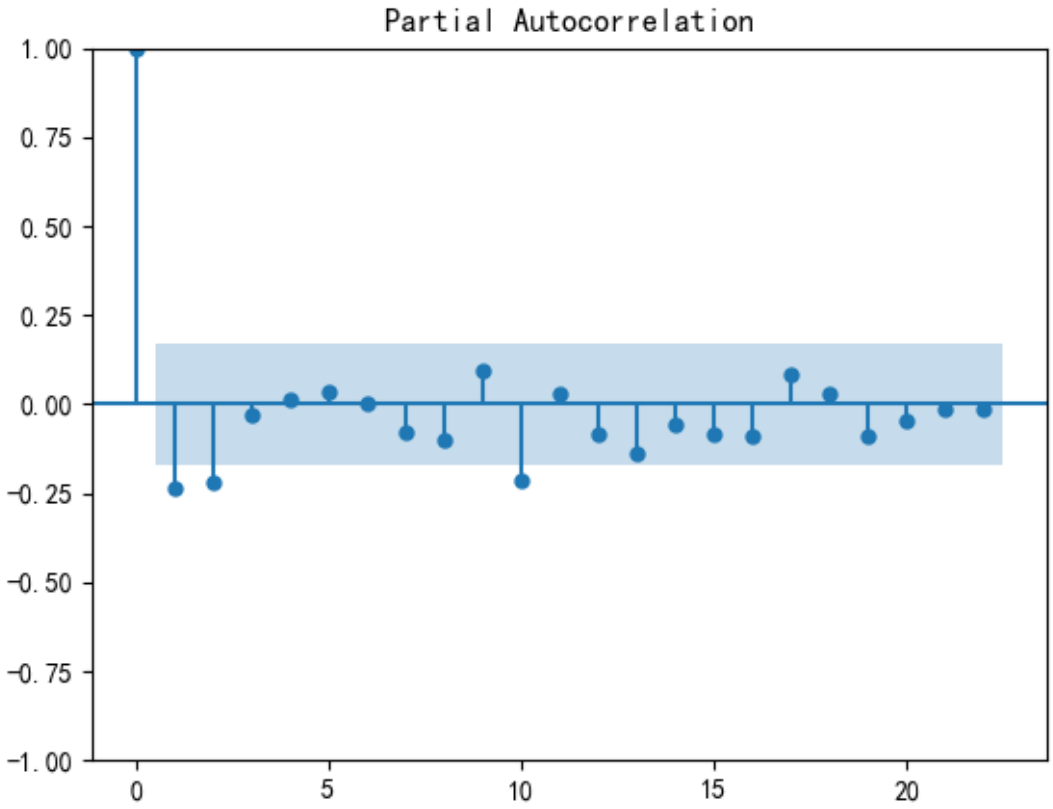

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

5、药明康德闪崩:


综合股价差分
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2021-08-04
2021-09-01
2021-10-08
2021-11-05
2021-12-03
2021-12-31
2022-02-07
日期


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20




### 4.3.3ADF 平稳性验证

经过对上图的观察，初步判断该序列为平稳时间序列，利用 ADF 平稳性进一步验证，代码和结果如下：

```
#自相关图长期大于0，作一阶差分
diffData = df.diff().dropna()
diffData.columns=["综合股价差分"]
diffData.plot()
plt.show()
plot_acf(diffData).show()
plot_pacf(diffData, method="ywm").show()
print("差分序列的ADF检验结果为：", ADF(diffData["综合股价差分"]))
```

| | ADF 值 | P 值 | Critical Value(1%) | Critical Value(5%) | Critical Value(10%) |
|---|---|---|---|---|---|
| cde | −11.5128 | 4.220225648473714e−21 | −3.4809 | −2.8837 | −2.5786 |

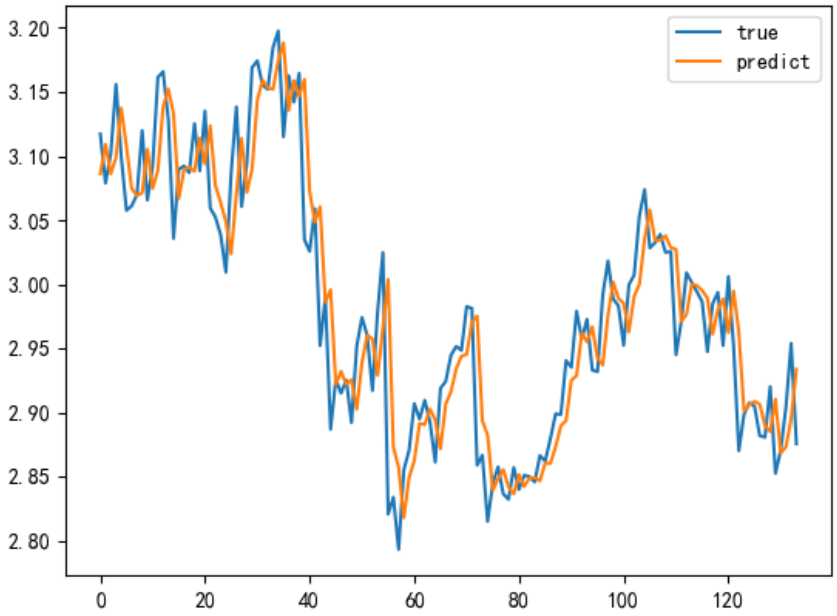
| | | | | | |
|---|---|---|---|---|---|
| cs | -4.8235 | 4.9026430670327886e-05 | -3.4861 | -2.8859 | -2.5798 |
| yb | -14.4301 | 7.655311950089794e-27 | -3.4805 | -2.8835 | -2.5785 |
| ym | -10.5235 | 9.549900612542664e-19 | -3.4809 | -2.8837 | -2.5786 |
| ymkd | -11.1692 | 2.6671770922020362e-20 | -3.4809 | -2.8837 | -2.5786 |

由上可知，该序列通过检验与判断相符，可建立 ARIMA 模型。为确定模型 p、q 阶数作出其自相关函数图、偏相关函数图，如上所示。从图可以看出，自相关系数图与偏自相关函数图均具有 1 阶截尾性，确定建立 ARIMA(1,1,1)模型。利用样本集数据拟合模型，并对最后测试样本数据进行预测，模型评估结果如下表所示，全部数据的预测图如下所示。可以发现 ARIMA(1,1,1)模型的预测能力良好，MAPE 值仅为 2.9089%，但该模型仅考虑医药行业综合股价的单个时间序列，并未将网络的搜索数据的影响考虑在内。因此下文将基于网络搜索数据研究预测股价，根据现有研究可知股价变动具有非线性特征，而自回归模型属于线性模型的一种，接下来将使用非线性算法模型进行建模拟合与预测。

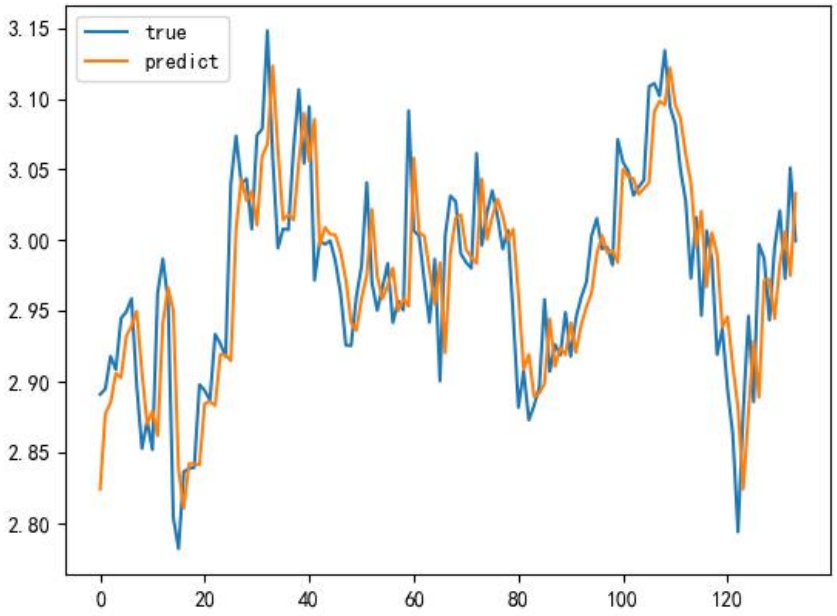
ARIMA 模型评价指标：

| | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| ARIMA(1,1,1) | 0.0857 | 0.0996 | 2.9089 |

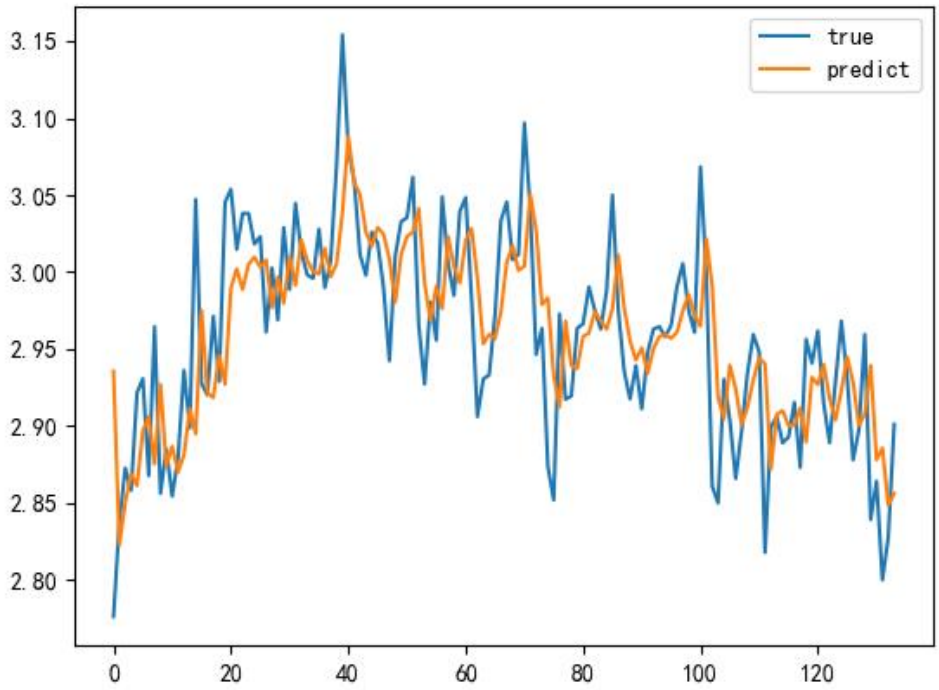
1、山东非法疫苗事件:



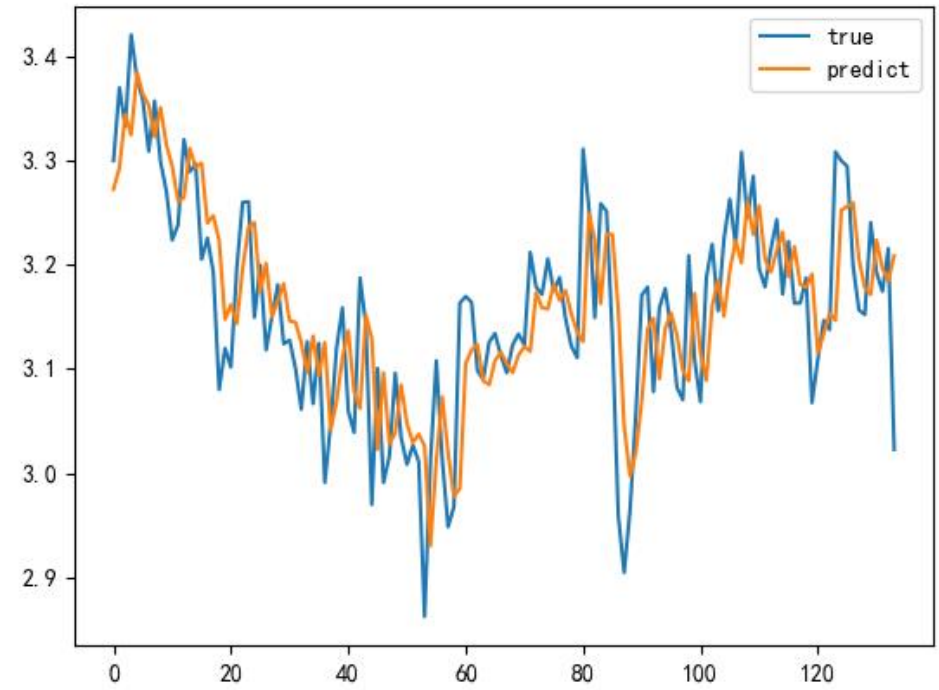

2、2017 版医保目录发布：



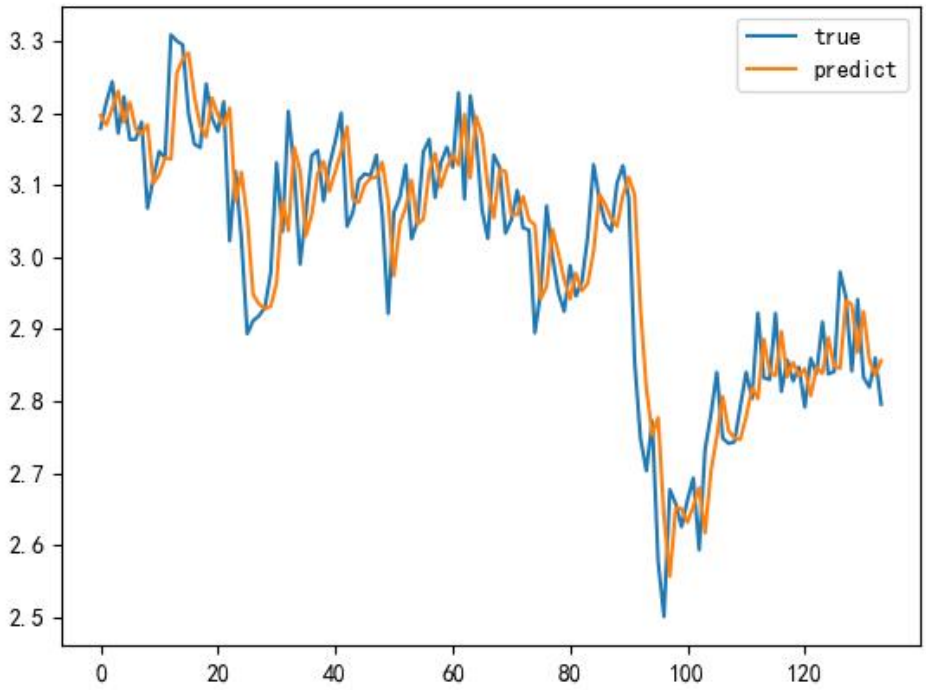

3、长春长生问题疫苗：




4、CDE 发布《以临床价值为导向的抗肿瘤药物临床研发指导原则(征求意见稿)》:




5、药明康德闪崩:




## 4.4 基于网络搜索数据的综合股价指数非线性预测研究

### 4.4.1 SVR 与 GBDT 的样本预处理

根据上节的时间序列分析，将综合股价指数的一阶滞后期序列与网络搜索关键词指数同时作为影响变量，以综合股价指数序列作为因变量，共同输入接下来进行预测研究的两类模型之中。

由于样本数据集中变量存在量纲上的差异，模型受到数量级较大变量的影响程度也会相对更大，而数量级较小的变量，模型受到其影响相对较小，在构造模型之前每个变量的权重因此而形成巨大差异，导致变量之间失衡严重，对于模型的建立造成一定的影响。对解释变量一阶滞后综合股价指数和网络搜索关键词指数进行 Max-Min 归一化预处理，能够避免数据的量纲影响，同时提高模型的预测能力。实际数据经由归一化处理后转化为[0,1]区间内的数值。对任意解释变量$C_i$(t)，将其归一化为$\tilde{C}_i$(t):

$$\tilde{C}_i(t)=\frac{C_{i,t}-C_{i,\min}}{C_{i,\max}-C_{i,\min}},\quad i=1,2,\cdots,n$$

为了使有限的股票样本历史数据中所挖掘出的有效信息最大化，同时引入交叉验证的方法，进而避免过拟合现象的发生，这是一种统计学上的实用方法，能够将样本数据集分割成较小子集。在机器学习的相关研究中，全部样木数据集通常会被划分成为两个部分，包括训练集和测试集。模型的训练与分析在训练集上进行，而利用测试集对模型的性能进行评估，并将最优模型选择出来。常用的交义验证方法主要有:留出法、K 折交叉验证和留一交叉验证等。

结合本文股票样本数据集，并综合算法的性能以及效率，对交叉验证方法的适用性进行考量后，本文选择的交叉验证方法为 K 折交叉验证，其基本思想是:将所有的样本数据集 D 随机划分为 K 个互不相交且大小相等的子集$D_1,D_2...D_K$,然后进行 K 次模型的训练和检验。在每次迭代计算的过程中，从 K 个子集中随机选取一个测试集，训练集由剩下的 K-1 个子集组成，这样交替进行 K 次且每个子集都被用作模型的检验。最后得到 K 个测试结果的平均值作为 K 折交叉验证下的模型性能评价指标。本文将各事件 135 个交易日的数据作为独立测试集，用以评价模型的预测精度，所使用的软件为 python。

### 4.4.2 SVR 预测模型构建与分析

利用 SVR 模型对股票市场这种非线性动态系统进行预测研究时，核函数的构造及相应参数的选择是影响模型性能的主要原因。支持向量回归不同于传统的线性模型，能够处理非线性问题，主要是通过核函数把数据映射到高维特征空间，然后在高维空间进行估计求解。常使用的核函数有以下几种类型:

（1）线性核函数： $K\left(x,x_i\right)=x\cdot x_i$

（2）多项式核函： $K\left(x,x_i\right)=\left(\left(x\cdot x_i\right)+1\right)^d$

（3）径向基核函数： $K\left(x,x_i\right)=\exp\left(-\frac{\left\|x-x_i\right\|^2}{2\sigma^2}\right)$

（4）Sigmoid 核函数： $K\left(x,x_i\right)=\tanh\left(\kappa\left(x,x_i\right)-\delta\right)$

目前，对于核函数及其参数的选择还没有统一的定论和标准，在各个应用领域中，不同的核与参数性能差别会很大。支持向量回归模型的预测精度很大程度

上受到核函数及其参数选择的影响，参数变动会隐式地改变核函数的映射关系，使映射特征空间的复杂程度发生变化，从而影响到 SVR 模型的预测精度。所以在核函数方面本文将分别对基于线性核函数、径向基核函数以及 sigmoid 核函数的模型进行训练，进行比较分析。同时在模型参数选择中，使用网格搜索法进行优化。网格搜索算法是一种大范围点集搜索方法，即将参数空间用等步长的线分割成网格，遍历各个参数网格点，然后通过 10 折交叉验证的方式，找出使交叉验证精准度最高的参数组合作为模型的最优参数。

本文使用 Python 软件中的 sklearn.svm 模块建立基于支持向量回归(SVR)的股价预测模型。SVR 模型的性能很大程度上受到惩罚参数 C、核函数类型及相应核参数的选择的影响:基于线性核函数的模型，主要参数为惩罚参数 C;基于径向基核函数的模型，主要参数为惩罚参数 C、核参数 gamma;基于 sigmoid 核函数的模型，主要参数为惩罚参数 C、核参数 gamma 和 coef0。同时，为探究突发事件环境下，网络搜索数据指标加入是否能够改善模型的预测性能，将仅以综合股价指数的一阶滞后期序列为输入变量的数据集训练得到基准模型。根据上述方法，在训练集上进行训练以得到各预测模型的最优参数集的设置，如下表：

各预测模型的最优参数集：

| 核函数 | 基准模型最优参数 | 核函数 | 网络搜索指数模型最优参数 |
|---|---|---|---|
| linear | C=32 | linear | C=32 |
| rbf | C=32, gamma=0.125 | rbf | C=32, gamma=0.125 |
| sigmoid | C=32, coef0=1, gamma=0.125 | sigmoid | C=32, coef0=0.03125, gamma=0.25 |

```python
import os
import time
import dataProcess
import numpy as np
import pandas as pd
import math
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor as GBR
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error
t1 = time.time()
if "corr" not in os.listdir():
    os.mkdir("corr")
if "learningData" not in os.listdir():
    os.mkdir("learningData")
#数据预处理
dataProcess.keywordCorr()
dataProcess.searchIndex()
dataProcess.priceShift()
x1_data = []
x2_data = []
x_data = []
y_data = []
```

```python
for i in os.listdir("learningData"):
    if "Shift.csv" in i:
        df = pd.read_csv(os.path.join("learningData", i))
        x1_data.append(df["lnPrice"].tolist())
        x2_data.append(df["searchIndex"].tolist())
        y_data.append(df["shiftPrice"].tolist())
for i in range(len(x1_data)):
    x_data.append([])
    x_data[i] = [[x1_data[i][j], x2_data[i][j]] for j in range(len(x1_data[i]))]
svr_linear= SVR(kernel="linear", C=32)
svr_rbf = SVR(kernel="rbf", C=32, gamma=0.125)
svr_sigmoid = SVR(kernel="sigmoid", C=32, gamma=0.125, coef0=0.1)
svr_sigmoid1 = SVR(kernel="sigmoid", C=32, gamma=0.25, coef0=0.03125)
GBR_ls1 = GBR(loss="squared_error", learning_rate=0.1, max_depth=3, n_estimators=20)
GBR_ls2 = GBR(loss="squared_error", learning_rate=0.45, max_depth=3, n_estimators=30)
GBR_lad1 = GBR(loss="absolute_error", learning_rate=0.05, max_depth=3, n_estimators=190)
GBR_lad2 = GBR(loss="absolute_error", learning_rate=0.4, max_depth=3, n_estimators=80)
GBR_huber1 = GBR(loss="huber", learning_rate=0.1, max_depth=3, n_estimators=20)
GBR_huber2 = GBR(loss="huber", learning_rate=0.1, max_depth=3, n_estimators=60)
Error = []
ErrorG = []
```

```
i = 0                                                                    ⚠5 ✓8
for t in range(len(y_data)):
    x1_test, x_test, y_test = np.array(x1_data[t]).reshape(-1, 1), np.array(x_data[t]), np.array(y_data[t])
    x1_train, x_train, y_train = [], [], []
    for k in range(len(y_data)):
        if k != t:
            x1_train.extend(x1_data[k])
            x_train.extend(x_data[k])
            y_train.extend(y_data[k])
    x1_train = np.array(x1_train).reshape(-1, 1)
    x_train = np.array(x_train)
    y_train = np.array(y_train)
    Error.append([])
    ErrorG.append([])
    svr_linear.fit(x1_train, y_train)
    y_linear = svr_linear.predict(x1_test)
    Error[i].append(mean_absolute_error(y_test, y_linear))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_linear)))
    Error[i].append(mean_absolute_percentage_error(y_test, y_linear))
    svr_rbf.fit(x1_train, y_train)
    y_rbf = svr_rbf.predict(x1_test)
    Error[i].append(mean_absolute_error(y_test, y_rbf))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_rbf)))
    Error[i].append(mean_absolute_percentage_error(y_test, y_rbf))
    svr_sigmoid.fit(x1_train, y_train)
    y_sigmoid = svr_sigmoid.predict(x1_test)
    Error[i].append(mean_absolute_error(y_test, y_sigmoid))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_sigmoid)))
```

```
    Error[i].append(mean_absolute_percentage_error(y_test, y_sigmoid))
    svr_linear.fit(x_train, y_train)
    y_linear = svr_linear.predict(x_test)
    Error[i].append(mean_absolute_error(y_test, y_linear))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_linear)))
    Error[i].append(mean_absolute_percentage_error(y_test, y_linear))
    svr_rbf.fit(x_train, y_train)
    y_rbf = svr_rbf.predict(x_test)
    Error[i].append(mean_absolute_error(y_test, y_rbf))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_rbf)))
    Error[i].append(mean_absolute_percentage_error(y_test, y_rbf))
    svr_sigmoid1.fit(x_train, y_train)
    y_sigmoid = svr_sigmoid1.predict(x_test)
    Error[i].append(mean_absolute_error(y_test, y_sigmoid))
    Error[i].append(math.sqrt(mean_squared_error(y_test, y_sigmoid)))
    Error[i].append(mean_absolute_percentage_error(y_test, y_sigmoid))
    GBR_ls1.fit(x1_train, y_train)
    y_ls = GBR_ls1.predict(x1_test)
    ErrorG[i].append(mean_absolute_error(y_test, y_ls))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_ls)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_ls))
    GBR_lad1.fit(x1_train, y_train)
    y_lad = GBR_lad1.predict(x1_test)
    ErrorG[i].append(mean_absolute_error(y_test, y_lad))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_lad)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_lad))
    GBR_huber1.fit(x1_train, y_train)
    y_huber = GBR_huber1.predict(x1_test)
```

```
    ErrorG[i].append(mean_absolute_error(y_test, y_huber))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_huber)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_ls))
    GBR_ls2.fit(x_train, y_train)
    y_ls = GBR_ls2.predict(x_test)
    ErrorG[i].append(mean_absolute_error(y_test, y_ls))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_ls)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_ls))
    GBR_lad2.fit(x_train, y_train)
    y_lad = GBR_lad2.predict(x_test)
    ErrorG[i].append(mean_absolute_error(y_test, y_lad))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_lad)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_lad))
    GBR_huber2.fit(x_train, y_train)
    y_huber = GBR_huber2.predict(x_test)
    ErrorG[i].append(mean_absolute_error(y_test, y_huber))
    ErrorG[i].append(math.sqrt(mean_squared_error(y_test, y_huber)))
    ErrorG[i].append(mean_absolute_percentage_error(y_test, y_ls))
    i += 1
Error1 = []
Error2 = []
for j in range(len(Error[0])):
    sum1 = 0
    sum2 = 0
    for k in range(len(Error)):
        sum1 += Error[k][j]
        sum2 += ErrorG[k][j]
    Error1.append(sum1/len(y_data))
    Error2.append(sum2/len(y_data))
```

```
data_single = [Error1[:3], Error1[3:6], Error1[6:9]]
df_single = pd.DataFrame(data_single, columns=["MAE", "RMSE", "MAPE"])
df_single.index = ["linear", "rbf", "sigmoid"]
df_single.to_csv("SVR_DoubleX.csv")
data_double = [Error1[9:12], Error1[12:15], Error1[15:]]
df_double = pd.DataFrame(data_double, columns=["MAE", "RMSE", "MAPE"])
df_double.index = ["linear", "rbf", "sigmoid"]
df_double.to_csv("SVR_SingleX.csv")
data_s = [Error2[:3], Error2[3:6], Error2[6:9]]
df_s = pd.DataFrame(data_s, columns=["MAE", "RMSE", "MAPE"], index=["ls", "lad", "huber"])
df_s.to_csv("GBR_DoubleX.csv")
data_d = [Error2[9:12], Error2[12:15], Error2[15:]]
df_d = pd.DataFrame(data_d, columns=["MAE", "RMSE", "MAPE"], index=["ls", "lad", "huber"])
df_d.to_csv("GBR_SingleX.csv")
t2 = time.time()
print(t2 - t1)
```

获得预测模型的最优参数集后，通过训练得到相应的 SVR 模型，为了度量模型在测试集上对股票预测准确度，本文选取平均绝对误差 MAE,均方根误差 RMSE,平均绝对百分比误差 MAPE 用以评价模型的预测性能，各评价指标的计算公式如下所示:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

其中$y_i$表示真实值，$\hat{y}_i$表示相应模型的预测值，n 为独立测试集样本容量，上述的评价指标通过计算预测值与洲试样本真实值之间的误差大小反映模型性能好坏，若模型对于股价预测效果越好，即模型预测精确程度越高，则这些指标的数值越小。上述模型在独立测试集上预测效果的评价指标如下表所示:

SVR 基准模型评价指标

| 基准模型 | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| SVR-linear | 0.0497 | 0.0632 | 1.6514 |
| SVR-rbf | 0.0733 | 0.0865 | 2.4364 |
| SVR-sigmoid | 0.1143 | 0.1354 | 3.7930 |

SVR 网络搜索指数模型评价指标：

| 网络搜索指数模型 | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| SVR-linear | 0.0466 | 0.0600 | 1.5450 |
| SVR-rbf | 0.0472 | 0.0606 | 1.5640 |
| SVR-sigmoid | 0.1144 | 0.1356 | 3.7994 |

由上表可以看出，基准模型中使用 linear 核函数的 SVR 模型各项评价指标数值均最小，在仅以综合股价指数的一阶滞后期序列为输入变量进行预测时效果最好。对于加入了搜索关键词指数的模型，其平均绝对误差、均方根误差和平均绝对百分比误差相对于基准模型而言都有所下降，取得了较好的预测效果，其中 linear 核函数的预测精准度最高，MAPE 值仅为 1.5454%，相对于基准模型提高了 0.1068%。说明网络搜索数据反映了投资者的关注度会影响其投资决策，并反映在相应医药行业板块股票市场的变动中，由此使得在原有训练预测模型的基础上预测效果进步提升。

### 4.4.3GBDT 预测模型构建与分析

GBDT 算法不仅能够处理离散变量，也可以处理连续变量，有着相对较高的预测准确率。同时它的损失函数较多，在处理异常值方面的鲁棒性较强，使得其对于受突发事件影响的股票市场变动预测有着更好的适用性。GBDT 算法用做回归时，需要根据样本的特征选择损失函数，可选择的损失函数有以下四种:

（1） 均方差损失函数： $L(y,f(x))=(y-f(x))^2$

（2） 绝对损失函数： $L(y,f(x))=|y-f(x)|$

（3） Huber 函数： $L(y,f(x))=\begin{cases}\frac{1}{2}(y-f(x))^2, & |y-f(x)|\le\delta\\ \delta(|y-f(x)|-\frac{\delta}{2}), & |y-f(x)|>\delta\end{cases}$

（4） 分位数函数： $L(y,f(x))=\sum_{y\ge f(x)}\theta|y-f(x)|+\sum_{y<f(x)}(1-\theta)|y-f(x)|$

本节选取其中的均方差损失函数、绝对损失函数、Huber 函数，进行训练测试模型精度，选出最适合的损失函数。基于梯度提升回归树的股价预测模型的构建过程与 SVR 模型相近，其中使用 Python sklearn 中的 GradientBoostingRegressor 函数作为 GBDT 模型的实现函数进行调用。GBDT 算法回归类参数分为 boosting 框架参数和弱学习器参数，结合本文数据特征，主要调节参数包括:学习率(learning_rate)、迭代次数(n_estimators) 和树深度(max_depth)。

各预测模型的最优参数集：

| 损失函数 | 基准模型最优参数 | 损失函数 | 网络搜索指数模型最优参数 |
|---|---|---|---|
| ls | learning_rate=0.1, max_depth=3,n_estimators=20 | ls | learning_rate=0.45, max_depth=3,n_estimators=30 |
| lad | learning_rate=0.05, max_depth=3,n_estimators=190 | lad | learning_rate=0.4, max_depth=3,n_estimators=80 |
| huber | learning_rate=0.1, max_depth=3,n_estimators=20 | huber | learning_rate=0.1, max_depth=3,n_estimators=60 |

利用网格搜索算法遍历各个参数的排列组合进行模型优化，并通过 10 折交叉验证获得最优的模型参数。根据上述小节中数据预处理后划分的训练集训练模型，同样用一阶滞后期序列为输入变量的数据集训练得到基准模型，得到各预测模型的最优参数集的设置，如上表所示。

选取与上节相同的评价指标进行比较分析，并通过对照分析研究了网络搜索

数据的影响。结果如下表所示:

GBDT 基准模型评价指标：

| 基准模型 | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| GBDT-ls | 0.0612 | 0.0771 | 2.0211 |
| GBDT-lad | 0.0575 | 0.0728 | 1.9090 |
| GBDT-huber | 0.0570 | 0.0720 | 2.0212 |

GBDT 网络搜索指数模型评价指标：

| 网络搜索模型 | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| GBDT-ls | 0.0533 | 0.0676 | 1.7750 |
| GBDT-lad | 0.0492 | 0.0639 | 1.6437 |
| GBDT-huber | 0.0529 | 0.0675 | 1.7750 |

通过上面的评价指标数据我们能够有到，基准模型中 GBDT 算法的拟含效果整体较好，其中使用绝对损失函数的预测模型更优。同时我们能够看到，与基准模型对比加入网络搜索关键词指标有助于提升模型的预测精度。在使用网络搜索数据的 GBDT 模型中，将绝对损失函数作为损失函数的模型预测效果要明显高于选择其他损失函数的模型，MAE、RMSE 和 MAPE 的数值均为最小，意味着使用该损失函数的 GBDT 模型更适用于在突发事件下股价变动的预测。

## 4.5 AR 模型、SVR 模型和 GBDT 模型综合比较分析

股票市场的变化特征十分复杂，在前面我们通过时间序列 ARMA 模型了解股价变动的线性特征,然后又利用 SVR 和 GBDT 算法模型探究了其非线性变化，各模型表现出不同的预测效果。

首先我们从医药行业综合股价指数序列进行分析，在不考虑其他因素影响的情况下，选取三种模型中各自表现最好的模型，如下表(4-11)所示。对比以下三个模型我们能够发现，它们都能较好的对综合股价指数进行预测，MAPE 值都在3%以内，且 MAE 和 MSE 的取值也处在较低的一个范围。但是相对于时间序列

AR 模型而言，算法模型 SVR 和 GBDT 表现出的预测精度水平更高，MAPE 值的精度提高了 50%左右，说明算法模型在对股市中股价变动预测研究有着更好的可行性和优越性。

AR、SVR 和 GBDT 基准模型评价指标：

| 模型 | MAE | RMSE | MAPE(%) |
|---|---|---|---|
| ARIMA(1,1,1) | 0.0857 | 0.0996 | 2.9089 |
| SVR-linear | 0.0497 | 0.0632 | 1.6514 |
| GBDT-huber | 0.0575 | 0.0728 | 1.9090 |

基于网络搜索数据的 SVR、GBDT 模型评价指标：

| SVR 模型 | MAE | MSE | MAPE | GBDT 模型 | MAE | MASE | MAPE |
|---|---|---|---|---|---|---|---|
| SVR-linear | 0.0466 | 0.0600 | 1.5450 | GBDT-ls | 0.0533 | 0.0676 | 1.7750 |
| SVR-rbf | 0.0472 | 0.0606 | 1.5640 | GBDT-lad | 0.0492 | 0.0639 | 1.6437 |
| SVR-sigmoid | 0.1144 | 0.1356 | 3.7994 | GBDT-huber | 0.0529 | 0.0675 | 1.7750 |

通过以上对比分析，我们认识到非线性算法模型对股价的预测性能，相较传统的时间序列模型有着很大的优势。本文的研究重点在于探究网络搜索数据对预测突发事件下股价变动影响，即通过引入这一变量，能否提升模型的预测精度。本节中基于 SVR 和 GBDT 算法模型结合网络搜索数据对医药行业综合股价指数进行拟合及例测，预测结果评价指标对上表所示。我们可以看到两种模型的 MAE 和 MAPE 均在一定程度上减少，说明模型预测准确度进步地提升，且 RMSE 值也有所降低，表明预测模型的稳定性也得到了改善。可以发现在模型内部不同参数的调整,对模型的预测能力有很大的影响。在对比使用不同核函数的 SVR 模型和使用不同损失函数的 GBDT 模型后，我们得出基于线性核函数的 SVR 模型和基于 lad 函数的 GBDT 模型预测性能最佳，MAPE 分别为 1.5454%和 1.6437%，其中 SVR 预测模型表现更优。

# 五、结果分析与展望

本文的主题内容是基于突发事件环境下的网络搜索数据对医药行业股票影响的分析与预测研究。首先，通过基于网络搜索数据改进的事件分析法，我们选取了五件使股价波动变化较大的事件，分析它们对医药行业股价的具体影响；然

后通过构建搜索关键词指标，作为输入变量，利用算法模型对股价预测，并与传统时间序列模型进行比较分析，得出以下主要结论：

(1)本次选取的五件突发事件对于医药行业板块的股票带来了巨大的冲击，在事件期窗口内都对股价产生了显著的负向作用，这种影响具有一定的持续性，异常收益率表现为短期内都为负值。

(2)网络搜索数据的时效性较强，能够对突发事件进行监测，并且是可以判断股市变动趋势的先行指标。一般在事件公告的当天，与事件相关的关键词搜索量大幅增加，在一定程度上验证基于百度指数的网络搜索数据对突发事件灵敏的反应及较强的刻画能力。虽然此类重大公共事件的发生不具备可预测性，即突发事件的网络搜索关注度变化发生在事件之后，但在实际应用中可以作为预警指标，利用网络搜索数据建立应对机制。

(3)搜索关键词指数对于突发事件环境下医药行业股票价格变动有着显著的格兰杰因果关系，根据实验结果可知引入该指标作为输入变量能够提高预测精度。同时预测模型的实证结果表明，相对于时间序列 ARIMA 模型，GBDT 和 SVR 模型预测性能更为优异，在上述模型中基于 linear 核函数的 SVR 模型和基于 lad 函数的 GBDT 模型结合网络搜索数据，能够更好地实现在突发事件情况下对医药行业综合股价指数的精准预测，充分把握股票价格变动的非线性特征，其中 SVR 模型的表现更优。

(4)经过研究，网络搜索数据可以提高股价预测的精度，对于突发事件的影响分析与解释有不可忽视的作用。此数据可用于观测股市震荡前兆，通过抓取搜索量数据变化有望防范股票市场非系统性风险。未来，网络搜索数据必将发挥更大的作用。