第3章

金融大数据分析案例

3.1 证券文本信息情感分析

3.1
视频讲解

3.1.1 案例背景

证券市场是一个由众多因素共同作用的复杂动态系统，其中，证券文本信息的作用不容忽视。这类信息涵盖新闻报道、公司公告、分析师评论及社交媒体上的讨论等，既包含客观数据和事实，也融入主观的观点与情绪，对投资者的心理状态和决策行为产生深远影响，进而对证券价格的形成和市场波动造成影响。

进行证券文本信息的情感分析，意在运用自然语言处理技术深挖文本中的情绪倾向及其强度，识别出信息是积极的、消极的还是中立的，并评估其情绪的强弱。这一分析过程使得投资者能够更准确地把握市场情绪的波动，理解舆论氛围的变化，为市场趋势和风险的预测提供新的视角。

此外，结合数学统计和机器学习技术对证券文本信息的情感与股价走势进行深入分析，旨在从海量的文本情感数据中挖掘出与股价变动密切相关的模式和特征，以预测未来的价格变化。这种分析不仅可以帮助投资者更有效地利用信息优势，而且有助于他们制定更加科学合理的投资策略，从而提高投资效益。

本案例以中国A股市场为背景，选取一批具有代表性的上市公司作为分析样本。通过收集和处理一段时间内的证券文本信息及股价数据，进行详尽的情感分析和走势预测。目的是为投资者提供一个有效的分析工具，增强他们对市场动态的洞察力和掌控力，从而在复杂多变的证券市场中做出更加明智的投资决策。

3.1.2 数据来源与运行环境

证券文本信息是与证券市场相关的各种非结构化的文本数据，如信息披露公告、市场舆情资讯、公函报告等，可以反映证券市场的运行情况、投资者的情绪和行为、上市公司的经营状况等，对于证券分析和交易有重要的价值和影响。本案例数据来自金融媒体，包括从2015年1月1日到2015年

10 月 21 日的 27 000 余条信息,包含标题文本、日期等信息,部分数据如表 3.1 所示。

表 3.1 证券文本部分信息

标　　题	日　　期
四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好	2015/1/1
[公司]GQY 视讯参股机器人公司股权降至 11%	2015/1/1
正泰电器	2015/1/1
葛洲坝高管股票账户短线交易:家属误操作	2015/1/1
云南旅游宣传片登陆纽约时报广场	2015/1/1
姚记扑克欲购公司或涉虚假宣传 跨界又生波折	2015/1/1
金利科技上市后迎来首亏 失败收购引发担忧	2015/1/1
上海外滩踩踏事故新华保险启动应急工作小组	2015/1/1
长城汽车哈弗 H5 优惠 2000 元 店内有现车	2015/1/2
江苏舜天官方宣布国安门将张思鹏加盟球队	2015/1/2
国泰君安国际:升新华保险至收集评级	2015/1/2
物产中拓易主　　浙江交通集团	2015/1/3

股票交易数据是指证券市场中产生的各种结构化的交易数据,如股票代码、名称、价格、成交量、成交额、涨跌幅等。本案例数据来自 Tushare 大数据开放社区。

本案例的运行环境是 Anaconda,它是一个用于数据科学的 Python 发行版,可以方便地管理包和环境,同时附带很多常用的数据科学包。编译器是 Jupyter Notebook,它是一个交互式的网页文档,可以将数据分析的代码、图像和文档全部组合到一个 Web 文档中,方便展示和分享结果。

本案例使用一些常用的数据分析库,例如:

- NumPy:提供高效的多维数组对象和相关操作,是数据分析的基础库。
- Pandas:提供强大的数据结构和数据分析工具,可以方便处理各种格式的数据。
- Matplotlib:提供丰富的绘图功能,可以生成各种类型的图表和可视化效果。
- Cnsenti:一个中文情感分析库,可以对文本进行情绪分析和正负情感分析。

3.1.3 分析过程与代码实现

1. 分析过程

(1) 收集加载证券文本信息的数据。

(2) 使用 Cnsenti 库对文本数据进行正负情感分析。Cnsenti 库可以使用默认的词典,也可以导入自定义的词典,可以得到文本中正面和负面情感词的个数和得分。

(3) 获取股票历史价格数据,如开盘价、收盘价、涨跌幅等。使用 Tushare 库获

取股票数据，并保存到 DataFrame 中。

（4）计算情感指数和股价走势之间的相关系数，使用 Matplotlib 库绘制情感指数和股价走势的折线图。

2. 代码实现

1）导入工具库及数据集

首先，通过导入所需的库，包括情感分析库、数据可视化库以及数据处理库，为后续的数据处理、情感分析和可视化做好准备。其次，通过使用 Pandas 库中的 read_excel()函数，从文件中加载证券文本信息数据，将其存储在 DataFrame 中，并通过输出前几行数据来了解数据的结构和内容。具体代码如下。

```
In []:
    # 导入所需的库
    from cnsenti import Sentiment
    import matplotlib.pyplot as plt
    import pandas as pd
    import numpy as np
    from datetime import datetime
In []:
    # 加载证券文本信息数据
    pre_data = pd.read_excel('stocknewsdata.xls')
    print(pre_data.head())
Out[]:
                    title                                              content  \
0  四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好  12 月 31 日，万科 A 尾盘放量拉升并涨停，股价收报 13.90 元。盘后龙虎榜数据显示，该股当日有...
1  [公司]GQY 视讯参股机器人公司股权降至 11%  全景网 12 月 31 日讯 GQY 视讯(300076)周三晚间公告称，公司参股的新世纪机器人因近年来...
2                    正泰电器  公司将出资 1512 万元获得江苏绿城信息技术有限公司 18%的股权，绿城信息是一家专业从事电动汽...
3      葛洲坝高管股票账户短线交易：家属误操作  中国经济网北京 1 月 1 日讯(记者 魏京婷)葛洲坝 12 月 31 日晚间发布公告表示，公司总经理和建生...
4          云南旅游宣传片登陆纽约时报广场  云南旅游宣传片登陆纽约时报广场...

             datetime
0 2015-01-01 00:05:10
1 2015-01-01 01:19:56
2 2015-01-01 06:39:36
3 2015-01-01 07:11:48
4 2015-01-01 12:31:12
```

2) 数据预处理

数据预处理对 DataFrame 中的日期数据格式进行转换,并对部分列的数据类型进行修改。首先,通过将 datetime 列中的日期数据转换为指定的年月日格式,并将转换后的结果存储在名为 date 的新列中,以便后续的操作和分析。其次,将 title 和 content 列的数据类型从 object 类型转换为 string 类型,这有助于更好地处理和分析文本数据。然后,通过查看 DataFrame 的数据类型,确认转换操作是否成功。具体代码如下。

```
In [ ]:
    # 转换日期数据格式
    pre_data['date'] = pre_data['datetime'].apply(lambda x: datetime.
strftime(x,'%Y%m%d'))
    print(pre_data.head())
Out[ ]:
                    title                                              content  \
0  四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好   12 月 31 日,万科 A 尾盘放量拉升并涨停,股价收报 13.90 元。盘后龙虎榜数据显示,该股当日有...
1  [公司]GQY 视讯参股机器人公司股权降至 11%   全景网 12 月 31 日讯 GQY 视讯(300076)周三晚间公告称,公司参股的新世纪机器人因近年来...
2                      正泰电器   公司将出资 1512 万元获得江苏绿城信息技术有限公司 18%的股权,绿城信息是一家专业从事电动汽...
3       葛洲坝高管股票账户短线交易:家属误操作   中国经济网北京 1 月 1 日讯(记者 魏京婷)葛洲坝 12 月 31 日晚间发布公告表示,公司总经理和建生...
4           云南旅游宣传片登陆纽约时报广场   云南旅游宣传片登陆纽约时报广场...

             datetime      date
0 2015-01-01 00:05:10  20150101
1 2015-01-01 01:19:56  20150101
2 2015-01-01 06:39:36  20150101
3 2015-01-01 07:11:48  20150101
4 2015-01-01 12:31:12  20150101
In [ ]:
    # 将 object 类型转换为 string 类型
    pre_data = pre_data.astype({'title':'string'})
    pre_data = pre_data.astype({'content':'string'})
    # 查看各列的数据类型
    pre_data.dtypes
Out[ ]:
    title                string
    content              string
```

```
    datetime      datetime64[ns]
    date                  object
    dtype: object
In []:
    # 将标题 title 转换为 list 类型
    text=pre_data['title'].tolist()
    print(text[:10])
  Out[]:
    ['四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好', '[公司]GQY 视讯参股机器人公司股权降至 11%', '正泰电器', '葛洲坝高管股票账户短线交易:家属误操作', '云南旅游宣传片登陆纽约时报广场', '姚记扑克欲购公司或涉虚假宣传 跨界又生波折', '金利科技上市后迎来首亏,失败收购引发担忧', '上海外滩踩踏事故新华保险启动应急工作小组', '长城汽车哈弗 H5 优惠 2000 元 店内有现车', '江苏舜天官方宣布国安门将张思鹏加盟球队']
```

接下来,使用情感分析库中的 Sentiment 对象对文本数据进行情感分析,并将计算得到的正负情感值添加到 DataFrame 中。首先创建一个 Sentiment 对象,并传入参数,包括正面词典文件路径、负面词典文件路径、是否融合自定义词典,以及词典文件的编码方式。接着,通过循环遍历文本数据中的每个元素,利用 sentiment_calculate()方法计算每个文本数据的正负情感值,并将结果存储在 pos 和 neg 列表中。然后,将计算得到的正负情感值添加到 DataFrame 中的新列 pos 和 neg 中,并通过输出 DataFrame 的前几行来查看结果。具体代码如下。

```
In []:
    # 创建 Sentiment 对象,并传入参数
    senti = Sentiment(pos='CNKI_positiveEmotionWords.txt',
                                                # 正面词典 txt 文件相对路径
                neg='CNKI_negativeEmotionWords.txt',
                                                # 负面词典 txt 文件相对路径
                merge=True,    # 融合 cnsenti 自带词典和用户导入的自定义词典
                encoding='utf-8')   # 两个 txt 文件均为 utf-8 编码
In []:
    # 运用 sentiment_calculate()进行正负情感计算
    length=len(text)
    pos=[]
    neg=[]
    for i in range(length):
        emotion = senti.sentiment_calculate(text[i])
        pos.append(emotion['pos'])
        neg.append(emotion['neg'])
```

```
    # 将正负情感值增加到 DataFrame
    pre_data['pos'] = pos
    pre_data['neg'] = neg
    print(pre_data.head())
Out[]:
                    title                                              content  \
0  四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好  12 月 31 日,万科 A 尾盘放量拉升并涨停,股价收报 13.90 元。盘后龙虎榜数据显示,该股当日有...
1  [公司]GQY 视讯参股机器人公司股权降至 11%  全景网 12 月 31 日讯 GQY 视讯(300076)周三晚间公告称,公司参股的新世纪机器人因近年来...
2                    正泰电器  公司将出资 1512 万元获得江苏绿城信息技术有限公司 18%的股权,绿城信息是一家专业从事电动汽...
3      葛洲坝高管股票账户短线交易:家属误操作  中国经济网北京 1 月 1 日讯(记者 魏京婷)葛洲坝 12 月 31 日晚间发布公告表示,公司总经理和建生...
4          云南旅游宣传片登陆纽约时报广场  云南旅游宣传片登陆纽约时报广场...

             datetime      date  pos   neg
0 2015-01-01 00:05:10  20150101  8.0  20.0
1 2015-01-01 01:19:56  20150101  0.0   0.0
2 2015-01-01 06:39:36  20150101  0.0   0.0
3 2015-01-01 07:11:48  20150101  0.0   0.0
4 2015-01-01 12:31:12  20150101  0.0   0.0
```

对经过情感分析后的正负情感值进行 Z-score 标准化处理,并计算每天的情感值。首先,通过计算每个情感值列的均值和标准差,对正负情感值进行 Z-score 标准化处理,以消除不同列之间的量纲差异,使得数据更加可比较和可解释。接着,通过计算每天的文本数量和每天的情感值总和,分别使用 groupby()方法对日期进行分组,以获取每天的文本数量和情感值总和。然后,将每天的情感值和文本数量存储在新的 DataFrame 中,并计算每天的平均情感值,将其线性扩大 10 倍,以便更直观地展示每天的情感趋势。再通过输出新的 DataFrame 来查看每天的情感值和文本数量的汇总情况。具体代码如下。

```
In []:
    # 对正负情感值进行 z-score 标准化处理
    pre_data['norm_pos'] = (pre_data['pos']-np.mean(pre_data['pos']))/(np.std(pre_data['pos']))
    pre_data['norm_neg'] = (pre_data['neg']-np.mean(pre_data['neg']))/(np.std(pre_data['neg']))
    # 最终情感值=正情感值-负情感值
```

```
    pre_data['emotion']=pre_data['norm_pos']-pre_data['norm_neg']
    print(pre_data[:10])
Out[]:
                    title                                              content  \
0  四机构逾 6 亿资金出逃万科 A 地产股仍获券商唱好  12 月 31 日,万科 A 尾盘放量拉升并涨停,股价收报 13.90 元。盘后龙虎榜数据显示,该股当日有...
1  [公司]GQY 视讯参股机器人公司股权降至 11%  全景网 12 月 31 日讯 GQY 视讯(300076)周三晚间公告称,公司参股的新世纪机器人因近年来...
2                    正泰电器  公司将出资 1512 万元获得江苏绿城信息技术有限公司 18%的股权,绿城信息是一家专业从事电动汽...
3      葛洲坝高管股票账户短线交易:家属误操作  中国经济网北京 1 月 1 日讯(记者 魏京婷)葛洲坝 12 月 31 日晚间发布公告表示,公司总经理和建生...
4          云南旅游宣传片登陆纽约时报广场  云南旅游宣传片登陆纽约时报广场...
5   姚记扑克欲购公司或涉虚假宣传 跨界又生波折  财信网(记者 陶炜 许立婷)一直在寻找跨界转型机会的姚记扑克(002605)近日宣布,公司准...
6    金利科技上市后迎来首亏 失败收购引发担忧  财信网(记者 彭彬)2010 年上市的金利科技(002464)迎来上市后的首亏。金利科技昨日发...
7    上海外滩踩踏事故新华保险启动应急工作小组  新浪财经讯 1 月 1 日消息,12 月 31 日晚上海外滩发生踩踏事故后,新华保险立即启动应急工作小组...
8   长城汽车哈弗 H5 优惠 2000 元 店内有现车  编辑点评:长城哈弗 H5 是基于哈弗的生产平台衍生出来的一款新车型,它的主体结构与现款哈弗并未有...
9      江苏舜天官方宣布国安门将张思鹏加盟球队  江苏舜天队在其官方微博写道:"今天上午,江苏国信舜天足球俱乐部与球员张思鹏就加盟一事达成一致...

             datetime      date   pos   neg   norm_pos   norm_neg   emotion
0 2015-01-01 00:05:10  20150101   8.0  20.0   0.028309   5.859919 -5.831611
1 2015-01-01 01:19:56  20150101   0.0   0.0  -0.303539  -0.350969  0.047430
2 2015-01-01 06:39:36  20150101   0.0   0.0  -0.303539  -0.350969  0.047430
3 2015-01-01 07:11:48  20150101   0.0   0.0  -0.303539  -0.350969  0.047430
4 2015-01-01 12:31:12  20150101   0.0   0.0  -0.303539  -0.350969  0.047430
5 2015-01-01 14:14:52  20150101   5.0   9.0  -0.096134   2.443931 -2.540065
6 2015-01-01 14:15:41  20150101  12.0   5.0   0.194232   1.201753 -1.007521
7 2015-01-01 15:24:33  20150101   5.0  15.0  -0.096134   4.307197 -4.403332
8 2015-01-02 02:54:52  20150102  11.0   0.0   0.152751  -0.350969  0.503720
9 2015-01-02 13:25:43  20150102   0.0   0.0  -0.303539  -0.350969  0.047430
```

```
In [ ]:
    # 计算每天的文本数量
    numberday = pre_data['emotion'].groupby(pre_data['date']).count()
    # 对每天的情感值进行汇总求和
    emotionday = pre_data['emotion'].groupby(pre_data['date']).sum()
In [ ]:
    # 新建每天情感值与文本数量的 DataFrame
    new_data=pd.DataFrame()
    new_data['emotion'] = emotionday
    new_data['number'] = numberday
    # 计算每天的平均情感值并线性扩大 10 倍
    new_data['avgemotion'] = new_data['emotion']/new_data['number'] * 10
    print(new_data)
Out[ ]:
              emotion  number   avgemotion
  date
20150101  -13.592810       8  -16.991012
20150102    0.764503       3    2.548344
20150103    0.911375       3    3.037915
20150104    3.632877      21    1.729941
20150105   -1.105290     127   -0.087031
   ...           ...     ...         ...
20151017   -9.281850      48   -1.933719
20151018   -2.334995      30   -0.778332
20151019   -2.530997     118   -0.214491
20151020    5.630377     150    0.375358
20151021    3.993248     136    0.293621
[292 rows x 3 columns]
```

3) 计算情感指数和股价走势之间的相关系数

为分析情感值与股票价格涨跌幅之间的相关性,以探索情感对股票市场的影响程度,利用 Tushare 库获取证券交易数据,并结合之前计算得到的每天的情感值数据,进行相关性分析。首先,导入 Tushare 库,并利用用户提供的 TOKEN 创建 Tushare 对象。接着,通过 Tushare 对象获取股票代码为'000001.SZ'(深圳市场上的平安银行)的交易数据,时间范围为 2015 年 1 月 1 日至 2015 年 10 月 21 日,并按交易日期进行排序。然后,将交易日期的列名改为'date',以便后续合并数据。接下来,

利用 Pandas 的 merge()函数将股票交易数据和情感值数据按照日期进行合并，形成最终的数据集。然后，计算最终数据集的相关系数矩阵，并提取情感值平均值(avgemotion)和股票价格涨跌幅(pct_chg)之间的相关系数。最后，输出相关系数矩阵和情感值与股票价格涨跌幅之间的相关系数。具体代码如下。

```
In []:
    # 导入 Tushare 库
    import tushare as ts
    # 运用 TOKEN 创建 Tushare 对象
    pro = ts.pro_api('bcf5ca838c2011c8ae95338962d6dec623ed6d5ff92d174f11b982fd')
    # 获取证券交易数据
    stock = pro.daily(ts_code='000001.SZ', start_date='20150101', end_
date='20151021')
    # 根据交易日期排序
    stock = stock.sort_values(by='trade_date')
    print(stock)
Out[]:
      ts_code   trade_date   open   high    low   close  pre_close  change  \
192  000001.SZ   20150105   15.99  16.28  15.60  16.02    15.84      0.18
191  000001.SZ   20150106   15.85  16.39  15.55  15.78    16.02     -0.24
190  000001.SZ   20150107   15.56  15.83  15.30  15.48    15.78     -0.30
189  000001.SZ   20150108   15.50  15.57  14.90  14.96    15.48     -0.52
188  000001.SZ   20150109   14.90  15.87  14.71  15.08    14.96      0.12
...     ...        ...      ...    ...    ...    ...      ...        ...
 4   000001.SZ   20151015   10.95  11.17  10.94  11.17    10.98      0.19
 3   000001.SZ   20151016   11.21  11.30  11.16  11.23    11.17      0.06
 2   000001.SZ   20151019   11.27  11.35  11.15  11.26    11.23      0.03
 1   000001.SZ   20151020   11.20  11.38  11.19  11.32    11.26      0.06
 0   000001.SZ   20151021   11.29  11.75  11.18  11.22    11.32     -0.10

     pct_chg      vol         amount
192     1.14  2860436.43  4.565388e+06
191    -1.50  2166421.40  3.453446e+06
190    -1.90  1700120.67  2.634796e+06
189    -3.36  1407714.21  2.128003e+06
188     0.80  2508500.23  3.835378e+06
..       ...      ...          ...
 4      1.73   485900.50  5.391037e+05
 3      0.54   570001.45  6.404889e+05
 2      0.27   723371.53  8.149110e+05
```

```
1      0.53     715163.28     8.076424e+05
0     -0.88    1332629.04     1.523592e+06
 [193 rows x 11 columns]
 In []:
    stock = stock.rename(columns={'trade_date': 'date'})
    # 合并 stock 与 new_data
    finally_data = pd.merge(stock,new_data,on=['date'],how='left')
    print(finally_data)
 Out[]:
       ts_code      date    open   high    low   close  pre_close  change  \
  0   000001.SZ  20150105  15.99  16.28  15.60  16.02     15.84       0.18
  1   000001.SZ  20150106  15.85  16.39  15.55  15.78     16.02      -0.24
  2   000001.SZ  20150107  15.56  15.83  15.30  15.48     15.78      -0.30
  3   000001.SZ  20150108  15.50  15.57  14.90  14.96     15.48      -0.52
  4   000001.SZ  20150109  14.90  15.87  14.71  15.08     14.96       0.12
 ...      ...        ...    ...    ...    ...    ...       ...        ...
 188  000001.SZ  20151015  10.95  11.17  10.94  11.17     10.98       0.19
 189  000001.SZ  20151016  11.21  11.30  11.16  11.23     11.17       0.06
 190  000001.SZ  20151019  11.27  11.35  11.15  11.26     11.23       0.03
 191  000001.SZ  20151020  11.20  11.38  11.19  11.32     11.26       0.06
 192  000001.SZ  20151021  11.29  11.75  11.18  11.22     11.32      -0.10

      pct_chg      vol         amount       emotion  number  avgemotion
  0      1.14  2860436.43  4.565388e+06  -1.105290    127     -0.087031
  1     -1.50  2166421.40  3.453446e+06   1.446429    150      0.096429
  2     -1.90  1700120.67  2.634796e+06  -6.277145    123     -0.510337
  3     -3.36  1407714.21  2.128003e+06  -7.795138    123     -0.633751
  4      0.80  2508500.23  3.835378e+06  12.712928    145      0.876754
 ...      ...         ...           ...        ...    ...           ...
 188     1.73   485900.50  5.391037e+05  -4.201336    190     -0.221123
 189     0.54   570001.45  6.404889e+05   1.883506    143      0.131714
 190     0.27   723371.53  8.149110e+05  -2.530997    118     -0.214491
 191     0.53   715163.28  8.076424e+05   5.630377    150      0.375358
 192    -0.88  1332629.04  1.523592e+06   3.993248    136      0.293621
 [193 rows x 14 columns]
 In []:
    # 计算相关系数矩阵
    corr_matrix = finally_data.corr(method='pearson')
```

```
    print('corr_matrix:\n',corr_matrix)
    # 提取 avgemotion 和 pct_chg 之间的相关系数
    corr_xy = corr_matrix.loc['avgemotion', 'pct_chg']
    print('corr_xy:\n',corr_xy)
Out[]:
corr_matrix:
                open      high       low     close  pre_close    change  \
open        1.000000  0.989640  0.991799  0.982183   0.994101  0.042047
high        0.989640  1.000000  0.985701  0.993278   0.985979  0.132190
low         0.991799  0.985701  1.000000  0.987939   0.987587  0.099473
close       0.982183  0.993278  0.987939  1.000000   0.977421  0.203652
pre_close   0.994101  0.985979  0.987587  0.977421   1.000000 -0.007819
change      0.042047  0.132190  0.099473  0.203652  -0.007819  1.000000
pct_chg     0.028310  0.115047  0.083645  0.186310  -0.022846  0.987557
vol         0.569371  0.636395  0.530626  0.602975   0.569725  0.213776
amount      0.680253  0.741215  0.649851  0.712554   0.677394  0.233475
emotion    -0.060667 -0.041861 -0.069979 -0.064129  -0.048458 -0.078958
number     -0.110004 -0.100517 -0.102999 -0.102025  -0.096421 -0.036067
avgemotion -0.047676 -0.033929 -0.052317 -0.052003  -0.036589 -0.076570

              pct_chg       vol    amount   emotion    number  avgemotion
open         0.028310  0.569371  0.680253 -0.060667 -0.110004   -0.047676
high         0.115047  0.636395  0.741215 -0.041861 -0.100517   -0.033929
low          0.083645  0.530626  0.649851 -0.069979 -0.102999   -0.052317
close        0.186310  0.602975  0.712554 -0.064129 -0.102025   -0.052003
pre_close   -0.022846  0.569725  0.677394 -0.048458 -0.096421   -0.036589
change       0.987557  0.213776  0.233475 -0.078958 -0.036067   -0.076570
pct_chg      1.000000  0.207488  0.219509 -0.092848 -0.022809   -0.089482
vol          0.207488  1.000000  0.981644  0.079991 -0.031304    0.040397
amount       0.219509  0.981644  1.000000  0.035107 -0.040788    0.004446
emotion     -0.092848  0.079991  0.035107  1.000000  0.032104    0.986909
number      -0.022809 -0.031304 -0.040788  0.032104  1.000000    0.020609
avgemotion  -0.089482  0.040397  0.004446  0.986909  0.020609    1.000000
corr_xy:
-0.08948154974788744
```

4) 数据可视化

为可视化情感值和证券价格涨跌幅之间的波动情况,更直观地观察两者之间的关系,绘制情感波动图和证券价格波动图,如图 3.1 所示,以展示情感值和证券价格之间的波动情况。首先,创建一个大小为(10,8)的图形对象,并将其分为两个子图。第一个子图用于绘制情感波动图,标题为'emotion fluctuation',横轴为日期,纵轴为情感值,线条颜色为红色。第二个子图用于绘制证券价格波动图,标题为'stock fluctuation',横轴同样为日期,纵轴为证券价格涨跌幅(pct_chg),线条颜色为蓝色。最后,通过调用 plt.show()方法显示绘制好的图形。整体而言,作用是可视化情感值和证券价格涨跌幅之间的波动情况,以便更直观地观察两者之间的关系。具体代码如下。

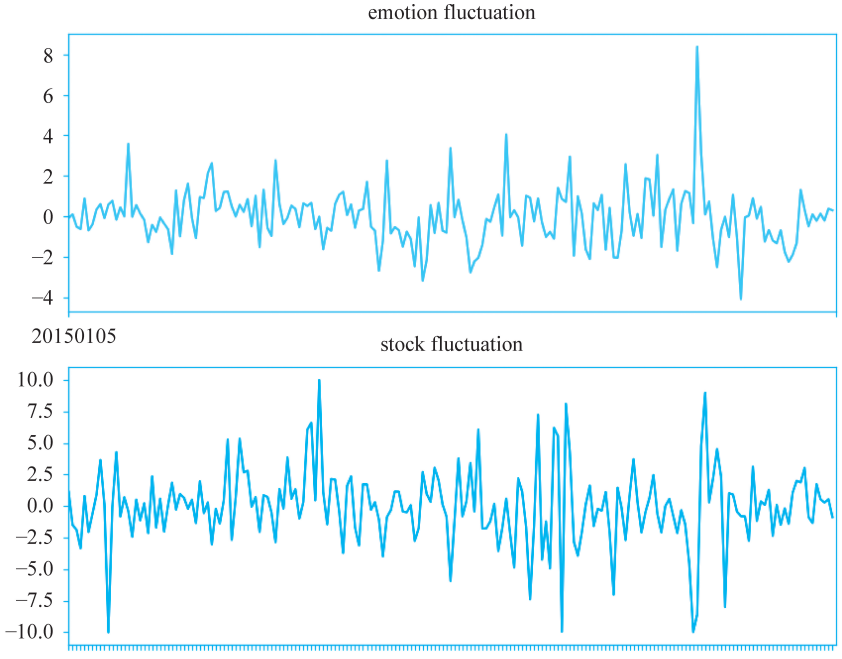

图 3.1 情绪与股价波动

```
In []:
    # 绘制情感波动图和证券价格波动图
    fig=plt.figure(figsize=(10,8))
    ax1 = fig.add_subplot(2,1,1)              # 绘制第 1 幅子图
    plt.title('emotion fluctuation')
    plt.xlim(0,len(finally_data))
    plt.xticks([0,len(finally_data)])
    plt.plot(finally_data['date'],finally_data['avgemotion'],'r-')
    ax2 = fig.add_subplot(2,1,2)              # 绘制第 2 幅子图
    plt.title('stock fluctuation')
```

```
        plt.xlim(0,len(finally_data))
        plt.plot(finally_data['date'],finally_data['pct_chg'],'b-')
        plt.show()
    Out[]:
```

3.1.4　小结

本案例揭示证券文本信息的情感倾向与股价波动之间的相互关系，突显文本情感分析在金融领域的应用价值。然而，研究在某些方面还存在局限，例如，使用的数据量有限，分析的时间跨度短，以及忽略可能影响结果的其他变量等。为提升研究的准确性和可靠性，建议未来的研究中扩充数据集规模，延伸研究的时间范围，并纳入更多可能影响股价变动的因素，从而增加分析的深度与广度。这些改进将有助于更全面地理解文本情感分析在预测股价走势中的潜力和局限，为投资者提供更精准的决策支持。

3.2　信贷风险预测

3.2-1
视频讲解

3.2.1　案例背景

信贷风险预测是金融领域中的一个核心任务，其目的是通过分析借款人的个人资料、银行交易记录、信用卡账单、上网行为等数据，来预测借款人可能出现的逾期还款行为。这一预测对于金融机构在进行信用评估和授信决策时至关重要，它直接关联到机构的营利能力和稳定性，同时也影响到借款人的信用记录和个人福祉。

传统的信贷风险评估方法主要依赖人工经验和简单的规则判断，或者使用基于统计学的模型，例如，逻辑回归和决策树。然而，这些方法面临多个挑战：人工判断可能缺乏客观性和准确性，难以适应市场和消费者需求的快速变化；统计模型难以捕捉数据中的复杂非线性关系，且依赖繁重的特征工程和调参工作；数据问题如缺失值、异常值和噪声等可能降低模型性能；高维度和大规模的数据集增加模型的计算和存储负担；多样化的数据来源和类型要求采用不同的处理和建模策略。

为应对这些挑战，机器学习和深度学习技术被广泛引入信贷风险预测中。这些先进的方法带来显著的优势：它们能够自动从数据中提取有用的特征和模式，无须依赖人工干预或者先验知识；能够有效处理高维度、大规模、多源和多类型的数据，增强模型的表达和泛化能力；通过采用神经网络、卷积神经网络（CNN）、循环神经网络（RNN）、长短期记忆网络（LSTM）、自编码器、生成对抗网络（GAN）等多样的网络结构和算法，可以更好地适应不同数据特性和预测任务的需求。

3.2.2 数据来源与运行环境

本案例所用数据集是一个关于信用卡客户违约情况的数据集,来源于 UCI 机器学习库,由 Yeh I. C.和 Lien C. H.在 2009 年发表的论文中使用。该数据集包含来自中国台湾地区的信用卡持卡人的个人信息以及从 2005 年 4 月到 2005 年 9 月的信用卡信息。共有 30 000 条记录,每条记录有 25 个变量,包括客户 ID、信用额度、性别、教育程度、婚姻状况、年龄、还款状态、账单金额、还款金额以及是否违约等。数据集的目标变量是 dpnm(default.payment.next.month),表示客户是否发生逾期 30 天以上的违约行为,1 表示是,0 表示否。

该数据集用于探索信用卡客户的违约概率与不同的人口统计和信用数据之间的关系,以及建立机器学习或深度学习的模型来预测客户的违约风险。

本案例的运行环境是 Anaconda,使用一些常用的数据分析库,例如:

- NumPy:用于进行高效的数值计算。
- Pandas:用于进行数据的读取、处理和分析。
- Matplotlib 和 Seaborn:用于进行数据的可视化。
- Scikit-learn:用于进行机器学习的数据预处理、模型训练和评估。
- TensorFlow 和 Keras:用于构建和训练深度学习的神经网络。

3.2.3 分析过程与代码实现

1. 分析过程

(1) 导入数据:加载 Default-of-Credit-Card-Clients 数据集。

(2) 主训练集探索:对主训练集进行基本的数据探索,如查看数据维度、缺失值、异常值、数据分布等,对数据进行可视化分析,如绘制柱状图、箱线图、散点图等,探索不同特征与目标变量之间的关系和区分度。

(3) 利用辅助训练集信息:对辅助训练集进行特征工程,如计算统计特征、分箱特征、交叉特征等,并将新生成的特征与主训练集进行合并。

(4) 特征筛选:对合并后的训练集进行特征筛选,如剔除缺失值过多或相关性过高的特征。

(5) 模型训练与预测:选择合适的机器学习或深度学习的算法,如逻辑回归、随机森林、梯度提升树、神经网络,对训练集进行模型训练,并对测试集进行模型预测。使用一些评价指标,如准确率、召回率、F1 值等,来评估模型的性能和效果。使用一些优化方法,如网格搜索、交叉验证等,来调整模型的参数和超参数,提高模型的泛化能力。

2. 代码实现

1) 导入工具库及数据集

加载所需的 Python 库,并加载信贷数据。首先,导入 Pandas、NumPy、Seaborn、Matplotlib.pyplot 等库,用于数据处理、可视化和建模。然后,通过 pd.read_csv()函数

从名为'default-of-credit-card-clients.csv'的 CSV 文件中加载信贷数据，将数据存储在 DataFrame 对象 df 中。具体代码如下。

```
In []:
    # 加载所需的库
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.preprocessing import StandardScaler
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import classification_report, confusion_
matrix, ConfusionMatrixDisplay
    %matplotlib inline
In []:
    # 加载信贷数据
    df = pd.read_csv('default-of-credit-card-clients.csv')
```

2）探索性数据分析

通过 df.head()函数查看数据的前 5 行，以了解数据的结构和内容。接着，通过 df.info()函数查看数据集的基本信息，包括每列的数据类型和非空值数量，以及通过 df.describe()函数查看数据集的常用统计信息，如均值、标准差、最小值、最大值等。然后，通过 df = df.drop('ID', axis = 1)删除数据中的 ID 列，因为 ID 通常是唯一标识符，对数据分析和建模没有实际意义。接下来，通过 df.duplicated().sum()统计数据中的重复值数量，发现有 35 个重复条目。通过 df = df.drop_duplicates()删除重复值，确保数据集中不含有重复条目。通过这些数据清洗步骤，可以保证数据集的质量，减少数据分析和建模过程中的干扰因素，从而更好地进行后续的数据探索和建模分析。具体代码如下。

```
In []:
    # 查看前 5 行数据
    df.head()
Out[]:
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	dpnm
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

```
5 rows × 25 columns
In []:
    df.info()
Out[]:
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 30000 entries, 0 to 29999
    Data columns (total 25 columns):
     #    Column      Non-Null Count       Dtype
    ---   ------      --------------       -----
     0    ID          30000 non-null       int64
     1    LIMIT_BAL   30000 non-null       int64
     2    SEX         30000 non-null       int64
     3    EDUCATION   30000 non-null       int64
     4    MARRIAGE    30000 non-null       int64
     5    AGE         30000 non-null       int64
     6    PAY_1       30000 non-null       int64
     7    PAY_2       30000 non-null       int64
     8    PAY_3       30000 non-null       int64
     9    PAY_4       30000 non-null       int64
     10   PAY_5       30000 non-null       int64
     11   PAY_6       30000 non-null       int64
     12   BILL_AMT1   30000 non-null       int64
     13   BILL_AMT2   30000 non-null       int64
     14   BILL_AMT3   30000 non-null       int64
     15   BILL_AMT4   30000 non-null       int64
     16   BILL_AMT5   30000 non-null       int64
     17   BILL_AMT6   30000 non-null       int64
     18   PAY_AMT1    30000 non-null       int64
     19   PAY_AMT2    30000 non-null       int64
     20   PAY_AMT3    30000 non-null       int64
     21   PAY_AMT4    30000 non-null       int64
     22   PAY_AMT5    30000 non-null       int64
     23   PAY_AMT6    30000 non-null       int64
     24   dpnm        30000 non-null       int64
    dtypes: int64(25)
    memory usage: 5.7 MB
```

从以上信息来看，数据集有 30 000 行、25 列，没有缺失值。

```
In []:
    # 查看数据集的常用统计信息
    df.describe()
Out[]:
ID    LIMIT_BAL    SEX    EDUCATION    MARRIAGE    AGE
      PAY_1    PAY_2    PAY_3    PAY_4    ...    BILL_AMT4
      BILL_AMT5    BILL_AMT6    PAY_AMT1    PAY_AMT2
      PAY_AMT3    PAY_AMT4    PAY_AMT5    PAY_AMT6
      dpnm
count    30000.000000    30000.000000    30000.000000    30000.000000
         30000.000000    30000.000000    30000.000000    30000.000000
         30000.000000    30000.000000    ...             30000.000000
         30000.000000    30000.000000    30000.000000    3.000000e+04
         30000.00000     30000.000000    30000.000000    30000.000000
         30000.000000
mean    15000.500000  167484.322667  1.603733  1.853133  1.551867  35.485500
        -0.016700  -0.133767  -0.166200  -0.220667  ...  43262.948967
        40311.400967  38871.760400  5663.580500  5.921163e+03
        5225.68150  4826.076867  4799.387633  5215.502567
        0.221200
std    8660.398374    129747.661567    0.489129    0.790349    0.521970    9.217904
       1.123802    1.197186    1.196868    1.169139    ...    64332.856134
       60797.155770    59554.107537    16563.280354    2.304087e+04
       17606.96147    15666.159744    15278.305679    17777.465775
       0.415062
min    1.000000  10000.000000  1.000000  0.000000  0.000000  21.000000  -2.000000
       -2.000000  -2.000000  -2.000000  ...  -170000.000000  -81334.000000
       -339603.000000  0.000000  0.000000e+00  0.00000  0.000000  0.000000
       0.000000  0.000000
25%    7500.750000    50000.000000    1.000000    1.000000    1.000000    28.000000
       -1.000000    -1.000000    -1.000000    -1.000000    ...    2326.750000
       1763.000000    1256.000000    1000.000000    8.330000e+02
       390.00000    296.000000    252.500000    117.750000    0.000000
```

```
50%    15000.500000    140000.000000    2.000000    2.000000    2.000000    34.000000
       0.000000    0.000000    0.000000    0.000000    ...    19052.000000
       18104.500000    17071.000000    2100.000000    2.009000e+03
       1800.00000    1500.000000    1500.000000    1500.000000
       0.000000
75%    22500.250000    240000.000000    2.000000    2.000000    2.000000    41.000000
       0.000000    0.000000    0.000000    0.000000    ...    54506.000000
       50190.500000    49198.250000    5006.000000    5.000000e+03
       4505.00000    4013.250000    4031.500000    4000.000000
       0.000000
max    30000.000000  1000000.000000  2.000000  6.000000  3.000000  79.000000
       8.000000  8.000000  8.000000  8.000000  ...  891586.000000
       927171.000000  961664.000000  873552.000000  1.684259e+06
       896040.00000  621000.000000  426529.000000  528666.000000
       1.000000
8 rows × 25 columns
In []:
    # 删除 ID 列
    df = df.drop('ID', axis = 1)
```

在下面的代码中检查重复项。可以确认没有缺失值或重复条目。

```
In []:
    # 统计重复值数量
    df.duplicated().sum()
Out[]:
    35
In []:
    # 删除重复值
    df = df.drop_duplicates()
In []:
    df.info()
Out[]:
    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 29965 entries, 0 to 29999
    Data columns (total 24 columns):
```

```
 #    Column        Non-Null Count     Dtype
---   ------        --------------     -----
 0    LIMIT_BAL     29965 non-null     int64
 1    SEX           29965 non-null     int64
 2    EDUCATION     29965 non-null     int64
 3    MARRIAGE      29965 non-null     int64
 4    AGE           29965 non-null     int64
 5    PAY_1         29965 non-null     int64
 6    PAY_2         29965 non-null     int64
 7    PAY_3         29965 non-null     int64
 8    PAY_4         29965 non-null     int64
 9    PAY_5         29965 non-null     int64
10    PAY_6         29965 non-null     int64
11    BILL_AMT1     29965 non-null     int64
12    BILL_AMT2     29965 non-null     int64
13    BILL_AMT3     29965 non-null     int64
14    BILL_AMT4     29965 non-null     int64
15    BILL_AMT5     29965 non-null     int64
16    BILL_AMT6     29965 non-null     int64
17    PAY_AMT1      29965 non-null     int64
18    PAY_AMT2      29965 non-null     int64
19    PAY_AMT3      29965 non-null     int64
20    PAY_AMT4      29965 non-null     int64
21    PAY_AMT5      29965 non-null     int64
22    PAY_AMT6      29965 non-null     int64
23    dpnm          29965 non-null     int64
dtypes: int64(24)
memory usage: 5.7 MB
```

利用 Seaborn 库绘制数据集中的三个特征('AGE'、'BILL_AMT4'和'PAY_AMT6')的核密度直方图，并通过相关热力图来查看变量之间的相关性。首先，使用 sns.displot()函数分别绘制'AGE'、'BILL_AMT4'和'PAY_AMT6'三个特征的核密度直方图，分别如图 3.2～图 3.4 所示，其中，参数 bins 指定直方图的柱子数量，参数 kde 设为 True 表示同时显示核密度估计曲线。接着，使用 plt.figure(figsize = (12,8))设定绘图的尺寸，然后利用 sns.heatmap()函数绘制数据集中所有特征之间的相关热力图，如图 3.5 所示，其中，颜色深浅表示相关性的强弱，越浅表示相关性越强，越深表示相关性越弱。这些图形有助于我们更直观地了解数据集中各个特征之间的分布情况和相互关系。具体代码如下。

```
In [ ]:
    # 绘制'AGE'核密度直方图
    sns.displot(df['AGE'], bins = 20, kde = True)
Out[ ]:
```

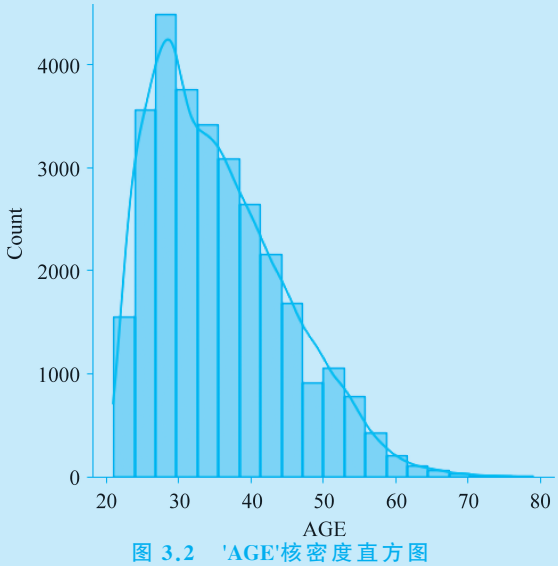

图 3.2 'AGE'核密度直方图

```
In [ ]:
    # 绘制'BILL_AMT4'核密度直方图
    sns.displot(df['BILL_AMT4'], bins = 30, kde = True)
Out[ ]:
```

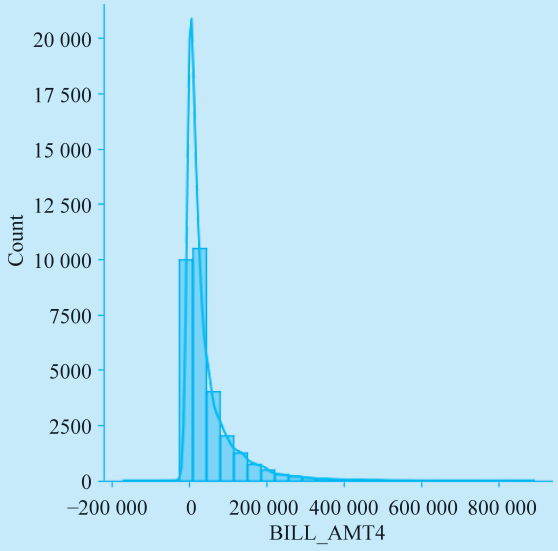

图 3.3 'BILL_AMT4'核密度直方图

```
In [ ]:
```