济南大学

毕 业 论 文

题　　目 基于文本情感倾向的机器学习股价波动预测

学　　院 数学科学学院

专　　业 金融数学

班　　级 金数 1501

学　　生 陈鑫冶

学　　号 20150908151

指导教师 陈文娟

二〇一九年五月二十日

摘　要

针对当下股市预测效果欠佳的问题，本文基于新闻标题情感倾向特征以及算法交易策略，构建了一套有效的解决方案。

在理论研究层面，详细阐述了文本挖掘和机器学习算法的实现原理。针对文本挖掘算法，介绍了本文使用的核心算法 TF-IDF 关键词提取技术，LDA 主题建模及关联规则挖掘隐式信息等方法。针对预测算法，介绍了决策树、支持向量机、朴素贝叶斯网络、GDBT 等统计机器学习方法和双向 LSTM 等深度学习方法。

在实证分析层面，介绍自然语言处理技术的基本理论，完整地阐述了项目的数据预处理、情感分析、特征建模、通过融合多种算法的交易策略以及回测机制的构建流程。并通过资金曲线描述盈利情况、净资产变化情况等合理分析回测结果。

在创新层面上，本文分析了新闻标题情感对于股价波动的影响，并基于情感分析提取新闻标题的情感因子，结合情感因子和股价技术指标为特征，构建融合了多种算法的集成模型预测 2012 至 2016 年的股价波动情况，模型准确率高达 87%，并通过模型的预测结果设计交易策略。为了模拟中国股市的 t+1 交易制度，严格按照手续费、滑点、印花税及涨跌停制度等情况设计回测程序，模拟交易策略在实盘运行的情况，回测结果包括最大回撤率、收益率期望、资金曲线等评估。最后依据回测结果分析算法交易在实盘中的优势和不足及其影响因素，为投资者投资决策提供了宝贵的参考。

关键词：自然语言处理；情感分析；机器学习；算法交易

ABSTRACT

In order to address the problem of poor forecasting effect of the current stock market, this paper constructs an effective solution based on the emotional characteristics of news headlines and algorithmic trading strategy.

At the level of theoretical research, the implementation principles of text mining and machine learning algorithms are elaborated in detail. For text mining algorithm, this paper introduces the core algorithm TF-IDF keyword extraction technology, LDA topic modeling and association rules mining implicit information and other methods. For prediction algorithms, statistical machine learning methods such as decision tree, support vector machine, naive Bayesian network, GDBT and two-way LSTM are introduced.

At the empirical level, the basic theory of natural language processing technology is introduced, and the data preprocessing, emotional analysis, feature modeling, transaction strategy by integrating multiple algorithms and the construction process of test-back mechanism are described. And through the capital curve to describe the profitability, changes in net assets and other reasonable analysis of the test results.

On the innovative level, this paper analyses the impact of news headline emotion on stock price volatility, extracts the emotional factors of news headlines based on emotional analysis, combines emotional factors and stock price technical indicators as characteristics, and constructs an ensemble model integrating various algorithms to predict stock price volatility between 2012 and 2016. The prediction accuracy of the model is 87%, and the trading strategy is designed through the prediction results of the model. In order to simulate the t+1 trading system in China's stock market, a test-back program is designed strictly according to the handling fee, slippage, stamp duty, rise-stop and fall-stop restriction, simulating the operation of trading strategies in the real market, and the test-back results include maximum drawdown, return rate, capital curve and so on. Finally, the advantages and disadvantages of the algorithm trading in the real market and its influencing factors are analyzed based on the test-back results. It provides valuable references for investors to make investment decisions.

Key words：Natural Language Processing；Emotional Analysis；Machine Learning；Algorithmic Transaction

目录

1 前言

1.1 板块选题背景和意义

股票市场是股票转让和流通的金融场所，同时高风险性伴随着的高利润，其市场结构和交易活动较一级市场更为复杂和活跃。鉴于其对经济产生的巨大影响力，股票市场也是一个国家或地区经济的金融活动的寒暑表。如今，对股票市场的预测从原来的定性分析到定量分析[1]，投资者已经不再单纯依据公司的业绩和市盈率来判断未来股价的趋势，更多的是综合宏观和微观的量化因素构建模型来判断股价的走势。

随着人工智能技术的普及与发展，机器学习如今已经应用于人们的生活很多方面并取得重大突破，例如无人驾驶，推荐系统，自然语言处理，语音识别，图像识别以及图像生成等。机器学习包括了深度学习、迁移学习、强化学习等领域。传统的统计机器学习方法如决策树，朴素贝叶斯，支持向量机，集成机器学习方法 Adaboost 等，这些浅层学习方法对于结构型数据有着很好的拟合效果，但是对于非结构型数据如文本、视频等仍然达不到理想的效果。相反，深度学习算法在处理这些非结构型数据有着出人意料的效果。2012 年，多伦多大学 Geoffrey Hinton 指导的学生在 ImageNetD 大规模视觉识别竞赛中夺得图像分类[2]，目标定位两个项目的冠军，深度学习于是开始为大众所知，深度学习的燎原之火也开始掀起。LSTM、CNN 及 GAN 等深度学习算法逐渐浮出水面，并各有千秋。各种经典深度卷积网络 AlexNet、LeNet-5 及 Inception-v3 模型等在各大赛事也风靡。如今，深度学习技术能够取得重大突破的根本原因不仅仅是数据量的日益积累，还有强大的硬件 GPU、TPU 的运算支持[3]。为了赶上这股“深度学习热“浪潮，国内外大量互联网公司、科研机构及高校开始大规模的加大深度学习技术的应用和投入。机器学习开始应用于股市预测和交易策略构建，从开始的机器学习方法如决策树、循环神经网络等预测股价开始，再到后来的强化学习交易策略的诞生。一般而言，一个高效的算法对股市的预测不仅需要一个合理的网络结构还需要庞大而高质量的数据集作为支撑，由于偶尔虚假的市场信息以及有限的数据，因此目前很多算法交易策略理论和方法在实践上困难重重。

机器学习在自然语言处理（Natural Language Processing）应用上起着至关重要的作用。自然语言技术是对人类特有的书面及语音行驶的自然语言信息进行处理和加工的计算机技术[4]，目的是为了计算机能够理解动物的语言，对其处理之后做出回应，不同的场景将会有不同的应用。随着信息网络的不断发展，语音、图形、视频等的信息也日益积累，对数据的处理已经不再局限于结构化数据, 业界不仅需要处理结构型数据，还要对非结构型数据例如文本处理的技术性人才，自然语言处理技术被认为是业界最为稀缺的技能之一[4-5]。自然语言处理技术已经渐渐成熟，例如：情感分析，推荐系统，看图说话，在线答题，人机对话等都涉及到了自然语言处理的技术，互联网时代每天都有大量的微博，朋友圈以及邮件和网络评论等，企业和科研单位对于不同

类型数据信息的有效提炼可以大大增强其业务能力和产品研发水平。常见的涉及自然语言处理技术应用的还有 Siri、翻译、拼写检查器、语音识别等等。

在利用自然语言处理技术的角度对股市预测研究上，目前有通过公司年度报告进行文本挖掘有效信息，并结合企业财务数据以及分析煤炭行业股市历史交易数据，或者通过回归分析上市公司年度报告发布对股价的波动影响。这种通过文本数据挖掘有效信息分析股价的方法具有一定合理性和有效性。但是这些数据的获取上存在一定困难，大部分公司不会随意公布重要信息以及真实财务数据，加之虚假汇报信息，这也给其模型预测带来一定的噪音影响，实践性较差。

影响股价变动的包括公司业绩、盈利情况、国家政策、股票成交量等定量因素，如果能够尽可能提取涵盖这些因素的信息将很大程度提高股价趋势的预测精度，这将给投资者无比宝贵的投资决策参考。本文认为，每日新闻标题中往往涵盖这些信息。通过研究自然语言处理技术对新闻标题的情感因素和其他文本特征的提取，并设计集成算法对特征集进行实验，最终对股价达到较好的预测效果。

1.2 文献综述与创新点

目前已有相关利用传统机器学习方法、强化学习 Q-learning 模拟股价走势以及文本挖掘市场信息预测股市波动的研究。例如，Gordon Ritter[6]在《Machine Learning For Trading》阐述了强化学习在动态交易中的策略研究，表示在适当选择奖励函数的情况下，强化学习技术(比如 Q-Learning 算法)能够有效地处理风险规避的情况，其以一个模拟市场的形式提供了一个理论证明，即使有交易成本，这个市场也允许统计套利，Q-Learning 算法构建的策略能够发现并利用这种套利机会。张昊[7]在《互联网财经新闻对股市影响效应的测度》中分析了财经新闻对股市的作用效应，通过 LDA 概率主题建模提取文本的情感倾向，构建主题-情感倾向指数，并采用测度方法——事件研究法分析新闻事件对个股的影响，根据格兰杰因果分析发现了金融主题情感倾向指数与金融板块指数收益率的因果关系及影响持续周期，另外创新性地在情感分析的基础上,利用主题模型将情感分析从一个文本一个情感倾向的形式转化为一个文本多个主题情感倾向的形式，进一步提出了一种文档集情感倾向指数的构建方法。还有，朱昶胜[8]在《基于 R 语言的网络舆情对股市的影响研究》研究中，以东方财富网的股评为研究对象，利用文本挖掘技术对非结构化文本数据转化为结构化的特征向量矩阵,结合股票收益率建立 SVR 回归模型预测未来的股票收益率来预测股价的涨跌趋势；研究结果表明，可以通过分析网络舆情来预测股价未来趋势。过往的研究中，还有对股价进行时间序列分析的研究，赵国顺[9]在《基于时间序列分析的股票价格趋势预测研究》通过对股价进行 7 日均线化降噪，摒弃传统的 GARCH 模型对股价的预测，并对均线趋势序列数据采用 GARCH 模型和 ARIMA 模型建模分析，发现其有短期预测的效果，并对投资者买卖时机和风险规避有一定的指导意义。

相较传统的交易策略以及过往的股市预测研究，本文解决了数据样本数量不够、股市预测效果不佳的难题，并将预测结果构建策略进行回测，分析准确率高的算法和盈利情况不对等的情况。本文基于新闻标题的情感特征分析每日新闻对股市的冲击，利用自然语言处理技术提取新闻标题的情感信息。通过评估特征集的重要性分析文本情感对股价波动的影响；结合情感特征、文本特征和股价指标特征构建的特征集，融合多种算法构建集成模型对股市历史数据进行训练，仅在较小数据量的情况（995 个样本）取得较高的预测准确率（准确率 87%，AUC 分数 0.956）。基于构建的模型训练过去四年股价历史数据预测未来四年的股价，利用预测结果构建算法策略。模拟中国股市交易制度设计回测机制进行策略历史模拟，并分析算法交易当前的利弊，给予股市投资者有价值的投资决策参考。

1.3 研究目标与研究内容

本论文目的是解决现存股市预测方法的不足，对数据收集可获性和历史样本量小的问题，从方法的高效性和算法鲁棒性的角度来设计和构建有效的预测模型解决问题。避免现存研究方法的低效和可行性低的弊端。

研究过程中，探索文本数据特征的提取方法，研究不同特征集在 GDBT 的表现以及算法预测准确率和实际回测收益率的关系。

研究方法上，首先通过探索性数据分析，对文本进行词频分析、TF-IDF 关键词提取、LDA 主题模型以及情感分析，有效的为合理文本特征的构建提供科学的经验支持。在特征工程上采用多种自然语言处理技术构建有效合理的文本特征，最终确定的方案是，文本特征的提取主要通过 2-grams 法构建词频矩阵、情感分析和 LDA 主题模型等构建文本特征；以及对股价均线和 MACD 指标等常见策略技术指标进行属性规约和股价特征建模，保证了特征工程的科学性和有效性。在文本情感特征的提取上，设计了一套以 Embedding 层、LSTM 和 Attention 机制等网络层构建的深度网络进行情感倾向训练和预测，预测结果 AUC 分数高达 0.97，极大的提高了情感倾向的预测效果和预测稳定性。最后对基模型如决策树、贝叶斯网络、支持向量机、逻辑回归、神经网络等多种机器学习算法通过 K 折交叉验证评估出性能最优的算法，最后设计融合方案构建融合算法，将融合算法的预测结果设计成交易策略，同时设计模拟真实市场交易的回测程序。利用构建好的策略进行回测，模拟策略在真实市场的交易情况并利用回测结果分析算法交易在实盘的优劣及需要改进的地方。

2 算法理论与问题描述

本文通过对问题合理的描述和转化，针对文本数据和结构型数据进行处理和分析，选择合适的文本挖掘算法和机器学习算法；本章节主要介绍本项目使用的算法和问题结合算法的解决方案。对于文本挖掘算法这部分，主要介绍文本挖掘算法的理论知识。在使用这些算法技术实现目标时，往往需要对文本进行预处理，比如分词，去停词，词干化等步骤，这些都是自然语言处理的重要组成部分，本文结构安排上，将这些技术放在后面实证分析的时候描述会更直接明了，这里默认都已经对文本进行了预处理。

全文完整数据管道操作（pipeline）如图 2.1 所示。

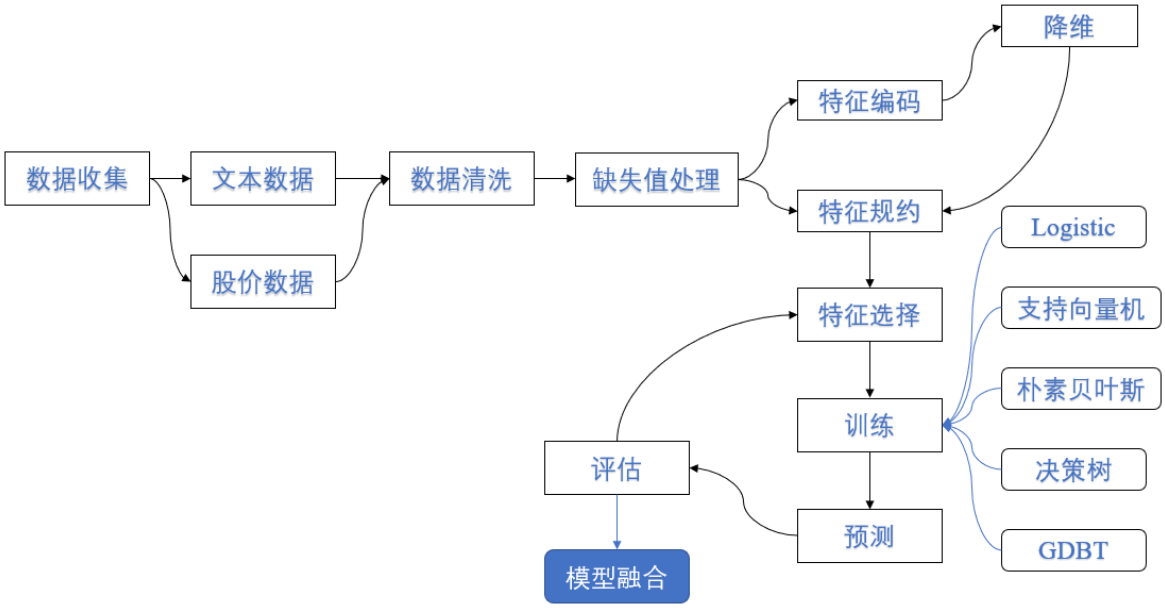

图 2.1 数据管道操作示意图

2.1 文本挖掘算法

2.1.1 TF-IDF 算法

什么是关键词，关键词就是该文章显而易见区分于其它文章的词语；任何一篇文章都有一些能体现文章意义和主旨的关键词，相反有些词就不是。如果单纯以词频确认关键词，对于无实意代词你、我、她/他由于数量较其他词语多，那么毫无疑问就是关键词，然而事实不是。因此有必要在数据清洗的前提下，比如去除停词等，来进行提取关键词的工作。

词频既不是关键词的充分条件也不是关键词的必要条件；比如在一篇介绍非洲历史的书里，“非洲”和“历史”毫无疑问是一个高频词，但是在已经知道这是一本介绍非洲历史的书的情况下，显然非洲就不应该是书里需要了解的词语；一般而言，希望得到的是书里介绍了非洲什么历史，不同非洲国家的历史，不同章节之间有什么区别性的关键词，然而这些关注点是词频定义的关键词所不能涵盖和解释的。因此这里我们

就要利用重新理解关键词的定义，然后根据 TF-IDF 算法提取。

当一个词在书里一个章节(书)提到的越多，在其他章节（书）提到的越少，那么这个词就是这个章节的关键词。这就是 TF-IDF 算法（Term Frequency-Inverse Document Frequency）的意义。由于现实中大量文本数据不包含有关键词的提前备注，因此需要从文本中提取一些有用的甚至是感兴趣的信息是不可或缺的。往往在挑书的时候会大致翻看一下内容来决定我们是否继续阅读或者购买，因此在互联网大数据海量文本时代，关键词提取算法对于快速获取信息显得尤为重要。在自然语言处理中，关键词提取技术跟机器学习方法类似，也分为有监督和无监督算法的区别。有监督方法需要人为的做大量的标注数据，给数据“贴上”标签，经过训练之后一般具有较高的准确度，目前往往是有监督技术的应用范围更广。无监督不需要提前对数据贴标签，因此实际应用更简单方便。常见的无监督关键词提取算法有 TextRank、TF-IDF 算法和主题模型算法。本文着重介绍无监督算法 TF-IDF 的关键词提取和主题模型。

词 *tf* 的计算公式如下：

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{2.1}$$

词 *idf* 的计算公式如下：

$$idf_i = log(\frac{|D|}{1+|D_i|}) \tag{2.2}$$

词 *tf-idf* 的计算公式如下：

$$tfidf_{i,j} = tf_{ij} \times idf_i \tag{2.3}$$

其中n_{ij}是词 i 在文档 j 中出现的频数，$\sum_k n_{kj}$就是文档中词的总数。D_i是词 i 在文档中出现的次数，$|D|$是文档的总数。通过公式（2.1），（2.2），（2.3）的计算，这样就是可以得到词 i 在文档 j 的 *tf-idf* 值。

2.1.2 LDA 主题建模

在一系列的文档中，主题模型在自然语言处理中主要是用来抽象主题的一种统计模型。往往判断文档之间相似性的方法是分析文档共同出现的单词的次数以及 TF-IDF 等。但是这些方法存在局限性，并不能考虑到文字背后的语义关联；比如当相似文档之间出现的共同单词极少甚至没有的时候，在判断文档相似性的时候往往需要借助主题建模的方法。另外，基于文档本身的关键词提取还不是很充分反映文档的隐式信息，比如在一片描述植物百科的书里描绘有多种植物，但是通篇未提植物二词，通过关键词提取并不能。

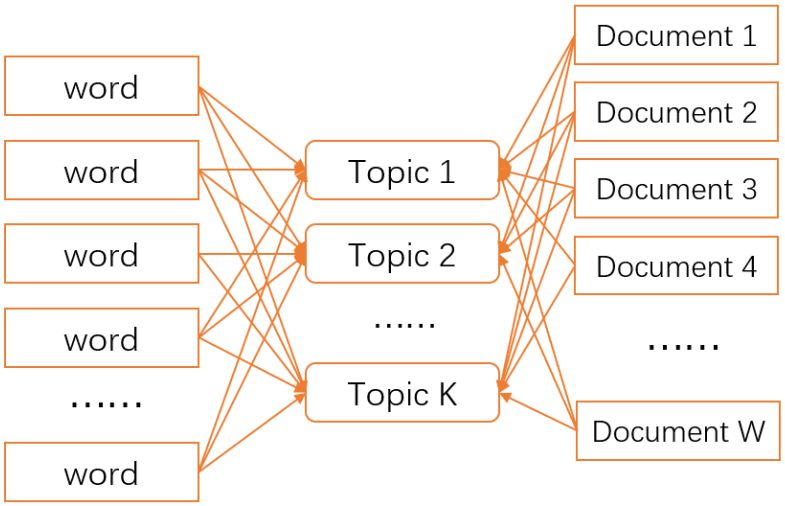

图 2.2 主题模型示意

通过主题模型（原理如图 2.2），可以提取文本中用词规律，将具有类似规律的文本联系到一起，以提取文本数据集中有用的隐含语义结构（latent semantic structure）信息。例如,文档阐述的主题中出现“国家”一词，该主题会涵盖一些对“国家”具有描述性特征的词语，借此可以分析文档中“国家”层面的一些信息。

LDA 主题模型（Latent Dirichlet Allocation）是主题模型的一种由 Blei 等人提出的一种生成式主题模型。LDA 是一种无监督的模型，只需要提供无标签的训练数据即可训练，节省人力及时间，泛化能力强，并且不容易过拟合。其假设每一篇文档中的每个词都是通过“一定的概率选择了某个主题，并从这个主题中以一定的概率选择某个词语”。LDA 也是包含文档、主题、的词三种结构一种三层贝叶斯概率模型，，能够有效地对文本进行建模。其和传统的空间向量模型（VSM）相比可以降维解决数据稀疏的问题，生成新的特征空间(主题)。其在文本聚类，相似度及主题分析都有大量应用。

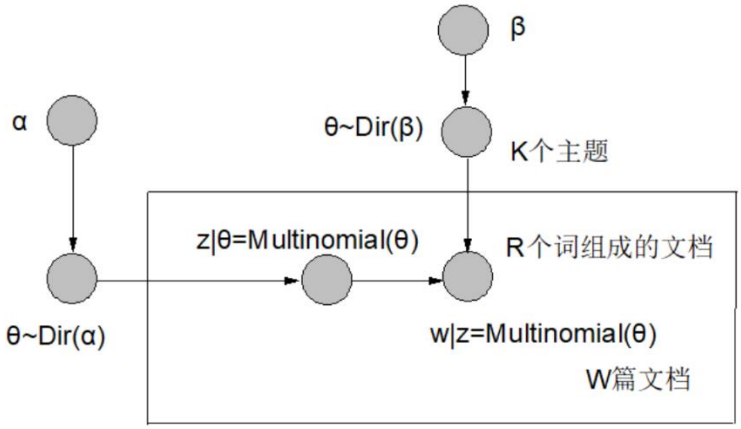

图 2.3 LDA 主题模型原理

LDA 主题模型（原理如图 2.3）采用词袋模型（Bags of words），即将每个文档是为一个词频向量；定义所有文档构建的词表大小为n，因此每个文档可以用一个n维

向量表示，通过 one-hot 编码对文档进行特征处理，例如一个n维向量(1,0,0,…,0)表示一个词。由R个词组成的文档r可以表示为$(q_1, q_2, q_3, \dots, q_R)$。假设有$W$篇文档，存在$K$个主题，记为$z_i, (i = 1,2,3, \dots, K)$。记α和$\beta$各为文档主题分布和主题词的先验分布的狄利克雷函数的先验参数，假设每篇文章由多个主题按一定比例随机构成，构成的比例服从多项分布：

$$z|\theta = Multinomial(\theta) \tag{2.4}$$

并且各个主题由文档词表中的词语按一定比例混合而成，混合比例也服从多项分布，记为：

$$w|z, \emptyset = Multinomial(\theta) \tag{2.5}$$

并且在文档r_j条件下生成词w_j属于第t个主题的概率，表示为$P(z = t|w_j)$，记为：

$$P(z = t|w_i) = \sum_{s=1}^{K} P(w_i|z = t) \times P(z = t|r_j) \tag{2.6}$$

2.1.3 关联规则

关联规则分析又称为关联挖掘分析，是一种对项目集合和对象集合的频繁模式，相关性及因果关联结果信息的提取方法。关联分析是一种极其实用简单的分析技术，可以发现存在于大数据集中的强关联性或相关性的信息，从而搜索事物中潜在的某些属性同时出现的模式规律。其在推荐系统及数据挖掘分析中都有广泛的应用。

关联分析的最终目标就是要找出强关联规则。在现实生活中，往往可以发现在商场中消费者买羽毛球拍之后买羽毛球比买乒乓球的可能性更大，或者说羽毛球拍和羽毛球一起购买的可能性更大。那么商场可以在放置羽毛球拍的旁边放置羽毛球，这样毫无疑问可以一定程度的促进消费，方便用户。这就是一个关联规则的实际应用。在文本中，通过关联分析可以一定程度有效分析文本的词频规律。

事件 X 和事件 Y 的支持度是 X 和 Y 两个事件同时出现占事件的总数，即：

$$Support = P(XY) = \frac{P(XY)}{P(\Omega)} \tag{2.7}$$

事件 X 对事件 Y 的置信度是 X 和 Y 两个事件同时出现占出现 X 的事件数，即：

$$Confidence = P(Y|X) = \frac{P(XY)}{P(X)} \tag{2.8}$$

事件 X 对事件 Y 的独立度是表示含有 X 的事件下，同时含有事件 Y 的概率，与 Y 发生的概率之比，即：

$$Lift = \frac{p(Y|X)}{P(Y)} \tag{2.9}$$

一般而言，如果$Lift(X \to Y) > 1$，则规则“$X \to Y$”是具有强关联性。如果$Lift(X \to Y) \leq 1$，则规则“$X \to Y$”是无效的关联规则。特别地，如果$Lift(X \to Y) = 1$，则表示X与Y相互独立。

2.2 统计机器学习算法

2.2.1 决策树和随机森林算法

决策树(Decision tree)[10]是一个有监督算法，其训练速度快，可解释性强，是最受欢迎的算法之一。决策树由有向边和节点组成，其原理是在特征空间执行递归二元分割。决策树有很多种，其中包括 ID3 决策树，还有 CART 决策树等，本文将用 ID3 决策树作为分类算法。ID3 决策树是基于信息熵（信息增益）来选择最优特征的一种算法。决策树可以解决分类和回归的任务，也可以解决本文的二分类任务（情感分数分评估和未来一天股价涨跌情况），假设样本空间D为此具有N个特征的样本所占比例$p_k(k = 1,2, \dots, N)$，即D的信息熵可以定义为：

$$Ent(D) = -\sum_{k=1}^{N} p_k \, log(p_k) \tag{2.10}$$

假定离散特征a有n个可能的取值(属性或者特征类别)设为$\{a^1, a^2, a^3, a^4, a^5, \dots, a^n\}$，用$a$对集合$D$进行划分，在产生的$n$分支节点，在$n$个分支节点中，第$v$个包含 D 在$a$的样本取值$a^v$，记为$D^v$，$v \in \{1,2,3,4,5, \dots, n\}$。计算信息增益：

$$Gain(D, a) = Ent(D) - \sum_{v=1}^{n} \frac{|D^v|}{|D|} Ent(D^v) \tag{2.11}$$

特征对目标分类准确性越高，信息增益越大，“纯度”就越大。因此划分特征的时候以信息增益最大的特征为特征划分的节点。

对于连续特征，假设连续特征为b，若b在样本空间D上出现n个不同值，需先将值从大到小排列，这里记为$\{b^1, b^2, b^3, b^4, b^5, \dots, b^n\}$。基于划分点$t$将 D 分为子集$D_t^+$和$D_t^-$，$D_t^+$是特征$b$在$D$上大于$t$的样本，$D_t^-$是$b$在$D$上小于$t$的样本。因此，对连续特征$b$，候选划分点集合为：

$$T_b = \left\{ \frac{b^i + b^{i+1}}{2} \middle| 1 \le i \le n-1 \right\} \tag{2.12}$$

从而，将区间$[b^i, b^{i+1})$的中位点$\frac{b^i + b^{i+1}}{2}$作为候选划分点，继而可像离散特征处理。

$$Gain(D, b, t) = \max_{t \epsilon T_b} Ent(D) - \sum_{\mu \in \{+,-\}} \frac{|D_t^\mu|}{|D|} Ent\left(D_t^\mu\right) \tag{2.13}$$

$Gain(D, b, t)$是样本集 D 基于划分点 t 二分后的信息增益。以$Gain(D, b, t)$最大的划分点划分。构建所有训练样本的根节点，根据最优特征划分子集，递归直至所有训练样本大部分被正确分类后，或者没有合适的特征为结束。

在机器学习中，随机森林是一个包含多个决策树的分类器， 一种以决策树为基学习器(又称基线模型)的 Bagging 算法[10]，并且其输出的类别是由个别树输出的类别的众数而定，并根据以下算法构建每棵树：

(1)用N来表示训练用例（样本）的个数，M表示特征数目。

(2)输入特征数目m，用于确定决策树上一个节点的决策结果；其中m应远小于 M。

(3)从N个训练用例（样本）中以有放回抽样的方式，取样N次，形成一个训练集，并用未抽到的用例（样本）作预测，评估其误差。

(4)对于每一个节点，随机选择m个特征，决策树上每个节点的决定都是基于这些特征确定的。根据这m个特征，计算其最佳的分裂方式。

(5)每棵树都会完整成长而不会剪枝，这有可能在建完一棵正常树状分类器后会被采用。

2.2.2 朴素贝叶斯网络

朴素贝叶斯分类[10]（Naive Bayes Classifier）是以贝叶斯定理为基础，通过对象的先验概率，利用贝叶斯定理计算出其后验概率，即该对象属于某一类的概率，并以最大后验概率的类作为该对象所属的类。

设$P(Y)$为先验概率，$P(Y|X)$为后验概率。D为样本空间（与决策树的类似的表示方式），其中事件U样本空间的一个划分。假设目标有 K 类，则有:

$$\mathrm{P}(U_i|\mathrm{A}) = \frac{P(A|U_i)P(U_i)}{\sum_{j=1}^{N} P(A|U_i)P(U_i)} \tag{2.14}$$

在朴素贝叶斯算法中，需要计算的参数为：先验概率$P(Y = c_k)$以及条件概率$P(X^{(j)} = x^{(j)}|Y = c_k)$。

在朴素贝叶斯法假设中，分类确定的条件下，用于分类的特征是条件独立的。即：

$$P(X = x|yY = c_k) = \prod_{j=1}^{N} P\left(X^j = x^j \middle| Y = c_k\right), k = 1,2,\dots,K \tag{2.15}$$

根据贝叶斯定理:

$$P(Y = c_k|X = x) = \frac{\prod_{j=1}^{N} P\left(X^{(j)}=x^{(j)}|Y=c_k\right)}{\sum_{j=1}^{K} P\left(X^{j}=x^{(j)}|Y=c_j\right)P\left(Y=c_j\right)}, k = 1,2,\dots,K \tag{2.16}$$

得到朴素贝叶斯分类器为:

$$y = f(x) = argmax_{c_k} \frac{P(Y=c_k)\prod_{i=1}^{N} P\left(X^{(i)}=x^{(i)}|Y=c_k\right)}{\sum_{i=1}^{K} P\left(X=x|Y=c_j\right)P\left(Y=c_j\right)}, k = 1,2,\dots,K \tag{2.17}$$

这里使用的是高斯贝叶斯分类器，它是朴素贝叶斯算法中的一种。它假设特征的条件概率分布满足高斯分布:

$$P\left(X^j|y = c_k\right) = \frac{1}{\sqrt{2\cdot\pi\sigma_k^2}} e^{\left(-\frac{\left(x^j-u_k\right)^2}{2\sigma_k^2}\right)}, k = 1,2,\dots,K \tag{2.18}$$

其中σ_k^2，u_k为某一分类中的方差和均值。

2.2.3 逻辑回归

逻辑回归（Logistic Regression）是一种解释性强，实现简单，训练速度快，易并行，以及在大规模数据情况下非常适用的一种分类和回归算法，其非常适合于数值型和离散数据，具有高偏差低方差的特点，因此非常容易欠拟合。目前广泛应用于 CTR

预估，推荐系统等。

采用逻辑回归分类思想[10-11]来解决拟二分类任务，具体内容如下：

将积极情感设为 1，消极情感设为 0，则其输出标记为$y \in \{0,1\}$,线性回归模型中产生的预测值为：

$$z = \omega^T x + b \tag{2.19}$$

z为实值，需将z转换为$0/1$值，采用可微的近似单位阶跃函数的“替代函数”——对数几率函数（sigmoid 函数，如图 2.4 所示）：

$$y = \frac{1}{1+e^{-z}} \tag{2.20}$$

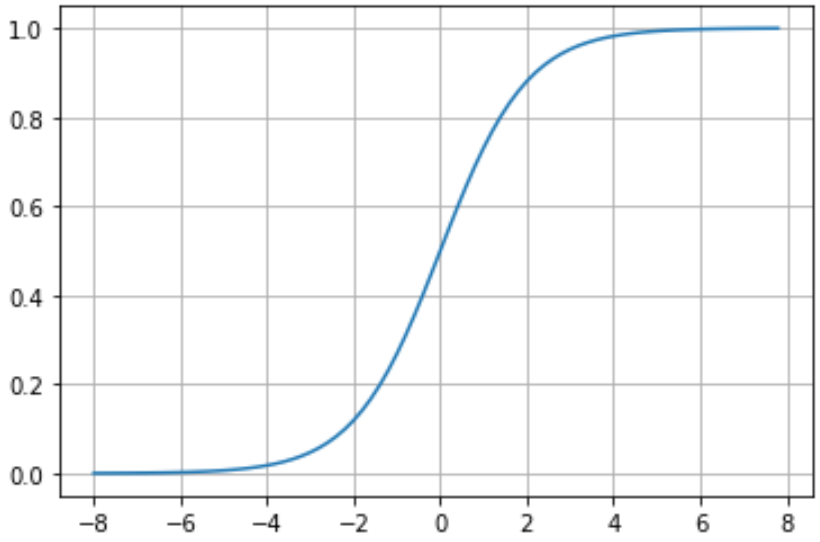

图 2.4 sigmoid 函数示意

将以上的对数几率函数带入“广义线性模型*”中，其中x是指文本经过 one-hot 编码转化后的特征：

$$y = g^{-1}(w^T x + b) \tag{2.21}$$

*注：广义线性模型指单调可微函数:$y = g^{-1}(w^T x + b)$。

可得到的公式如下：

$$y = \frac{1}{1+e^{-(w^T x+b)}} \tag{2.22}$$

此式利用线性回归模型的预测结果去逼近真实标记的对数几率，这种分类学习方法中参数w与b的值，利用后验概率估计$p(y=1|x)$，则可得到以下公式：

$$ln\frac{p(y=1|x)}{p(y=0|x)} = w^T x + b \tag{2.23}$$

之后，利用极大似然法估计求取迭代方程，估计w与b，步骤如下:

给定数据集$\{x_i, y_i\}_{i=1}^m$，利用极大似然求对概率回归模型最大化解：

$$\ell(w,b) = \sum_{i=1}^m ln\, p\,(y_i|x_i; w, b) \tag{2.24}$$

在这里令$\beta = (w,b)$，$\hat{x} = (x;1)$以及：

$$p_1(\hat{x};\beta) = p(y=1|\hat{x};\beta), p_0(\hat{x};\beta) = p(y=0|\hat{x};\beta) = 1 - p_1(\hat{x};\beta) \quad (2.25)$$

则$\ell(w,b) = \sum_{i=1}^{m} ln\, p\,(y_i|x_i;w,b)$式的似然项可以写成：

$$\ell(y_i|x_i;w,b) = y_i p_1(x_i;\beta) + (1-y_i)p_0(x_i;\beta) \quad (2.26)$$

因此，最大化式（2.24）等价于最小化式：

$$\ell(\beta) = \sum_{i=1}^{m}\left(-y_i\beta^T\hat{x}_i + ln\left(1+e^{\beta^T\hat{x}_i}\right)\right) \quad (2.27)$$

最后利用梯度下降法进行迭代求解得：

$$\beta^* = arg\min_{\beta}\ell(\beta) \quad (2.28)$$

尽管，梯度下降法的求解结果可能是局部最优的，但是一般默认为最优解，为了解决陷入局部收敛的问题，可以稍微将训练速度提高。

2.2.4 支持向量机

支持向量机[10-11]（Support Vector Machine）是定义在特征空上间隔最大的分类器。本质是一种二分类模型，采用核技巧后，可以用于非线性或者线性分类，原理见图 2.5。

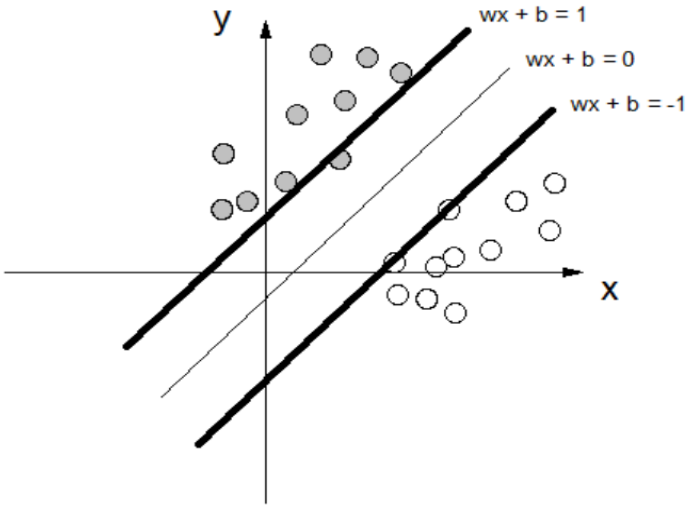

图 2.5 支持向量机原理示意

对于给定K维特征空间上的数据集$U = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_N}, y_N)\}$。其中$\overrightarrow{x_i} = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, \dots, x^{(K)})^T, y_i = \{-1,1\}$。$\overrightarrow{x_i}$是第$i$个样本数据，$y_i$为第 i 个样本的标记。

在本文中，假设带标签的样本数据集$\{x_i, y_i\}_{i=1}^{m}$，找到一个划分超平面，划分超平面的公式为：

$$0 = \vec{w}^T x + b \quad (2.29)$$

其中$\vec{w} = (w_1; w_2; \cdots; w_d)$为法向量，$b$为位移项，则空间上任意点$x$到超平面的距离为：$r = \frac{|\omega^T x + b|}{||w||}$，如图所示对于样本的分类，这里规定，即对于$(x_i, y_i) \in D$，$y \in \{0,1\}$有：

$$\begin{cases}\vec{w}^T x + b \geq 0.5, y_i = 1 \\ \vec{w}^T x + b \leq 0.5, y_i = -1\end{cases} \quad (2.30)$$

对于给定的训练数据集以及超平面$0 = \vec{\omega}^T x + b$，定义样本点与超平面的几何间隔为：$\delta_i = y_i(\frac{\vec{w}}{||\vec{w}||} \cdot \overrightarrow{x_i} + \frac{b}{||\vec{w}||})$，定义关于所有样本点到超平面的几何间隔距离最小值$\delta = min_{\overrightarrow{x_i}} \delta_i$。

之后可以将原始问题转化为带约束的最优化问题：

$$max_{\vec{w},b} \delta \quad (2.31)$$

$$s.t. \quad y_i \left(\frac{\vec{w}}{||\vec{w}||} \cdot \overrightarrow{x_i} + \frac{b}{||\vec{w}||}\right) \geq \delta, i = 1,2, \dots, N \quad (2.32)$$

支持向量机通常将一个数据样本点距离超平面的远近来表示分类预测的可靠程度。一般来说，一个样本距离超平面越远，则该样本的分类越可信；一个样本距离超平面越近，则分类越不可信。

2.2.5 GDBT 算法

GDBT 算法全称（Gradient Boosting Decison Tree），是 Boosting 集成算法[10]的一种， GBT（Gradient Boosting Tree)、GTB（Gradient Tree Boosting）、GBRT（Gradient Boosting Regression Tree）、MART（Multiple Additive Regression Tree）本质都是指的同一类算法，本文简称该算法为 GDBT。GDBT 可以处理回归和分类问题，在本文，主要介绍 GDBT 的二分类算法。

与 Boosting 另一种集成算法 Adaboost 的不同，GDBT 不是根据前一轮迭代弱学习器的误差率来更新训练集的权重，GDBT 算法（多个决策树的集成）的集成思想实际是由每一棵回归树以上一轮回归树的残差作为预测值，从而达到相对于一个树更好的效果，并且利用了前向分布的算法。对于 GDBT 来说，其弱学习器只限定了 CART 回归树模型，同时迭代方法与 Adaboost 有所不同。

在 GBDT 算法的迭代中，假设前一轮迭代得到强学习器是$f_{t-1}(x)$, 损失函数是$L(y, f_{t-1}(x))$, 算法本轮迭代的目标是找到一个 CART 回归树模型的弱学习器$o_t(x)$，让本轮的损失函数$L(y, f_t(x) = L(y, f_{t-1}(x) + o_t(x))$最小。

对于损失函数的拟合方法，GDBT 采用损失函数的负梯度来拟合本轮损失的近似值，假设第t轮的第i个样本的损失函数负梯度记为：

$$u_{ti} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)} \quad (2.33)$$

利用$(x_i, r_{ij})(i = 1,2,3,..,m)$拟合 *CART* 回归树，得到第$t$棵回归树，对应的叶节点区域$u_{tj}, j = 1,2,3, \dots, J$。其中$J$为叶子节点数目。

针对叶子节点里的样本，求出损失函数的最小，也就是拟合叶子节点最优的输出值c_j

$$c_{tj} = arg\ min_c \sum_{x_i \in R_{ij}} L(y_i, f_{t-1}(x_i + c)) \quad (2.34)$$

因此本轮决策树拟合函数记为：

$$h_t(x) = \sum_{j=1}^{J} c_{ij} I(x \in R_{tj}) \quad (2.35)$$

本轮最终的集成模型为：

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} c_{tj} I(x \epsilon R_{tj}) \quad (2.36)$$

通过损失函数的负梯度拟合，从而找到了一种通用的拟合损失函数的误差方法。在本文中，项目描述的是二分类的问题，下面介绍二分类 GDBT 算法的里理论。

其中，损失函数记为：

$$L(y, f(x)) = log\ (1 + exp\ (-yf(x))) \quad (2.37)$$

其中$y\epsilon\{-1, +1\}$，负梯度误差记为：

$$r_{ti} = -\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)} = \frac{y_i}{1+\exp\ (y_i f(x_i))} \quad (2.38)$$

对于生成的决策树各叶子节点的最佳负梯度拟合值为：

$$c_{tj} = arg\ min_c \sum_{x_i \epsilon R_{tj}} log\ (1 + exp\ (-y_i(f_{t-1}(x_i) + c))) \quad (2.39)$$

2.3 深度学习算法理论概述

接下来介绍本文情感倾向分数提取使用到的神经网络层。主要是 Embedding 层、双向 LSTM 层、Attention 层、全连接 Dense 层、Spatial dropout 层、Global Max pooling 层和 Global Average Pooling 层。

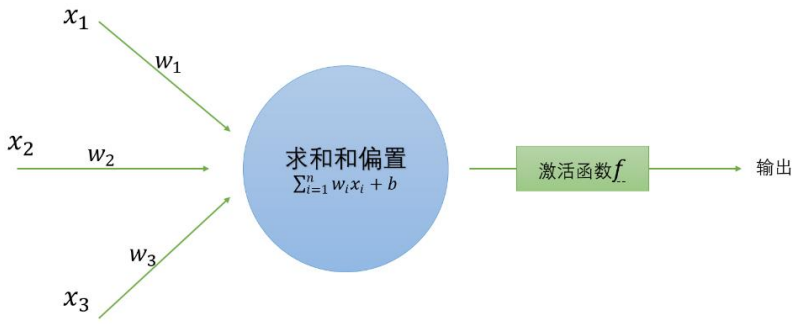

图 2.6 神经元示意

假设输入为$x = (x_1, x_2, x_3, \ldots, x_n)$,权重为$w = (w_1, w_2, \ldots, w_n)$。假设$f$为激活函数，那么对于每一个神经元(如图 2.6 所示)的输出y就是：

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \quad (2.40)$$

本文项目采用$relu$激活函数，如下：

$$f(x) = max\ (0, x) \quad (2.41)$$

另外为了对训练模型使得准确率提高，损失函数分合理选择也尤为重要，这里采用$binary\ crossentropy$的损失函数，这是个二分类问题的常见损失函数，也成为$logloss$损失函数。

对数损失函数通过惩罚错误的分类，达到分类器准确率的度量，要最大化分类准

确率就是要最小化损失函数。对数损失函数的实现公式如下：

$$L(Y,P(Y|X) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}log(p_{ij})) \quad (2.42)$$

其中Y为实际目标值，X为输入变量，$L(Y,P(Y|X)$为损失函数，N,M分别对应样本容量和可能的类别数目。p_{ij}是输入变量x_i属于类别 j 的概率；y_{ij}对应的预测是否正确，若是正确，则为 1，反之为 0。

虽然有时容易损失函数迭代收敛至局部最优，为了让损失函数达到更快，更稳定的收敛至最小值，神经网络优化器采用 RMSProp 优化器。RMSProp算法的全称叫 Root Mean Square Prop，一定程度解决了优化中经过更新之后参数的变化范围过大的问题。

One-hot 编码对应的是每份文本中存在的字用 1 来代替，不存在的用 0 来代替。这里选用了所有文本中频数前 5000 的词进行 One-hot 编码转换。然而单纯 One-hot 编码转换会造成冗余且过于稀疏的特征集，并且单词之间的关系没有得到体现[5]，为了解决该问题，于是这里采用基于 CBOW 方法的 Word2vec 模型，实现该模型的神经网络层是 Embedding 层，因此这里在整体网络结构中嵌入 Embedding 层处理上述遗留问题，原理如图 2.7 所示。

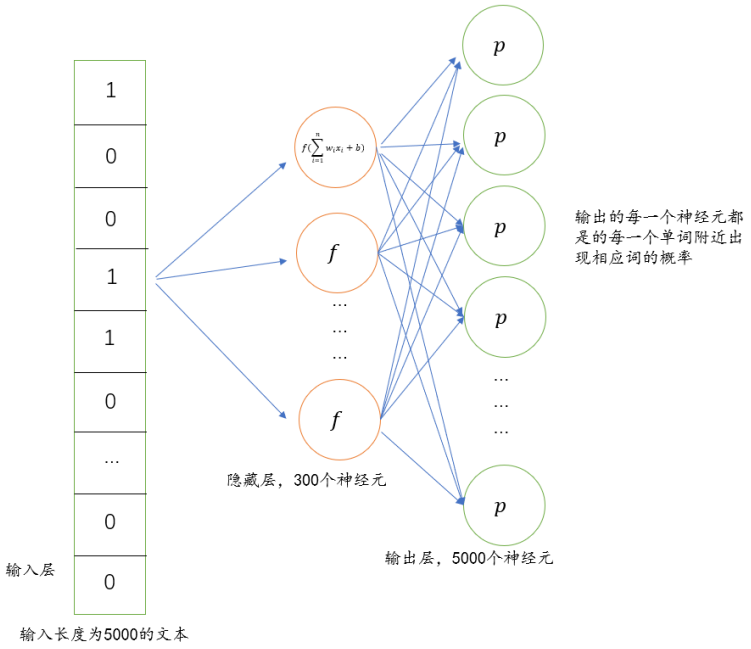

图 2.7 Embedding 原理示意

循环网络层结构(见图 2.8)：

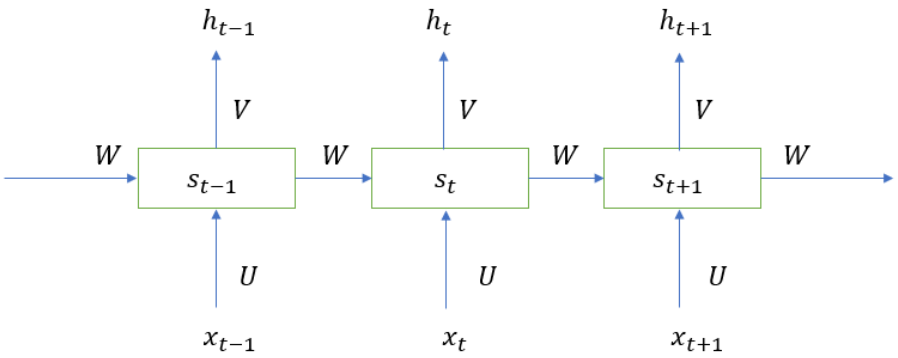

图 2.8 循环神经网络层示意

其中：

一个输入对应的是$X = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \ldots, \mathrm{x}_{t-1}, \mathrm{x}_{t,}, \mathrm{x}_{t+1}, \ldots, \mathrm{x}_N)$。

s_t为 t 时刻的隐层状态的神经元，亦称为网络记忆单元；一个隐层状态中包含多个神经元，记为$S = (s_1, \mathrm{s}_2, \ldots, \mathrm{s}_{t-1}, \mathrm{s}_t, \mathrm{s}_{t+1}, \ldots, \mathrm{s}_T)$。

h_t为 RNN 网络层的输出，记为$H = (h_1, h_2, \ldots, h_{t-1}, h_t, h_{t+1}, \ldots, h_T)$。

U为输出序列信息X的权重参数矩阵。W为S的权重参数矩阵。V为H的权重参数矩阵。

记函数f为隐层状态的激活函数，o为输出的激活函数，则当前隐层状态s_t的计算公式为：

$$s_t = f(Ws_{t-1}, Ux_t) \tag{2.43}$$

$$h_t = o(Vs_t) \tag{2.44}$$

与传统的全连接神经网络层相比，RNN 其网络数据传递方式发生了改变，可以很好的考虑了特征的时间序列性，针对股价预测和文本翻译的序列数据有很好的拟合效果。在相当长的序列中，RNN 理论上可以处理这种长期依赖的关系，然而实践证明 RNN 处理这种关系效果非常不好，具体可以参考 Y. Bengio[12]的论文《Learning long-term dependencies with gradient descent is difficult》。考虑到 RNN 可能会随着序列的增长，可能发生梯度消失的问题，导致较前的信息可能有所缺失，解决这个问题可以采用 RNN 的变种双向 RNN[13]或者双向 LSTM 层，这是对采用双向 LSTM。

如图 2.9 所示是循环神经网络 LSTM 的神经元 LSTM cell[14]的一个图例，LSTM cell 中有三个门，分别为遗忘门、输入门、输出门。其中遗忘门从 cell 中决定丢弃什么信息，输入门决定会有多少信息会进入到 cell 中来，输出门基于细胞的状态来决定输出。

在遗忘门中：

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.45}$$

在输入门中：

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.46}$$

$$\widetilde{C_t} = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.47}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t \quad (2.48)$$

在输出门中：

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.49)$$

$$h_t = o_t \tanh(C_t) \quad (2.50)$$

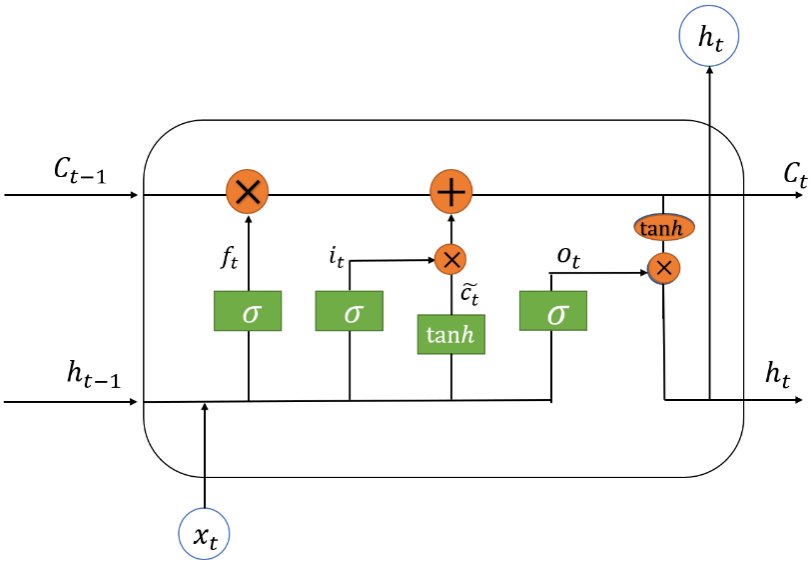

图 2.9 LSTM cell 示意

每个权重和输出以及隐状态的关系如下：

$$h_t = f(w_1 x_t + w_1 h_{t-1}) \quad (2.51)$$

$$h_t' = f(w_3 x_t + w_5 h_{t+1}') \quad (2.52)$$

$$o_t' = f(w_4 h_t + w_6 h_t') \quad (2.53)$$

双向 LSTM 的原理如图 2.10 所示。

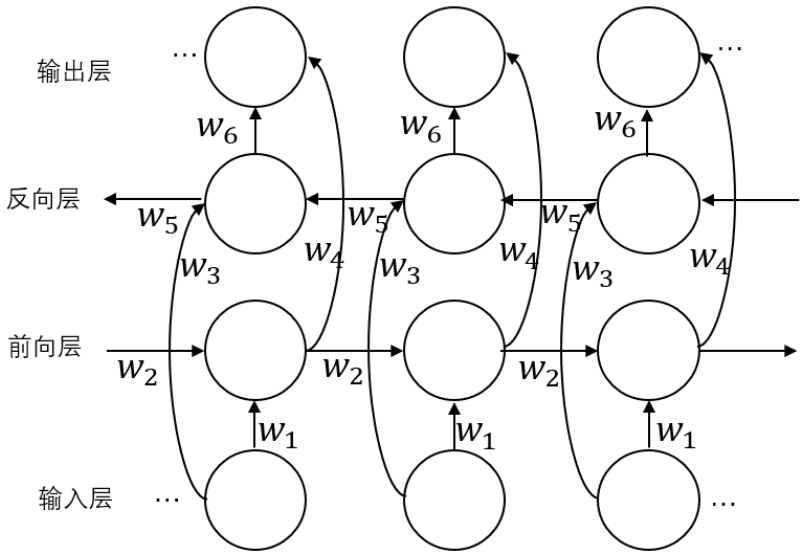

图 2.10 双向 LSTM 示意

另外采用两层全连接层产生最后预测，图 2.11 是全连接神经网络的示意。

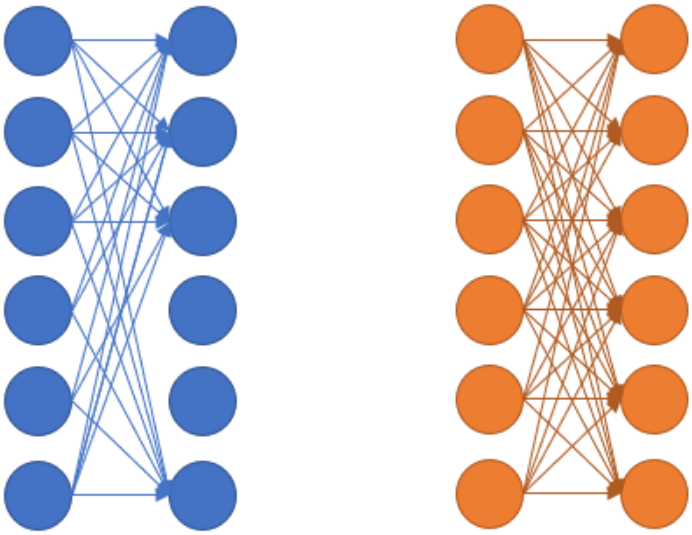

图 2.11 非全连接层和全连接层示意

本项目为了防止过拟合，引入池化层和 Dropout 层。图 2.12 是池化层（Max Pooling 层和 Average Pooling 层）的大致示意，原理是将信息进行降维，从而防止过拟合，这里主要使用的两种池化层包括 Global Max pooling 和 Global Average Pooling，其中，网络层中的 Dropout 层[15]是 Spatial dropout 层。

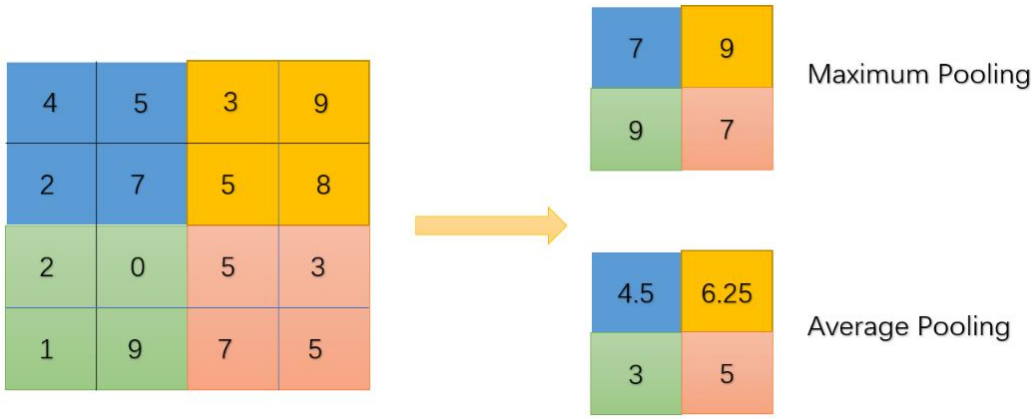

图 2.12 Pooling 层示意

在应用于问答系统的深度神经网络架构中采取注意力机制（Attention Mechanism）往往可以取得较好效果[16-17]，注意力机制通过对输入向量（例如文本的 one-hot 编码输入或者上一层的输出，本文是双向神经网络层的输出）赋予不同信息不同的权重，越是重要的信息赋予的权重越大，从而达到提取关键信息的效果。Bahdanau 等研究学者首先提出将注意力机制应用于文本翻译中，后来学者们相继提出一些改进模型，例如 Self-Attention 和 Key-Value Attention 模型等。本文利用基于自注意力机制（Self-Attention）的 B-iLSTM 网络（双向 LSTM）构建提取文本情感倾向的模型（Attention-Based Bi-LSTM）。图 2.13 是注意力机制的示意图。

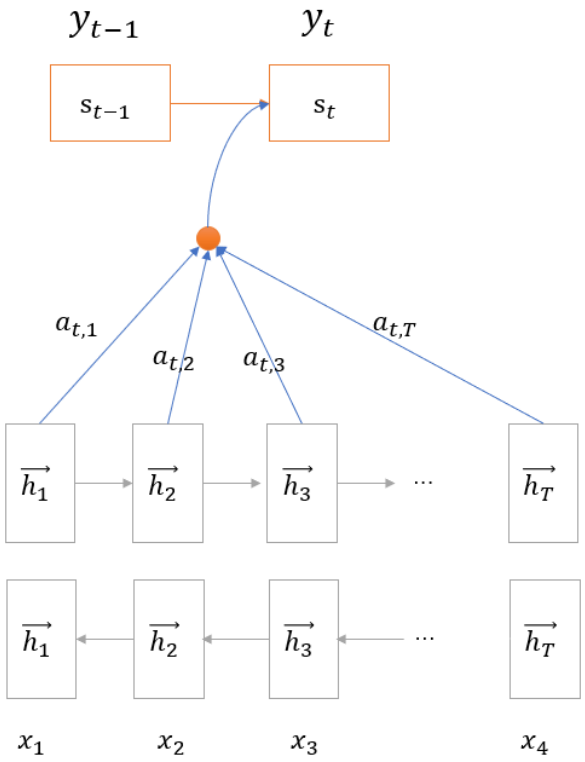

图 2.13 Attention 机制示意

其中h_j是指第 j 个输入在 encoder（在本文中，LSTM 层作为 encoder）里的输出，e_{tj}有：

$$e_{tj} = g(S_{t-1}, h_j) = V \cdot \tanh(W \cdot h_j + U \cdot s_{t-1} + b) \quad (2.54)$$

函数g可以用一个小型的神经网络来逼近，它用来计算S_{t-1}和h_j的关系分数，分数越大说明关注度越高，注意力分布就会更集中在这个输入上，该函数也称之为校准模型(alignment model)。

a_{tj}是一个权重，a_{tj}原理和 *softmax* 函数类似，得到一个条件概率$P(a|e)$。

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})} \quad (2.55)$$

对于 decoder 部分，S_t是指 *decoder*(在本文中，Dense 层作为 decoder)在t时刻的状态输出，S_{t-1}是指 decoder 在$t-1$时刻的标签，函数f是一个 RNN，有：

$$S_t = f(S_{t-1}, y_{t-1}, c_t) \quad (2.56)$$

其中$c_t = \sum_{j=1}^{T_x} a_{tj} h_j$

2.4 问题与算法解决方案

通过前面介绍单一算法的理论知识，本章节主要描述对问题的算法应用。本文的算法应用框架如图（图 2.14）。①针对非结构数据例如文本，经过 One-hot 编码、词频编码，TF-IDF 等文本向量化，LDA 和 PCA 降维后，提取为文本特征；另外对 Jigsaw 数据经由双向 LSTM 和 Attention 机制构建的神经网络训练，训练好的模型对新闻标

题进行情感预测，将预测结果与 Being 词配对产生的积极和消极情感作为情感特征。②针对股价数据，为了更好的反应未来股价波动的信息，进行技术策略指标衍生；另外对产生的特征进行统计指标(平均值、标准差、总和、最值和百分位数等)衍生。这样将非结构型数据产生的有效而显著的特征转化为结构型数据，从而更适合算法的拟合。另外对算法比如决策树、支持向量机、朴素贝叶斯、随机森林和 GBDT 等构建一个融合算法方案，AUC 分数高达 95.6%。③利用算法的预测结果设计算法交易策略，并设计回测机制，完整模拟真实市场交易过程，结合资金曲线进行结果分析。

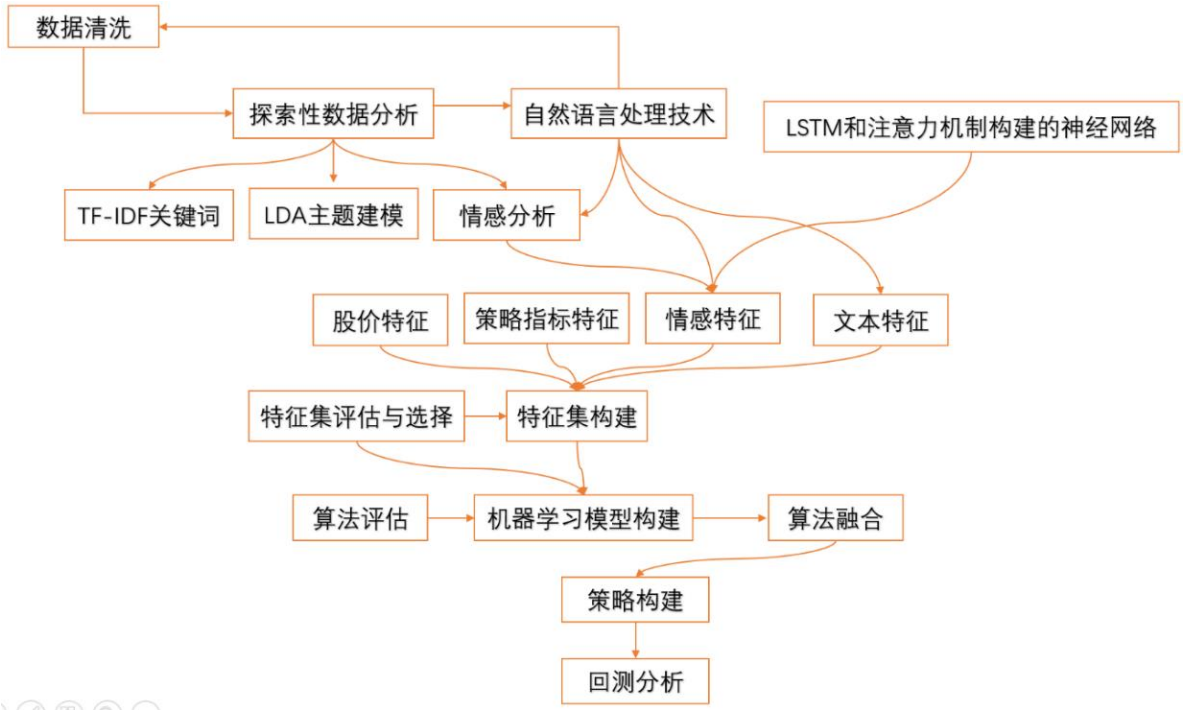

图 2.14 研究框架与技术实现

3 算法策略构建与实证分析

3.1 基于自然语言处理技术的文本预处理

3.1.1 数据背景介绍和数据标注

本文选用 Kaggle 收集的开源数据，分析 2008 年到 2016 年的道琼斯工业平均指数数据(Dow Jones Industrial Average)和 Reddit WorldNews Channel 这八年的热门新闻。研究内容是采用自然语言处理技术分析和挖掘文本特征，通过探索性数据分析，分析文本情感及其相关性和主题关系，有效地对文本特征进行建模和集成算法设计。

本文主要研究两份文本数据和一份股价数据，其中一份文本数据来自 Reddit WorldNews Channel。Reddit WorldNews Channel 是根据 Reddit 用户投票排名得出的每日热门前 25 的新闻标题，表 1 是热门度第一的部分新闻标题（每一行代表的单位是一日），Date 表示的是新闻主题的发布筑起，表 1 的是热度第一的新闻标题。另一份文本数据集是由 Jigsaw 和 Conversation AI 在 Kaggle 平台提供的数据，来自于 Civil Comments 对数据的开源以及加州伯克利大学对其标注。

表 3.1 Reddit WorldNews Channel 热度第一的部分新闻标题

DATE	TOP1
2008/8/8	B'Georgia 'downs two Russian warplanes' as countries move to brink of war''
2008/8/11	b'Why wont America and Nato help us? If they wont help us now, why did we help them in Iraq?'
2008/8/12	b'Remember that adorable 9-year-old who sang at the opening ceremonies? That was fake, too.'
2008/8/13	b' U.S. refuses Israel weapons to attack Iran: report'
2008/8/14	b'All the experts admit that we should legalise drugs '
2008/8/15	b"Mom of missing gay man: Too bad he's not a 21-year-old cheerleader, then they'd still be looking for him"
2008/8/18	b'In an Afghan prison, the majority of female prisoners are serving 20-year sentences for being victims of rape '
2008/8/19	b"Man arrested and locked up for five hours after taking photo of police van ignoring 'no entry' sign"

本文选用的股价数据是道琼斯指数，该指数是由道琼斯公司创始人查尔斯·亨利·道（Charles Henry Dow 1851-1902 年）设计的一种算术平均股价指数，也是世界上历史最悠久，普及最广的股票指数。该指数一般是由四种股价平均指数构成，本文研究的是第一种，即道琼斯工业平均指数（Dow Jones Industrial Average）。本文选用的数据时间周期是 2008 年至 2016 年。数据包含最高价，最低价，成交量，收盘价，开盘价。

另外本文对数据进行标注，下一天股价上升标注为 0，下降标注为 1.。所有数据观测得标注为 0，1 的次数大致相等。

3.1.2 文本数据清洗和分词

图 3.1 文本预处理流程概况

图 3.1 是本文的文本预处理过程，文本数据中存在大量的无用词（比如停词和语助词等），以及无意义的符号（数据中的“b""就是），本文在提取信息和分词前做了一些预处理。首先是对文本进行正则表达式匹配，去除原始文本中的所有数字和一些无用的符号。经过清洗后的数据我们就可以进行分词了。中文分词和英文分词略有不同，英文可以根据空格进行分词，然而中文不可以。中午分词的难度要远大于英文，英文的分词主要根据空格即可，但是中文需要结合语义。同样的一句话比如：中国会将来必然走向长远，我们可以将其断句为：“中国/会/将来必然走向长远”或者“中国会/将来必然走向长远”，这样将会两种截然不同的意思，因此中文分词由于一些歧义导致分词会比英文复杂的多。本文分析的是英文文本数据，所以就不过多讨论中文分词。文本中具体一句话在上述的过程的转变过程如图 3.2。

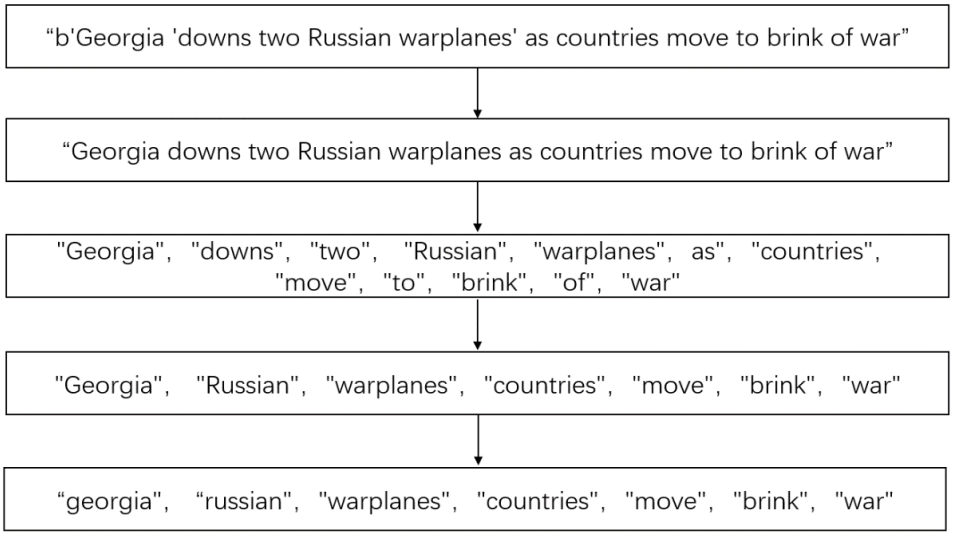

图 3.2 文本预处理过程

之后对分完词以后的数据需要进一步处理。由于在句中存在大量语气助词以及介词等停用词，这些都不具备很多有价值的信息量而且出现频次极高，会对有用的信息产生噪音干扰。因此需要将其进行清除。进一步加工，上面这句话将会变为："Georgia"，"Russian"，"warplanes"，"countries"，"move"，"brink"，"war"。但为了保证词与词之

间的统一，需要将大小写全部统一为小写，这样为后面的 one-hot 以及 TF-IDF 编码工作做准备。

最后将分词后的数据转化为整洁文本格式，符合整洁数据结构：（1）每个变量是一列（2）每次记录为一行（3）每次的记录结果就由一张表展示。因此也可以定义为：（1）每行一个词条（2）每个词条所属的句子都有所标注（3）所有的句子绘制成一张表。最后处理之后的格式（仅展示两句话）如表 2 所示：

表 3.2 整洁文本格式示意

每个词所属的文本段落	词
1	police
1	government
2	law
2	peace
….	….

进过整洁文本处理后，就可以进行探索性数据分析了，但是如果是为了作为后面的情感倾向分数提炼，我们只需进行到词干化就可以直接文本向量化了，进而得到的是一个计算机能够处理的结构化数据。

3.2 基于词频的新闻标题关注点提取

接下来对热度第一的新闻标题的词进行频数统计，一个句子如果出现两个以上的相同的词，这里都会当作一个词来处理。将词频由大到小排序，提取词频数超过 20 的词，进行词云展示，结果如图 3.3 所示，可见，在热度第一的新闻标题里，最常出现的四个词是“world（世界）”、“people（人们）”、“police（警察）”、“government（政府）”、“law(法律)”和“british（英国人）”。从词云（图 17）中显示的词可以一定程度看出新闻里受大部分人们关注的是国家和政治话题

图 3.3 Top1 词云

通过统计热度排名前四的热度词以及词频数较大的词(如表 3.3)，可以看初热度之间新闻的内容主题大致相近，基本都是世界、人民、警察、政府、战争、北部、俄罗斯等政治主题的话题。另外，这四个主题的词除了词频数细微的差别，高频词范围大致相近，差异甚微。通过词频可以侧面看出新闻热衷的报道的主题范围。

表 3.3 热度前四的新闻标题词频统计

top1	count1	top2	count2	top3	count3	top4	count4
world	107	police	102	police	94	people	90
government	99	people	97	world	90	police	82
people	90	government	91	government	86	world	77
police	84	israel	83	people	74	israel	76
war	60	world	67	israel	72	government	73
internet	59	war	66	north	67	u.s	67
law	57	u.s	60	uk	65	north	66
wikileaks	57	israeli	56	china	60	war	64
drug	55	russia	56	court	60	china	61
israel	55	uk	56	u.s	59	israeli	60
million	55	china	55	israeli	58	uk	53
uk	55	law	55	war	57	found	52
china	54	court	52	korea	56	korea	52
north	51	north	52	russian	55	russia	52
found	47	country	48	president	53	children	50
canada	46	killed	47	minister	49	killed	49
british	45	korea	46	found	45	million	48
saudi	44	human	45	internet	43	president	48
chinese	43	million	45	children	42	amp	45
court	43	ban	44	iran	42	rights	45

接下来，将词频数转化词频率，也就是这个词在所有句子里出现的频率。有时一个主题的词在另一个主题中可能频率不一样，如果两文本相差很近，那么这种词与词之间的关系散点图应该是趋于一条支线的。通过可视化对比热度 TOP1 和 TOP2 与 TOP3 文本的相关性，可见，TOP1 和 TOP2 与 TOP3 的词频关系大部分集中在一条直线上，因此三者的相似性较高的。但是 TOP1 和 TOP3 比 TOP1 和 TOP2 的关系更为分散，因此 TOP1 更近似于 TOP2。如图 3.4。

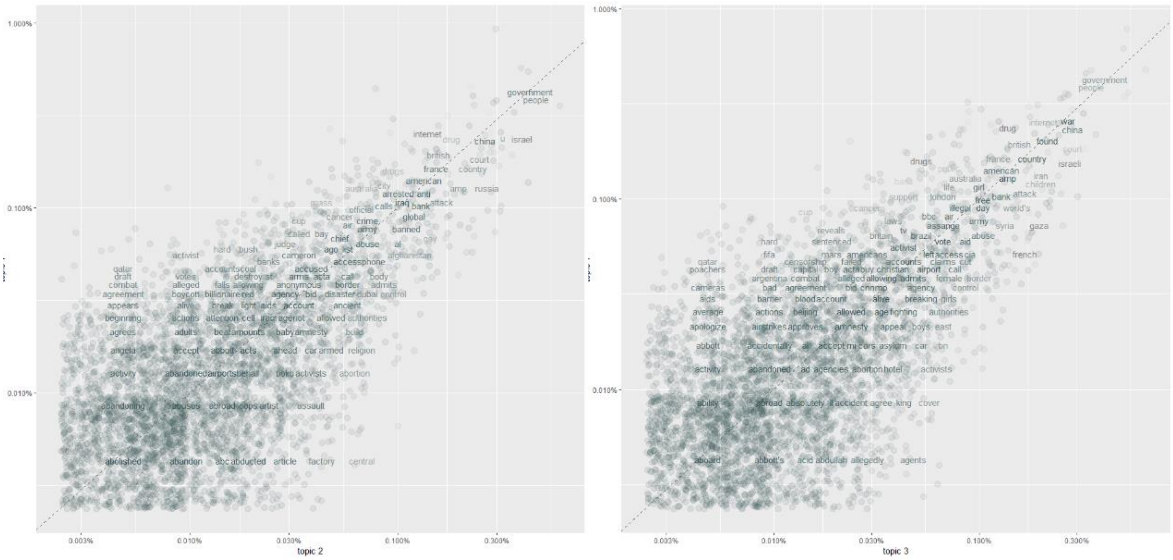

图 3.4 Top1、Top2 和 Top3 的词频率散点关系

3.3 新闻标题热度关键词分析：TF-IDF 关键词提取技术

通过计算 25 个热度排名的 TF-IDF 值得到如下表，表 3.4 是 TF-IDF 前 30 个词，可以看出这些词涉及到不同领域，并且在 25 个热度排名中较为均匀，没有重复。因此可以一定反映出不同热度的文章具有一定的区分性。进而也可以一定程度反应不同热度的主题，虽然效果不是很理想。

表 3.4 计算的 TF-IDF 值排序

topic	word	n	tf	idf	tf_idf
topic 25	mau	4	0.000207	3.218876	0.000665
topic 21	alawite	6	0.000305	1.832581	0.000558
topic 15	psy	4	0.000202	2.525729	0.000509
topic 2	childcare	5	0.00024	2.120264	0.000508
topic 22	phaseout	3	0.000155	3.218876	0.000499
topic 18	cypriots	3	0.000154	3.218876	0.000495
topic 18	flamenco	3	0.000154	3.218876	0.000495
topic 16	edwin	3	0.000153	3.218876	0.000491
topic 17	bylaw	3	0.000152	3.218876	0.000489
topic 9	ibrahim	4	0.000192	2.525729	0.000485

topic 19	nordbank	3	0.000149	3.218876	0.000481
topic 14	falciani	3	0.000148	3.218876	0.000478
topic 12	jelly	3	0.000148	3.218876	0.000477
topic 6	morphine	3	0.000146	3.218876	0.000471
topic 4	chertoff	3	0.000145	3.218876	0.000466
topic 7	yafo	3	0.000144	3.218876	0.000465
topic 9	mausoleum	3	0.000144	3.218876	0.000463
topic 3	nominate	3	0.000143	3.218876	0.000461
topic 1	isk	3	0.000141	3.218876	0.000453
topic 18	lungs	4	0.000205	2.120264	0.000435
topic 14	butter	4	0.000198	2.120264	0.000419
topic 8	plasma	4	0.000194	2.120264	0.000412
topic 2	portugal's	4	0.000192	2.120264	0.000407
topic 11	sand	8	0.000391	1.021651	0.000399
topic 24	casualty	3	0.000157	2.525729	0.000396
topic 24	vanuatu	3	0.000157	2.525729	0.000396
topic 25	borno	3	0.000155	2.525729	0.000391
topic 25	dioxin	3	0.000155	2.525729	0.000391
topic 25	kph	3	0.000155	2.525729	0.000391
topic 25	slugs	3	0.000155	2.525729	0.000391

可视化热度前 9 的 *tf-idf 值*较高的词，如图 3.5 所示。通过区分不同热度的关键词，可以大致推测出人们对不同主题的喜爱程度；比如，热度排名第一（图 3.5 的前 10 热度的关键词）的关键词有 ops(运营)、isk(风险)、undemocratic(不民主)、unsold(未售出)和 venture(风险)等，热度排名第二的关键词有 albino(白化病)、childcare(儿童保育)、loves(爱)、owes(欠)、parades(游行)和 yakuza(黑帮)等，因此可以明白公众对政治和商业主题的话题较健康类的话题更感兴趣，当然健康类的话题也是人们的第二关注点。

图 3.5 前 10 热度的关键词

3.4 文本主题分类:LDA 主题模型

为了确认最优主题数 K，本文采用基于相似度的自适应最优 LDA 模型选择方法以确定主题数进行主题分析。该方法可以在较少的迭代找到最优的主题结构，如图 3.6 所示。

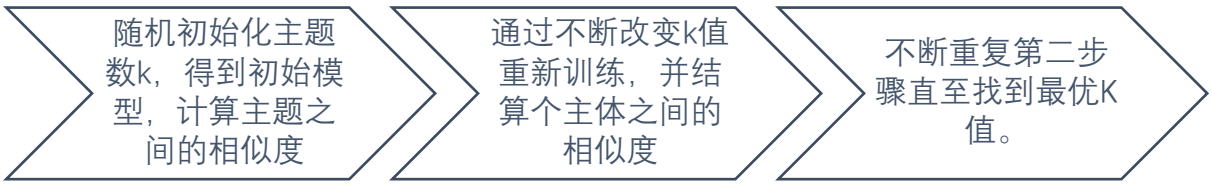

图 3.6 LDA 主题模型主题确认方法

假设 V_1 和 V_2 是两个 n 维向量，$V1 = (v_{11}, v_{12}, v_{13}, \dots, v_{1n})$，$V2 = (v_{21}, v_{22}, v_{23}, \dots, v_{2n})$，则$V_1$和$V_2$的夹角$\theta$的余弦值通过一下计算：

$$cos\theta = \frac{V_1 V_2}{|V_1 V_2|} = \frac{\sum_{i=1}^{n} v_{1i} v_{2i}}{\sum_{i=1}^{n} (v_{1i})^2 \sum_{i=1}^{n} (v_{2i})^2} \quad (3.1)$$

计算结果如图 3.7：

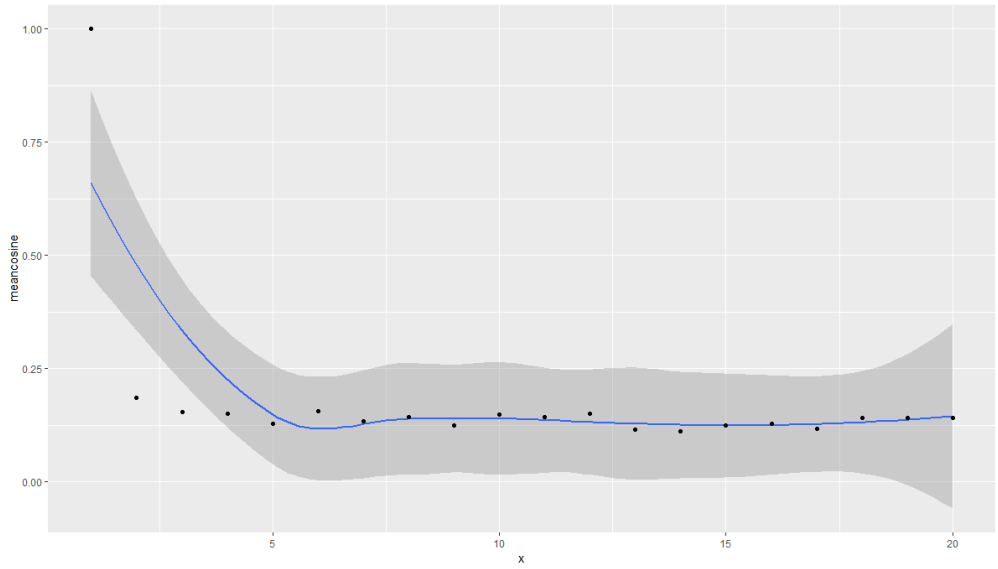

图 3.7　不同主题之间的相似度度量

将文本所有数据进行迭代，根据图 3.7，从主题数 K=5 开始，余弦相似度下降到最低端，之后基本保持稳定。因此最优主题数我们可以定义为 6。接下来对确认好的主题进行统计,罗列主题里包含的概率前 15 的词语，情况如表 3.5 所示：

表 3.5　不同主题的词语

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
police	war	government	people	israel	u.s
killed	gaza	court	china	nuclear	president
world	death	anti	amp	russia	russian
al	bank	million	korea	uk	iran
military	syria	international	time	law	minister
official	european	attack	dead	british	news
muslim	calls	human	drug	israeli	war
report	leader	found	protest	north	internet
climate	palestinian	united	military	chinese	global
kill	day	army	top	water	children
ukraine	protests	iraq	sex	free	security
wikileaks	troops	protesters	officials	air	germany
media	forces	ban	country	saudi	south
canada	europe	video	found	pay	workers
city	days	billion	isis	oil	secret

3.5 新闻隐含信息分析：关联规则

为了分析词语之间的出现联系，可以采用 n_grams 法[5]，这里采用 2-grams 法，2-grams 法是将一句话以词进行两两连续切分，例如“今年冬天会很冷”这句话经过 2-grams 法处理后，得到的二元词即“今年”、“年冬”、“冬天”、“天会”、“会很”、“很冷”。然后提取所有新闻标题统计频数大于 35 的二元词，将二元词之间有公共词语的

都连接起来。然后可视化其关系，如图 3.8 所示，即可得到词语之间存在的连续关系和相关性。

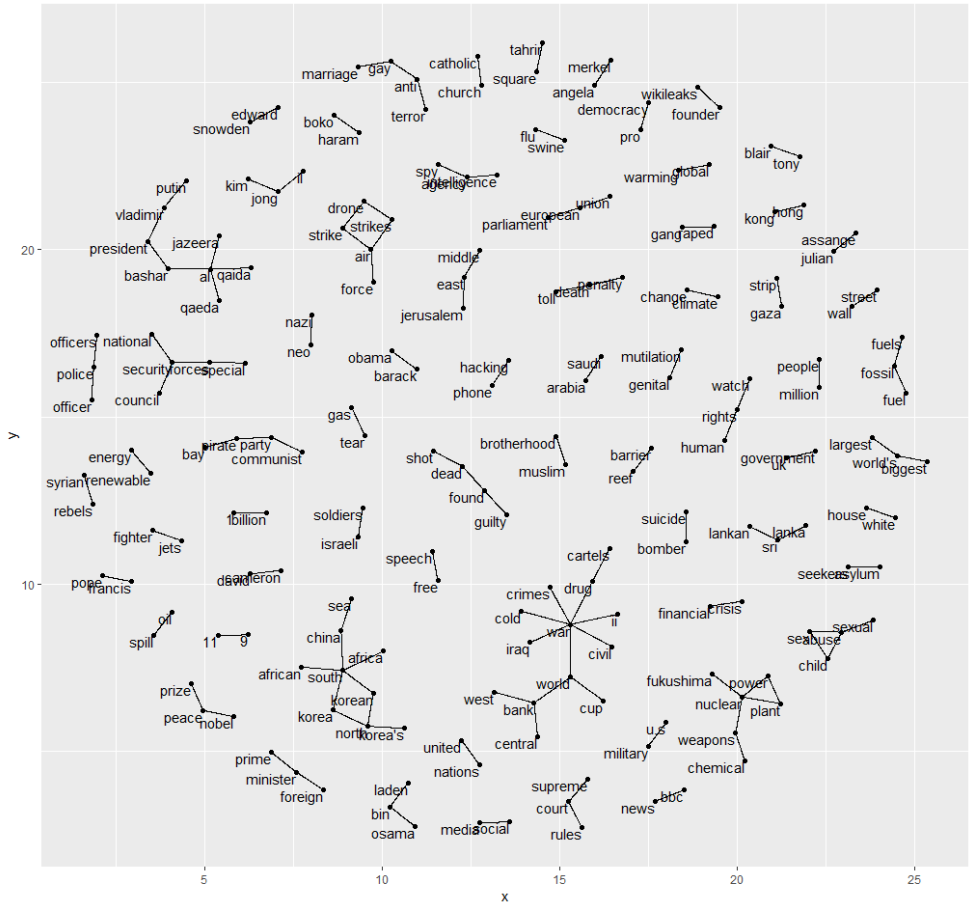

图 3.8 词语关联

大致分析文本的内容主题。本文通过对所有文本进行分词、去停词、词干化和整洁文本转化后，对分词进行 Apriori 算法的关联规则挖掘，结果如表 3.6。

表 3.6 关联规则

lhs		rhs	support	confidence	lift	count
{prime,world}	=>	{minister}	0.107088989	0.938325991	2.606607	213
{government,prime}	=>	{minister}	0.12066365	0.9375	2.604312	240
{prime}	=>	{minister}	0.199095023	0.925233645	2.570237	396
{police,prime}	=>	{minister}	0.102061337	0.918552036	2.551676	203
{people,prime}	=>	{minister}	0.106586224	0.913793103	2.538456	212
{korean}	=>	{north}	0.106083459	0.712837838	2.367002	211
{arabia}	=>	{saudi}	0.101055807	0.707746479	3.665906	201
{korea,world}	=>	{north}	0.105077929	0.694352159	2.30562	209
{government,korea}	=>	{north}	0.114630468	0.693009119	2.30116	228
{korea,people}	=>	{north}	0.110105581	0.678018576	2.251384	219
{gaza}	=>	{israel}	0.13323278	0.674300254	1.684904	265
{human,people}	=>	{rights}	0.108597285	0.672897196	2.256986	216

{korea}	=>	{north}	0.199095023	0.671186441	2.228698	396
{human}	=>	{rights}	0.194067371	0.66897747	2.243838	386
{north}	=>	{korea}	0.199095023	0.661101836	2.228698	396
{people,rights}	=>	{human}	0.108597285	0.660550459	2.27701	216
{north,people}	=>	{korea}	0.110105581	0.659638554	2.223765	219
{rights}	=>	{human}	0.194067371	0.650927487	2.243838	386
{government,human}	=>	{rights}	0.100050277	0.650326797	2.181282	199
{north,world}	=>	{korea}	0.105077929	0.647058824	2.181356	209
{government,north}	=>	{korea}	0.114630468	0.645892351	2.177424	228
{china,killed}	=>	{people}	0.116641528	0.640883978	1.168394	232
{israeli,war}	=>	{israel}	0.104072398	0.640866873	1.601362	207
{israeli,people}	=>	{israel}	0.111613876	0.637931034	1.594026	222
{protest}	=>	{government}	0.133735546	0.636363636	1.131124	266
{killed,police}	=>	{people}	0.141277024	0.634311512	1.156412	281
{protesters}	=>	{police}	0.106083459	0.62797619	1.236678	211
{national}	=>	{government}	0.106083459	0.62797619	1.116215	211
{million,police}	=>	{people}	0.102061337	0.62654321	1.14225	203
{al}	=>	{government}	0.135746606	0.626450116	1.113502	270

根据置信度和独立度可视化关联规则，如图 3.9 所示，大小表示独立度，颜色深浅表示置信度。从图中可以看出：human ↔ rights，north ↔ korea，prime → minister 兼具高置信度和独立度。

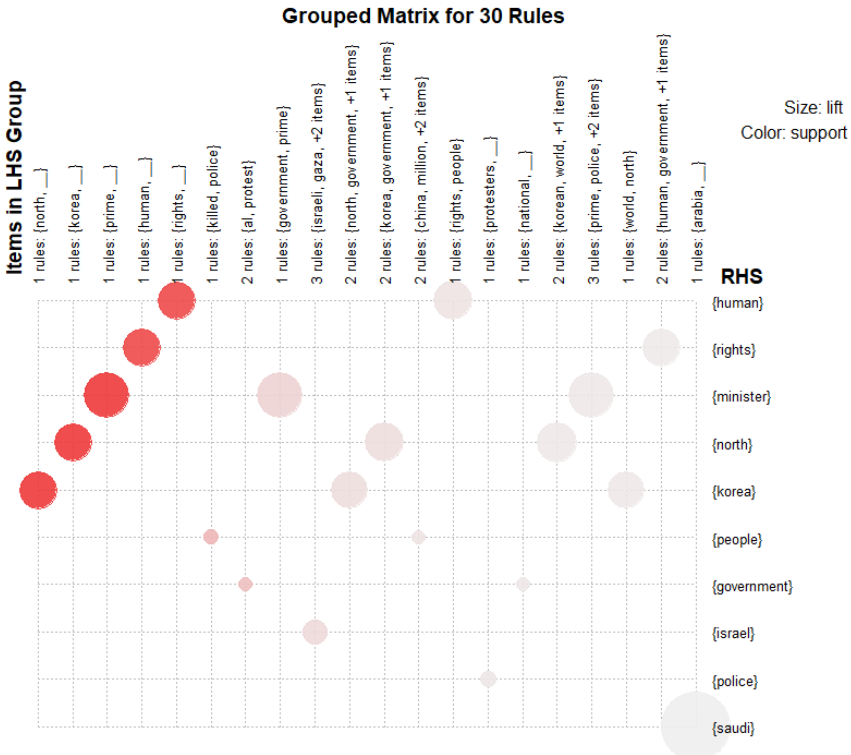

图 3.9　关联规则可视化

3.6 基于无监督的情感特征构建与评估

3.6.1 情感特征构建

情感分析，又称意见挖掘、倾向性分析等[17]，在自然语言处理中，情感分析一般是指判断文本所表达的情绪状态，换句话说，情感分析就是对主观性文本进行情感色彩倾向的分析和预测。情感分析技术在企业舆情分析、话题监控和口碑分析有着重要的应用；百度的 AI 开放平台就有情感分析的技术服务。情感分析的应用场景亦非常广泛，用户在购物网站（网易云，知乎，天猫、亚马逊、淘宝等）、电影评论网站、旅游网站上发表的评论分成正面评论和负面评论或者积极评论和消极评论等；或者还可以分析用户对于某一产品的整体使用感受，分析情感对象，用于对产品的改进等等。处理情感分析有很多种方法和分析角度。常见的情感分析等有利用机器学习有监督的技术对标注好的数据进行分类，例如对评论好坏的分类或者描述的真假性等等。目前有大量的深度学习应用于情感分析的例子，比如百度 AI 开放平台就有情感倾向分析的应用。如今互联网信息时代，数据变得日益庞大。在知乎，大众点评等网上可以搜到大量的用户评论消息，这些信息体现了文本的各种情感色彩和倾向性，如喜、怒、哀、乐和批评、赞扬等。如果对文本进行有效的情感词提取将会获得大量用于评估情感的信息。

如图 3.10 是本章节的技术概况，文中通过 being 的情感词典对文本进行匹配，从而以词的个数为依据划分文本的积极和消极性。然后利用 TF-IDF 文本向量化进行情感的区分以分析特征对情感的划分优劣性。

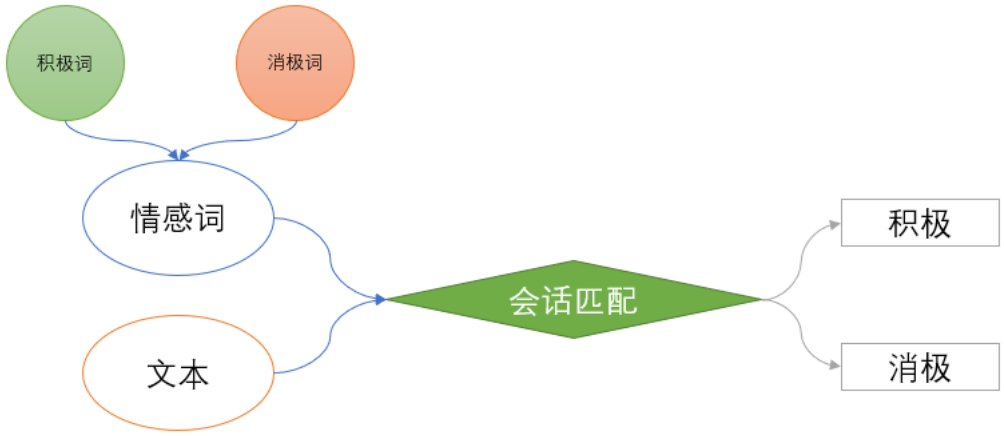

图 3.10 基于无监督的情感特征构建与检验

对所有$positive$（积极）词和$negative$（消极）各赋权重 1，对文本进行累积求和得到情感评估分数$emotionscore$：

$$emotionscore = positive - negative$$

接下来对情感词匹配的会话进行的文本词分类，如图 3.11，并通过词云的形式显示词频数大于 120 的词。另外注意，这些词并不一定出现在情感词表，但一定出现在文本中，比如“阳光很灿烂，好心情”，如果“好心情”是情感正面词，那么经过分词和去停用词后匹配的正面词是“阳光”、“灿烂”、“好心情”。文本中的正面词(黄色)有 top、free、support、peace、intelligence 等，从字面意思可见这些词都具有积极的意义，文本中的负面词(红色)有 killed、death、attack、prison 等，从字面意思可见这些词都具有消极的意义。因此这种方式进行的情感评估可解释性强并且较为科学。

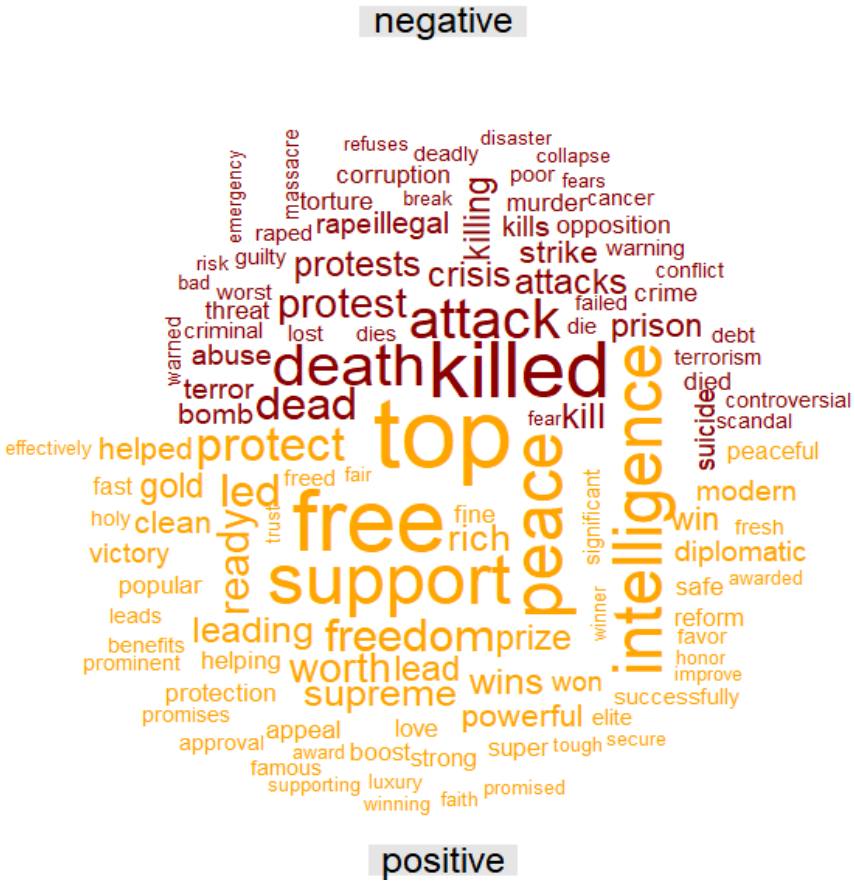

图 3.11 文本积极和消极情感词

接下来对 25 个热度（这里视为 25 个主题）进行情感分析。对每一个主题的会话（单位为日）进行情感评估分数可视化，显示部分主题（前十热度）的情感波动情况如图；从图 3.12 中可以大致推断这些主题的情感波动都较为剧烈,文本情感信息丰富。并且主题之间差距较大。因此测量的情感分数构建的特征方差也较高，特征显著性也应该较高。单纯从情感词的方式对文本进行情感分数建模往往没有考虑到情感词之间的叠加效果以及词于词之间的相互作用效果，为了充分对新闻标题进行情感信息的提取，我们需要采用一种有监督的情感倾向分数预测对情感信息进行更有效提取。

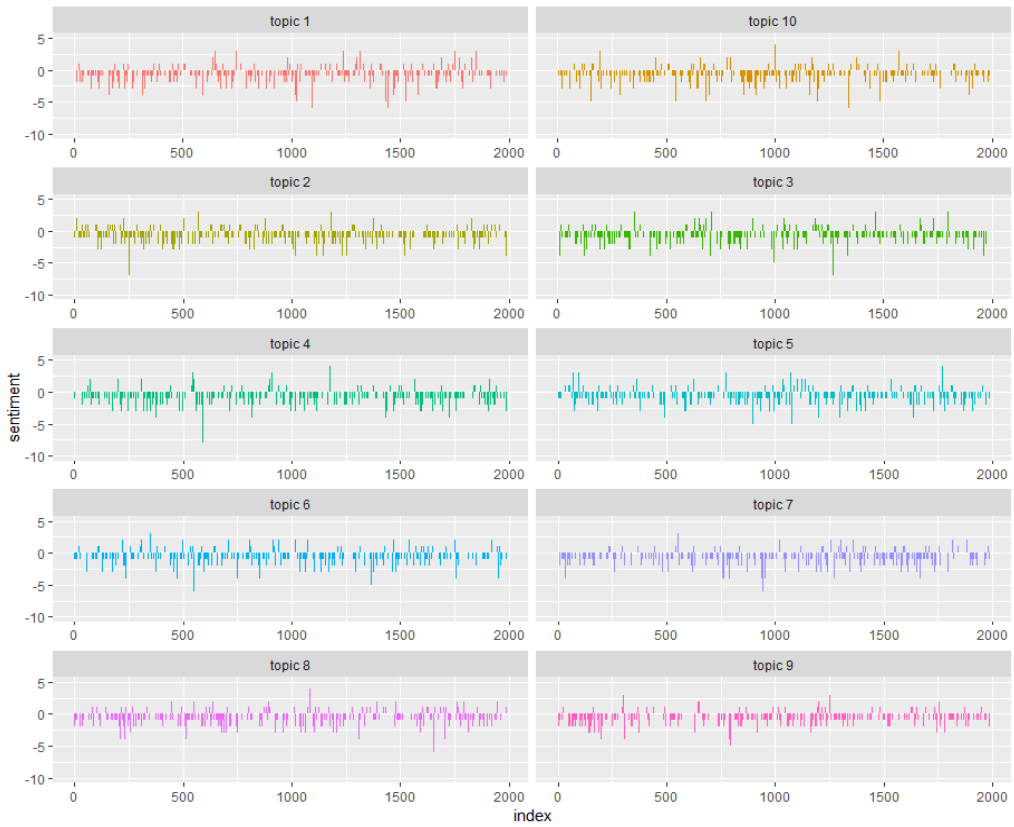

图 3.12 主题之间的情感波动

3.6.2 特征显著性检验与评估

二分类问题的预测结果将会是一个发生概率，这里将以阈值 0.5 区分两种分类情况（即大于 0.5 的归为一类，反之为另一类）。其中准确率是针对预测结果的，表示预测为正的样本中有多少个正样本。而预测为正有两种可能，一种预测情况为TP（为被模型预测为正的正样本，可以称作判断为真的正确率），另一种是FP（为被模型预测为正的负样本,可以称作误报率），这是属于产生误差的一种方式。表 3.7 为TP,FP,FN,TN的解释。

表 3.7 预测混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP(真正例)	FN(假反例)
反例	FP(假正例)	TN(真反例)

准确率P的表达式如下：

$$P = \frac{TP}{TF+FP} \tag{3.2}$$

召回率是针对样本的，它表示样本中的正例有多少被预测正确。分别有两种可能，

一种是把正例的预测正确为TP（为被模型预测为正的正样本，可以称作判断为真的正确率），另一种就是把反例预测错误了FN（被模型预测为负的正样本，亦可以称作漏报率），这是产生误差的另一种方式。

召回率R的公式如下：

$$R = \frac{TP}{TP+FN} \tag{3.3}$$

其准确度$Accuracy$公式如下：

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FN+FP)} \tag{3.4}$$

其中TN为被模型预测为负的负样本，可以称作判断为假的正确率。如果准确率P与召回率R的结果越高，则效果就越好，但二者之间存在着矛盾性，因此就需要加权求平均，来综合二者之间的关系。在这里引入 $F1$ 分数加权调和平均，即：

$$F1 = \frac{(\alpha^2+1)P*R}{\alpha^2(P+R)} \tag{3.5}$$

其中α为权重值。参数α为 1 时F公式如下：

$$F1 = \frac{2PR}{P+R} \tag{3.6}$$

ROC 的全称是 Receiver Operating Characteristic，该指标源于二战的雷达信号分析技术。对分类器在二分类问题存在数据不均衡时的预测结果有更好的评估合理性。根据分类器的预测结果对测试样本进行排序，按该顺序依次将样本作为正例进行预测，以“真正比例(TPR)”为纵轴，以“假正例率(FPR)”为横轴，即可得到 ROC 曲线。ROC 曲线纵坐标是对正类预测对的概率，横纵标是对反类预测错的概率，TPR与FPR的计算公式如下：

$$TPR = \frac{TP}{TP+FN} \tag{3.7}$$

$$FPR = \frac{FP}{TP+FP} \tag{3.8}$$

AUC(Area Under Curve)被定义为 ROC 曲线下的面积，这个面积的数值不会大于 1。使用 AUC 值作为评价标准是因为大部分时候 ROC 曲线并不能清晰的说明哪个分类器的效果更好，而作为一个数值，AUC 分数具有更好的区分性。分类器给出的是一个预测的概率，这个概率我们可以设置一个合理的阈值来区分正反类，阈值越高以为着正类预测的概率越大，反类预测错的概率也越大，因此正常 ROC 曲线都是随着横坐标的延伸，纵坐标也相应延伸，构成一个向左上方弯曲的曲线。一般正常情况希

望的是预测效果是预测正类对的概率很大，相应预测反类错的概率很小，那么 ROC 曲线就越往上方靠近，AUC 分数也就越大。因此，AUC 分数越大的分类器效果越好。

表 3.8、3.9、3.10、3.11 是高斯贝叶斯、Logistic 分类算法、决策树和支持向量机在测试集的表现情况。

表 3.8 GAU evaluation

	precision	*recall*	*f1-score*	*support*
0	0.81	0.94	0.87	463
1	0.38	0.15	0.21	122
micro avg	0.77	0.77	0.77	585
macro avg	0.59	0.54	0.54	585
weighted avg	0.72	0.77	0.73	585
AUC: 0.5413730835959352				

表 3.9 Logistic evaluation

	precision	*recall*	*f1-score*	*support*
0	0.83	0.97	0.89	463
1	0.67	0.24	0.35	122
micro avg	0.82	0.82	0.82	585
macro avg	0.75	0.6	0.62	585
weighted avg	0.8	0.82	0.78	585
AUC: 0.7314732854158552				

表 3.10 Decisiontree evaluation

	precision	*recall*	*f1-score*	*support*
0	0.86	0.9	0.88	463
1	0.56	0.45	0.5	122
micro avg	0.81	0.81	0.81	585
macro avg	0.71	0.68	0.69	585
weighted avg	0.8	0.81	0.8	585
AUC: 0.67789363736147				

表 3.11 SVM evaluation

	precision	*recall*	*f1-score*	*support*
0	0.83	0.94	0.88	463
1	0.53	0.25	0.34	122
micro avg	0.8	0.8	0.8	585
macro avg	0.68	0.59	0.61	585
weighted avg	0.76	0.8	0.77	585
AUC: 0.7363948589030911				

综上看，根据 AUC 评估指标得知，情感分类效果最好的是 SVM 和 Logistic 分类

器，AUC 分数高达 0.73 左右。*F1* 分数高达 0.82 左右。分类效果最不理想的是 RNN 神经网络，可能由于数据采样不足造成，但是 SVM 和 Logistic 已经达到理想效果，因此在这可以不考虑 RNN 神经网络。因此可以认为构建的情感分数特征区分积极和消极的会话是显著的。为后续的预测作为特征之一。

3.7 基于 Attention-Based Bi-LSTM 模型的情感分析

图 3.13 情感特征生成示意

本章的思路框架如图 3.13 所示，通过构建的基于 Attention-Based Bi-LSTM 模型（基于注意力机制的双向 LSTM 模型）训练一份已经标注的 Jigsaw 文本数据集，利用训练好参数的模型预测新闻标题的情感倾向。在这份文本数据集中，标注了包含不同地域、宗教信仰、人种、性别等会话的情感分数，如图 3.14 所示，对会话 Toxic 分数设定阈值为 0.5，划分 Toxic 和非 Toxic 的会话，统计两类会话在不同宗教信仰和人种的区别。可以看出，基督教和白人中的非 Toxic 会话频数最低，穆斯林教的 Toxic 会话频数最高，白人的 Toxic 会话频数也是最高的。另外通过图 3.15，可以得知，情感分数都在 0 至 1 的打分范畴，其中 Toxic 比重远低于非 Toxic 的会话比重。

图 3.14 情感分数随宗教信仰和人种的分布情况

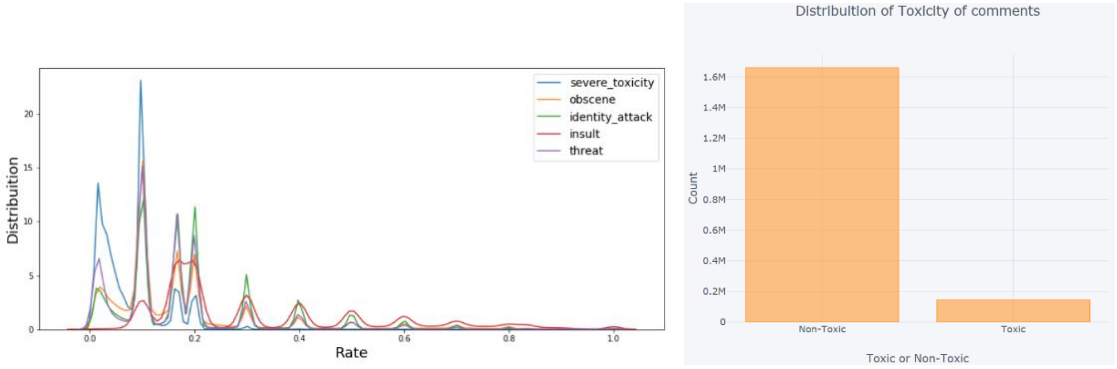

图 3.15 不同情感分数的分布情况

网络模型的整体构建上，综合 Embedding、Bi-directional LSTM 层和全连接层等构建一个完整的训练网络。在神经网络结构中引入注意力机制可以极大提高情绪识别的准确度[18]，通过引入 Embedding 层可以防止网络模型对文本信息的提取“断章取义”，忽视词与词之间的关系。另外，为了防止过拟合，本文引入三种类型的 Dropout 层，包括了 SpatialDroupout1D、GlobalAveragePooling1D、 GlobalMaxPooling1D 三种 Dropout 层，这样保证了整个网络模型的科学性，对文本情感倾向的提取准确度有极大的提高。本文模型对通过训练 Jigsaw 文本数据集预测 Reddit WorldNews Channel 文本数据集的六个维度的情感倾向（见图 3.16），包括 severe toxicity（恶毒性质）、obscene（淫秽）、identity attack（人身攻击）、insult（侮辱）、threat（威胁）、sexual explicit（性暴露）。

图 3.16 情感分析生成的八个情感维度

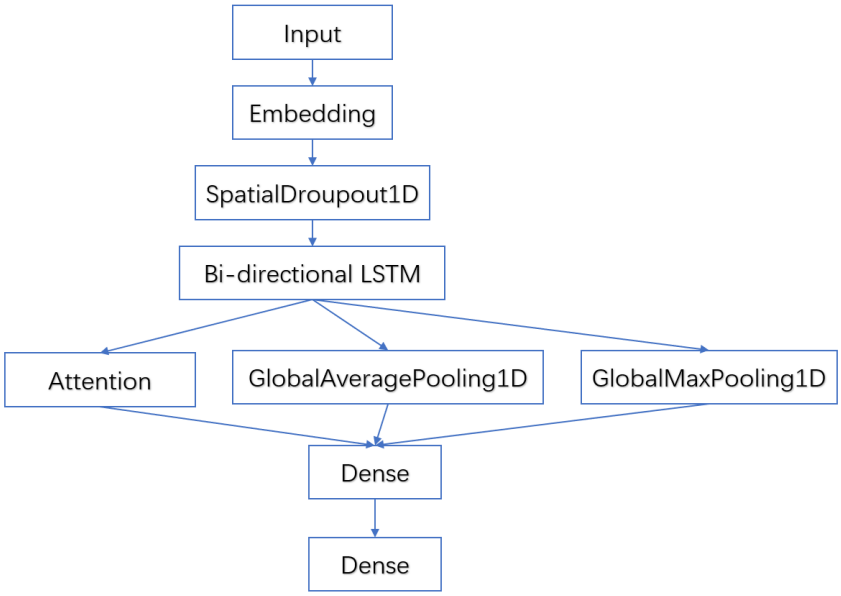

图 3.17 完整网络层解决方案

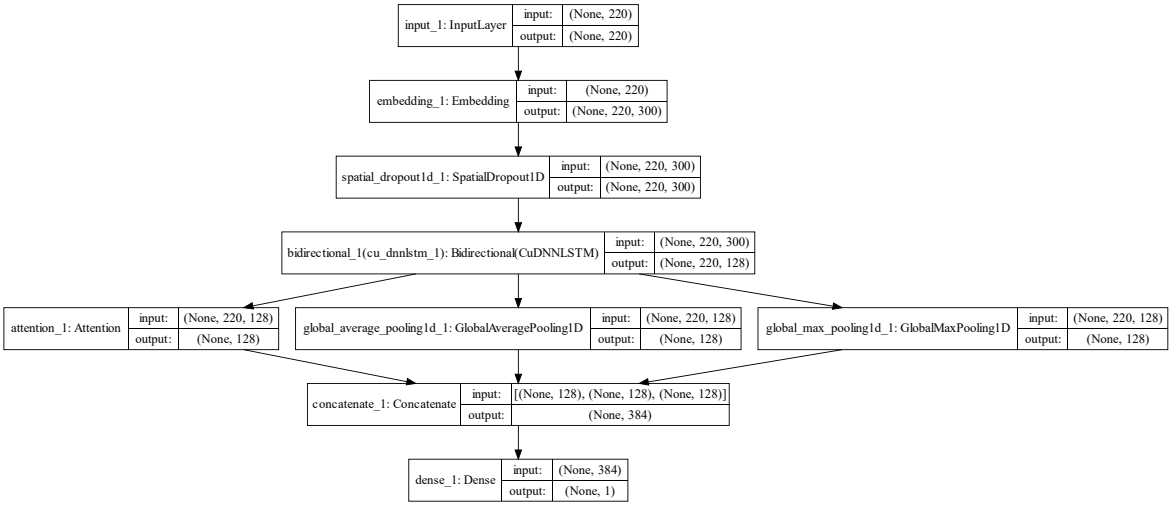

图 3.18 网络层连接关系

网络层的连接关系如图 3.17，3.18 所示，总计参数 2534738 个。为了加速运算，这里用的事 Nivida 1070Ti 显卡进行训练，一个 batch 训练 2048 组数据，总共训练 10 个周期，通过五折交叉验证完成训练和验证过程（训练集和测试集比例为 4:1）。完成一次训练时间大概在 2 个小时。之后将训练结果和模型直接在原始数据集预测，构建新的情感特征。

每一个周期的的训练集损失情况如下，可以明显看出，损失随着周期数的增长有所下降，精度也随之上升。但是过多的训练周期可能导致过拟合，考虑时间成本和降低过拟合的风险，这里只训练 6 个周期。根据表 3.12 显示的交叉验证第一折的训练情况，训练集和测试集（表中含 val 的列）随着每一周期（epoch）的梯度下降训练，

除了第 5 周期陷入异常，层的损失函数值（loss）随着训练周期的进行也进一步下降，准确率 acc 也逐步上升。最终对 Toxic 的五折交叉验证的 5 份预测结果取算术平均，最终预测结果 AUC 分数高达 0.97！接下来将训练的模型在 Reddit WorldNews Channel 新闻标题数据集中预测情感分数，构建新的情感特征。

表 3.12 训练周期变化情况

epoch	loss	acc	val_loss	val_acc
1	0.1169	0.9584	0.0923	0.9646
2	0.0935	0.9644	0.0909	0.9652
3	0.0893	0.9658	0.0881	0.9662
4	0.0867	0.9664	0.0927	0.9633
5	0.0847	0.9673	0.1025	0.9585
6	0.0831	0.9677	0.0942	0.9628

如图 3.19 所示（左归一化前的区分，右归一化后的区分）是对文本所有情感特征归一化降维后的分类（股价第二天上升或者下降可视化），透过直方图和散点图可以看出，即使降维了会造成原始信息的一定程度损失，不同分类情感特征分布是有差别的，因此初步可以断定当天新闻标题情感特征对未来一天的股价波动具有一定冲击。但是这个差别还不是很明显，因此对股价单纯新闻头条的情感特征对股价涨跌还是远远不够的，后续再根据信息增益进一步评估特征的显著性。

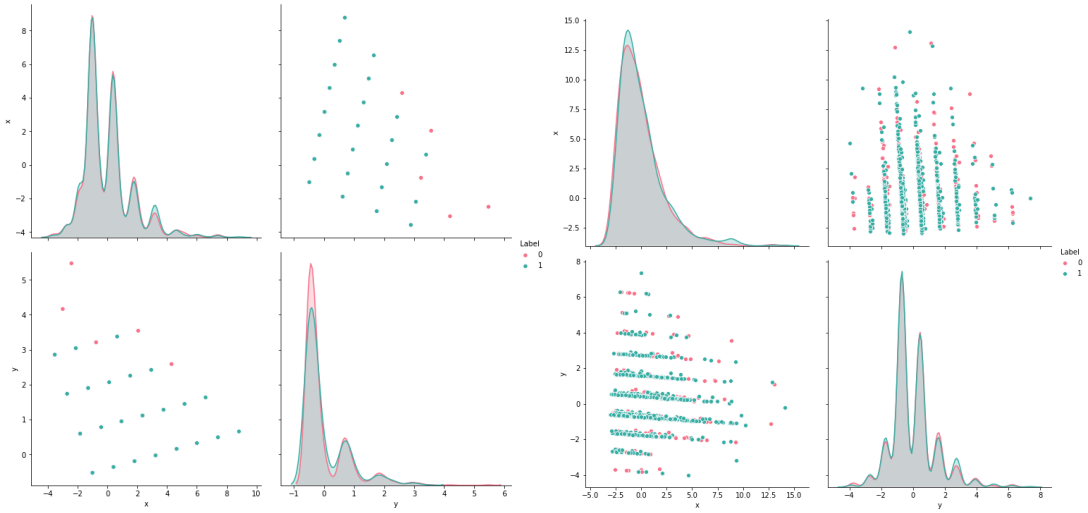

图 3.19 以第二天股价涨跌进行分类的情感特征

3.8 股价技术指标分析与特征集构建

3.8.1 策略指标描述性统计分析

本文选用道琼斯股价指数八年的数据（从 2008 年到 2016 年，记录单位为一日）。原始数据中包含开盘价，收盘价，最低价，最高价，除权收盘价，日交易量。图 3.20 是除权收盘价随交易日变动的情况，可以看出价格趋势是先下降再上升，并且小周期波动情况较大，总体趋势是上升的。

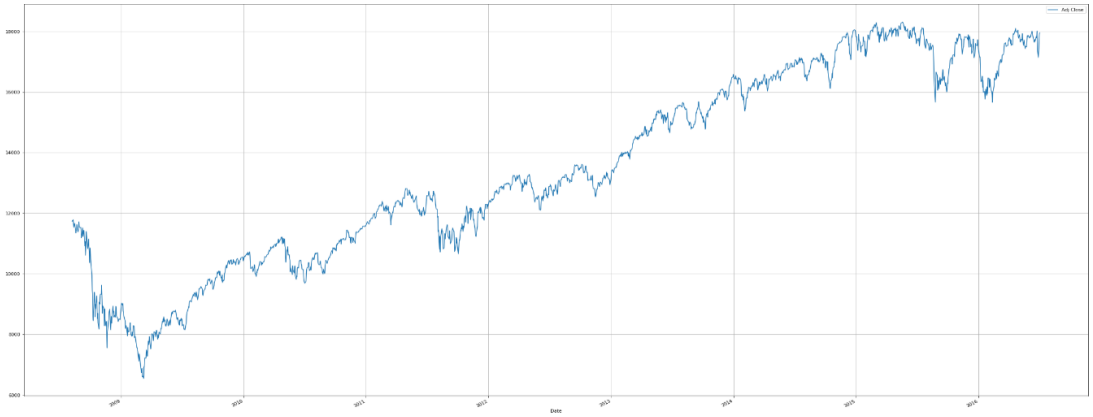

图 3.20 股价 8 年以来的走势图

接下来我们对五个指标进行直方图可视化，并做两两变量的散点图和直方图（见图 3.21）。通过散点图，可以看见除了日交易量，指标之间的相关性极大，并且分布情况也几近一致。

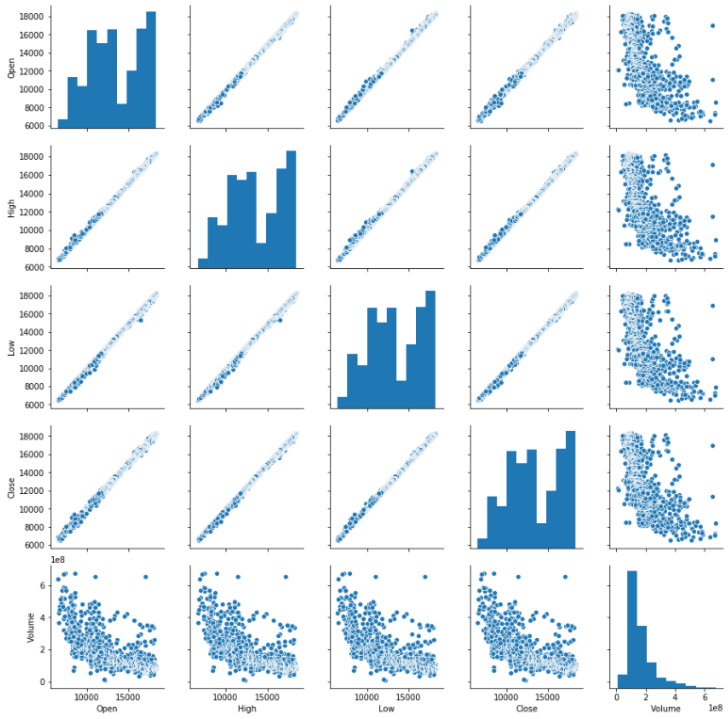

图 3.21 开盘价、最高价、最低价、收盘价和成交量散点图

接下来试着通过 *Pearson* 相关系数进行统计分析，分析变量之间的相关性程度。

Pearson 相关系数的定义如下所示。

*Pearson*相关系数衡量两个序列的线性相关的程度，假设X、Y是两个序列，记$X=(x_1,x_2,x_3,x_4,\dots,x_N)$，$Y=(y_1,y_2,y_3,y_4,\dots,y_N)$

相关系数定义如下:

$$r=\frac{Cov(X,Y)}{Std(X)\cdot Std(Y)} \tag{3.9}$$

其中Cov是协方差，Std是标准差，E是期望。

协方差的计算公式如下:

$$Cov(X,Y)=E[(X-E(X)(Y-E(Y)]=\sum_i^N\frac{(x_i-E(X)(y_i-E(Y))}{N-1} \tag{3.10}$$

协方差代表了两个序列同时偏离均值的程度，相关系数是用协方除以两个序列的标准差的乘积，相关系数的特性如下:

（1）.若$r>0$，说明两个序列的变化呈现正关系。

（2）.若$r<0$，说明两个序列的变化呈现负关系。

（3）.若$r=0$，说明两个序列的变化相互独立。

如图 3.22 是五个指标的相关性统计，可以看见开盘价，收盘价，最高价，最低价，收盘，除权收盘价两两之间的相关系数(通过 Pearson 相关系数定义)快接近为 1(图 37 中数字只是近似，由于计算过程精度有限造成的误差)，和交易量的价格都是高度负相关。因此为了避免数据冗余以及研究的重复，这里只以除权收盘价为研究对象进行分析。

除权收盘价的计算公式如下：

记复权因子为δ，r_t为第t天的涨跌幅，则第t天的复权因子为：

$$\delta_t=\prod_{i=1}^{t}(1+r_t) \tag{3.10}$$

当天除权收盘价c_t为：

$$c_t=\frac{\delta_t\cdot c_1}{\delta_1} \tag{3.11}$$

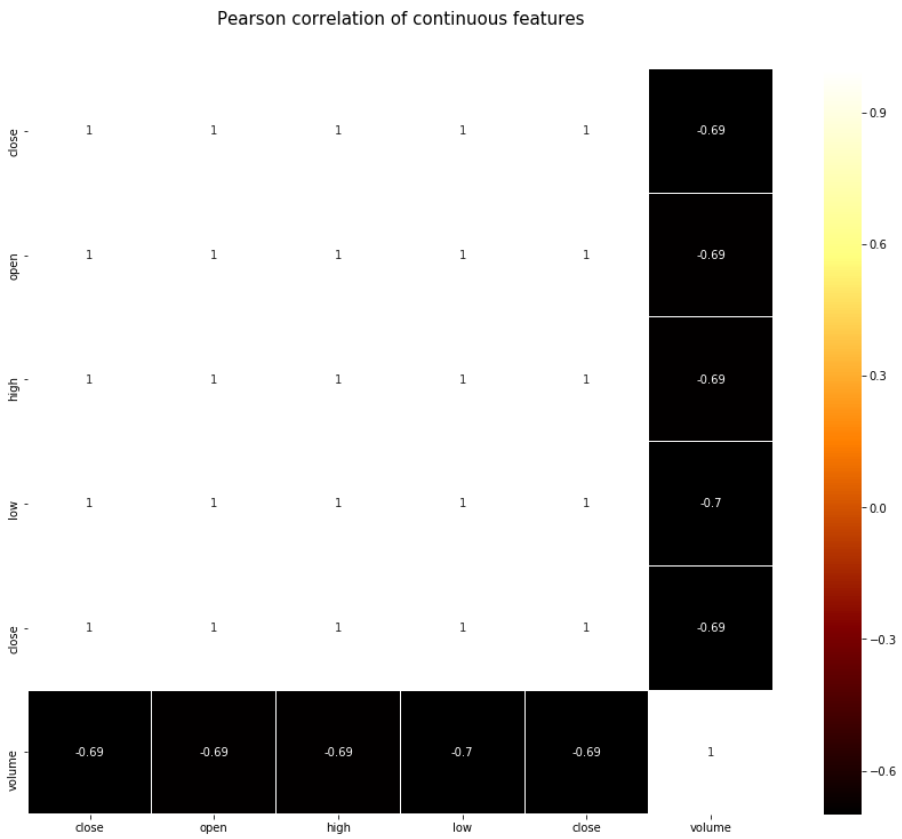

图 3.22 开盘价、最高价、最低价、收盘价和成交量相关性

3.8.2 股价技术指标特征规约

表 3.13 策略指标标示

股价指标	本文表示符号
CR 指标	*cr*
KDJ 指标	*kdjk,kdjd,kdjj*
SMA 指标	*close_5_sma,close_10_sma*
MACD 指标	*macd,macds,macdh*
BOLL 指标	*boll,boll_ub,boll_lb*
RSI 指标	*rsi_6,rsi_12*
WR 指标	*wr_10,wr_6*
CCI 指标	*cci,cci_20*
TR、ATR 指标	*tr,atr*
DMA 指标	*dma*
TRIX，MATRIX 指标	*trix,trix_9_sma*
VR，MAVR 指标	*vr,vr_6_sma*

技术指标是通过股价基本指标收盘价、成交量、涨跌率等根据统计学原理构建的预测未来趋势的参考指标。技术指标基于的假设是投资者的心理情绪以及市场价格的

变化因素都反映在了技术指标中，过往大量投资者依据技术指标把握住了股票买卖的时机获得收益。对股价趋势具有一定的成效。因此以技术指标作为特征规约的方法具有一定的科学性。技术指标顾名思义主要分为三类①趋向型技术指标②强弱型技术指标③随机买入型技术指标。本文所选的技术指标分布在三个类别中（见表 3.13），最大程度保证股市价格潜在信息的充分获取。

接着，对生成的多个策略指标进行相关性统计，根据策略指标的两两相关性系数（图 3.23）可以看出来，高相关性的占少数，高度正相关性和高度负相关性的指标大致相同。移动均线之间的相关性接近为 1。因此接下来尽可能选择避免特征冗余的算法进行分析。通过 LightGBM 和 XGBoost 实现的 GBDT 算法对特征冗余不会太敏感，因此后面可以进行特征贡献度的评估和算法建模，从而提取高显著性特征集。

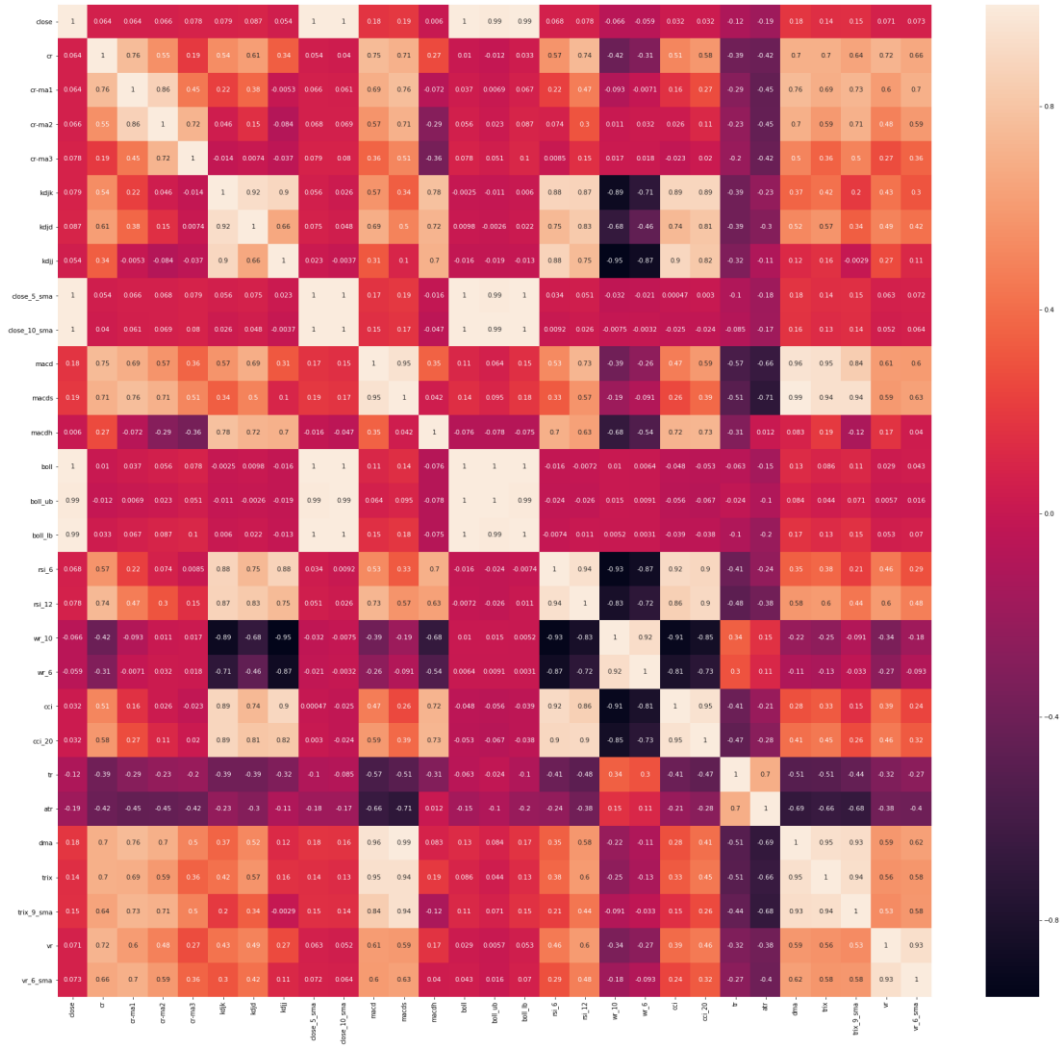

图 3.23 策略指标相关性

3.9 算法策略构建与回测

3.9.1 特征工程描述

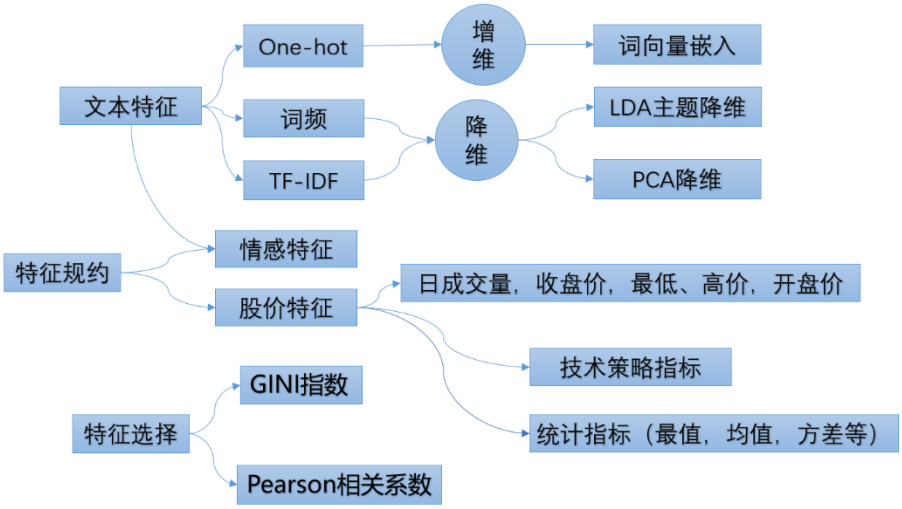

图 3.24 特征构建与评估

特征工程通过图 3.24 所示的框架进行，通过对文本特征，连续特征，离散特征分别处理；对文本进行 2-grams，3-grams 法分离，one-hot，tf-idf 特征提取，另外利用情感分析特征规约；对连续特征进行股价技术指标衍生，对所有股价技术指标统计特征（最大值，最小值，平均值，标准差）提取，分别对特征集进行分类算法的 AUC 分数评估，根据互信息论对特征重要度进行排序。另外对高维特征如文本 one-hot 编码，TF-IDF 等特征进行 PCA 和 LDA 降维（*pca1*、*pca2* 表示 PCA 降维得到的两个主成分，*lda1*、*lda2*、*...lda6* 代表 LDA 降维得到的特征），根据 ROC 曲线和 AUC 分数评估特征显著性。

GDBT 算法的基模型 CART 决策树是通过基尼指数（而非熵）构建的信息增益（见公式 2.11）实现特征选择的，基尼指数可以描述随机变量不确定性的程度，信息增益是描述特征对分类的区分纯度，一般地，特征包含的信息量越大，不确定的因素越高，特征重要度也就越低，对应信息增益就越大，基尼指数具体公式如下：

记$p_k = \frac{|D_i|}{|D|}$为样本（以目标类别或者特征类别分类的样本）D_i在样本全集 D 的概率（频率），则：

$$Gini(i) = \sum_{p_k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2 \qquad (3.12)$$

其中$\sum_{k=1}^{K} p_k = 1$， p_k是第p_k个变量(特征)发生的可能性，$Gini$是基尼指数，K 是变量(特征或者目标)种类的总数。当$p_k = \frac{1}{K}$时，$Gini$取最大。对应特征 E 下,假设样本全集 D 被分为D_1和D_2假设的条件概率有：

$$Gini(D,E) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2) \qquad (3.13)$$

其中，$Gini(D_1)$和$Gini(D_2)$是D_1和D_2分别以目标类别计算的基尼指数。

传统 GDBT 算法根据基尼指数进行特征选择，但 LightGBM 的 leaf-wise 策略是根据当前所有叶子节点中选择收益最大的节点进行分裂。因此在计算特征重要度上略有不同，每个特征的重要度计算应该是特征的平均信息增益或者在模型中使用该特征的次数（the number of times a feature is used in a model），图 3.25 是基于信息增益描述的一个特征重要度排序，并从高到低特征重要度依次降低进行排名。特征对目标的区分效果就越好(越纯)，因此它的熵或者基尼指数越小，因此信息增益也越大，说明这个特征也就越重要，比如区分人类的性别，第二性征的区分效果比男女的身高效果更好，因为第二性征的区分度明显，以该特征分类的基尼指数也就越小，信息也增益越大。

可以看出 *wr* 指标和 *kdj* 指标对未来一天股票价格变动影响最大。从图 3.25 来看，*positive* 和 *mean* 指标的特征贡献度最低，除了 *positive*、*negative* 其他情感特征的特征贡献度都大于 10 日均线和布林带(*boll*)的特征贡献度，因此新闻标题情感因素对股价波动的影响是较为显著的。

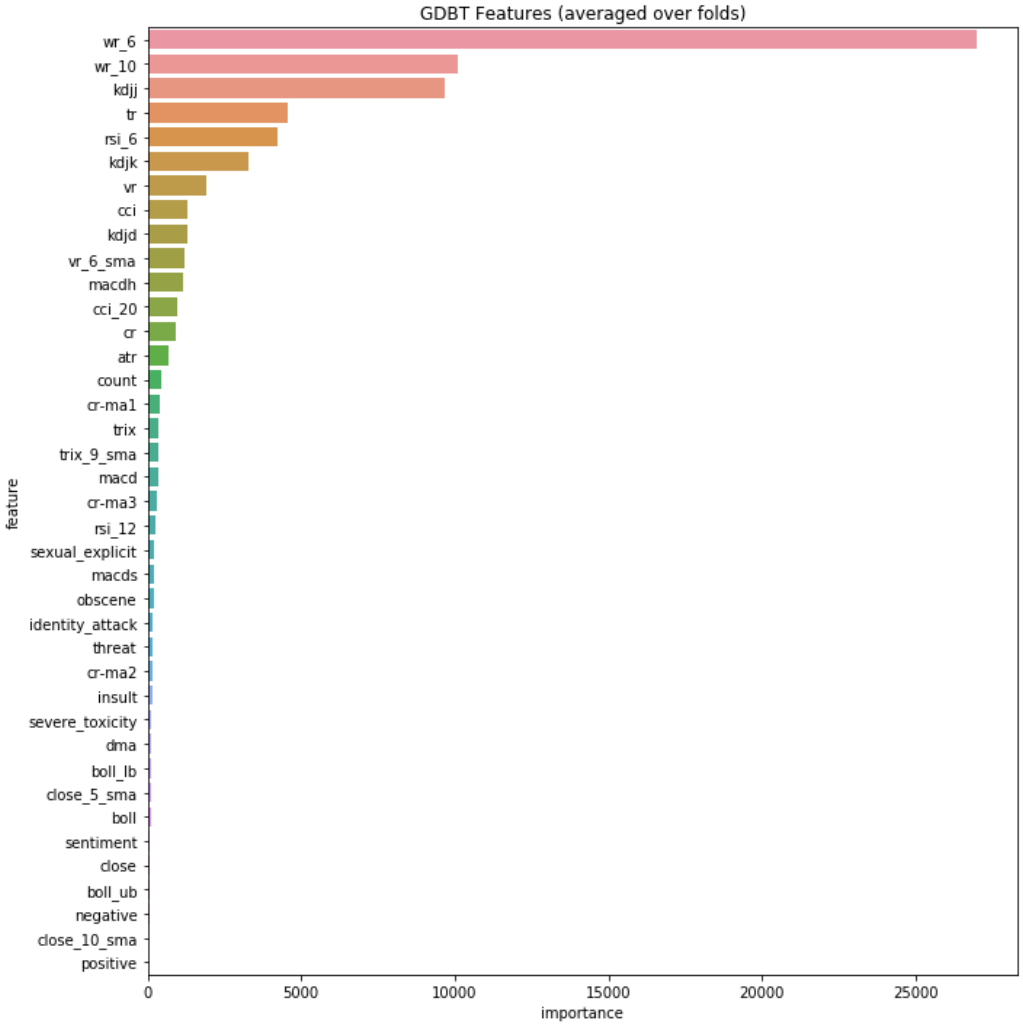

图 3.25 特征重要度排名

3.9.2 算法融合方案

根据如上，发现情感分析特征和策略指标的特征预测效果是最好的。于是我们将该特征集作伪最终特征集。分别利用决策树算法，高斯贝叶斯，支持向量机，随机森林，以及 GBDT（GBDT 通过 XGBoost 和 LightGBM 实现）进行训练。之后，将训练的测试结果进行 Stacking 融合和 Blending 融合。模性的 AUC 分数见表 3.14。表 3.14 中个别算法采用了简写，如决策树用 DT 表示，随机森林用 RF 表示，支持向量机用 SVM 表示。

表 3.14 算法 AUC 分数评估

Models	SVM	Neural Network	DT	RF	GBDT	Logistic	Blending	Stacking(Bayes)	Stacking(Logistic)	Stacking(GBDT)
AUC	0.940	0.749	0.733	0.884	0.892	0.926	0.916	0.780	0.881	0.878

算法融合方法多样，其中 Stacking 融合和 Blending 融合，以及 Boosting 融合和 Bagging 融合，本文将用 Stacking 融合和 Blending 融合训练所有算法。一般而言，当各个基线模型的效果越好，Blending 融合效果越好。当基线模型（Baseline model）的预测结果之间相关性越低，那么 Stacking 融合效果越好。在不对算法筛选的情况下，直接融合基线模型 DT（决策树），SVM（支持向量机），LOG（Logistic 分类），RF（随机森林），GBDT（GBDT 算法）等算法，并对比不同融合算法的性能差异。

将不同模型的预测结果融合起来生成最终预测，融合的模型越多，效果往往会越好。由于结合了不同的基线预测，它们的性能至少等同于最优的基线模型。给每一个模型分配一个权值w_i，权值总和$\sum_{i=1}^{n} w_i$为 1。融合模型定义如下：

$$e(x) = \sum_{i=1}^{n} w_i f_i(x) \qquad (3.14)$$

Blending融合的原理既是将输入数组 X 通过多个预处理 $pipeline$ 输入至多个基学习器$f_i(x)$，将所有的基线模型的预测结果加权平均起来，生成最终的预测结果P。

当所有单模型训练完成之后，所有模型的集成构建成一个带权值的线性分类器，通过训练权值参数，提高融合模型的分类效果，Stacking 融合是将多个基线模型$f_i(x)$的预测结果作为 Meta-classifier（另一个基线模型）的训练特征集，然后输出预测结果 P。图 3.26 是 Stacking 融合原理。

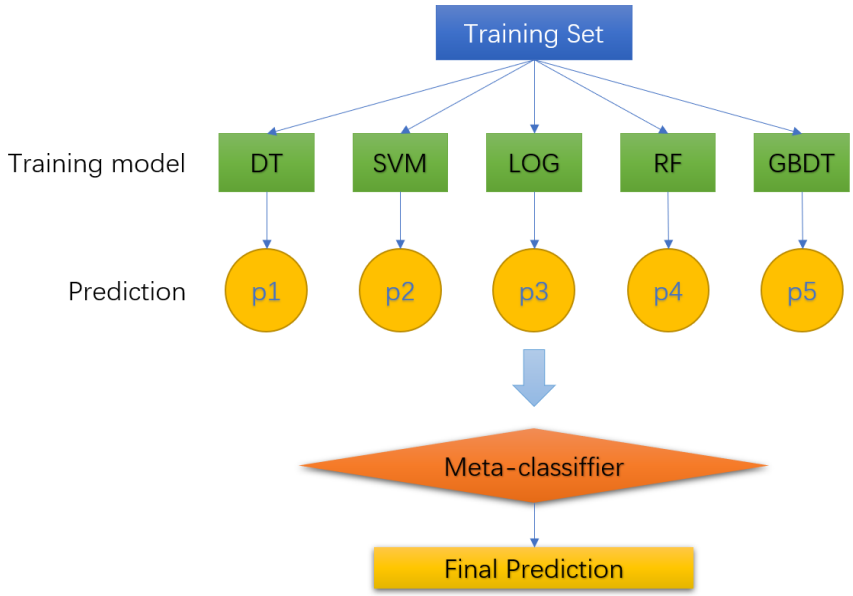

图 3.26 Stacking 融合示意

图 3.27 是算法的预测结果之间的相关性。

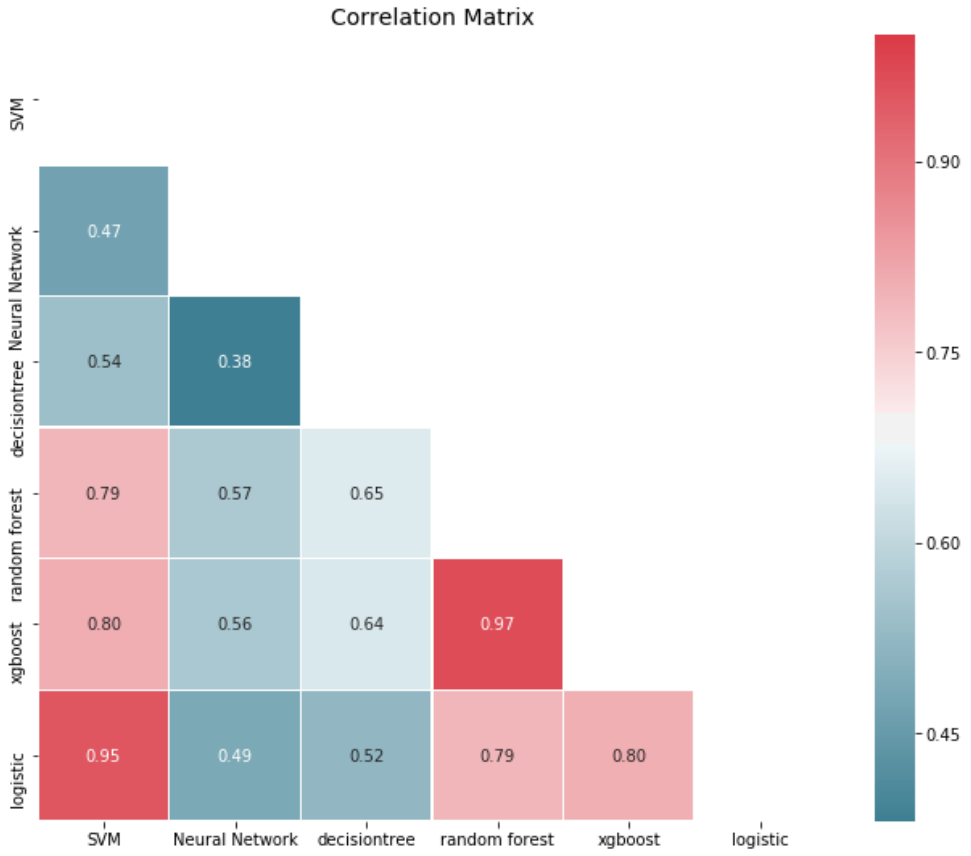

图 3.27 预测结果相关性

通过 Pearson 相关系数描述（图 3.27）的相关性强度得知，其中是 GBDT（XGBoost

实现）和 Random forest（随机森林）的预测结果相关性最高为 0.97, 其次 SVM 和 Logistic 预测结果相关性为 0.95。其余分类算法之间的相关系数都小于 0.80。值得注意的是 SVM 在这个数据集中的预测 AUC 分数最高（0.94），其次 logistic（0.926）。对于融合方法，Blending 融合分数最高，达到 0.916。对于 stacking 融合，目标分类器的不同会导致结果差异较大。比如设置高斯贝叶斯为目标分类器，预测的 AUC 分数仅 0.636；设置 logistics，预测的 AUC 分数为 0.883，如果设置目标分类器为 XGBoost，那么预测的 AUC 分数为 0.882。

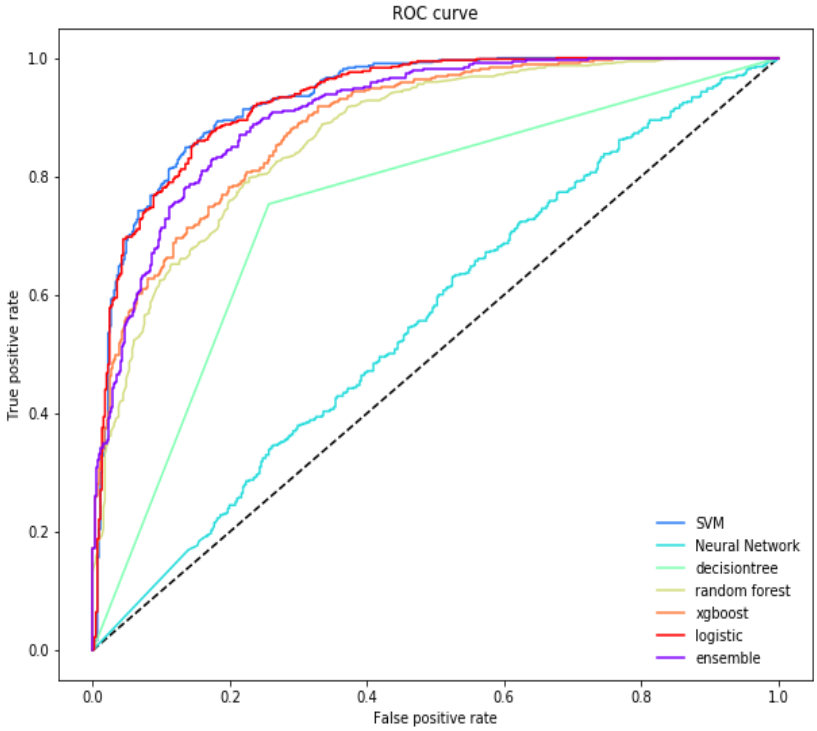

图 3.28 多分类器的 ROC 曲线

对于如图 3.28 所示分类性能的 ROC 曲线描述，越往左上方偏斜，说明分类效果越好，可以看出 Blending 融合的之后的预测结果不比 SVM（支持向量机）和 Logistic（逻辑回归）的效果好，因此后面需要删除一些效果差的分类器，比如 Decisiontree（决策树）和 Neural Network（神经网络）。于是本文最终的算法融合方案是采用 GBDT、Logistic、RandomForest 和 SVM 的 Blending 融合。最终 Ensemble 的在测试集上的 AUC 分数为 0.95。

3.9.3 量化交易系统构建及回测演示

依据前面构建的算法，训练 2008 年 8 月 8 日至 2012 年 7 月 18 日的道琼斯价格数据，回测 2012 年 7 月 19 日至 2016 年 7 月 1 日的历史数据。根据当天构建的特征集预测下一天的股价上涨和下跌情况，从而根据预测情况开仓和平仓。本文通过算法

预测了下一天涨跌幅的可能性，可以设置不同的阈值来多头和空头来根据市场情况来设置保守和激进的策略。这里为了衡量算法交易的稳定性，尽可能将参数和环境保守设置，例如以当天收盘价格买入。

由于预测结果是一个在 0 和 1 之间概率，跟进原来的数据标注情况，1 为股价上涨，0 为股价下跌；因此越靠近 1 的预测结果说明股价第二天下降的可能性会越大，反之，越靠近 0 说明股价第二天上升可能性越大。这里将预测结果小于 0.5 的设置为开仓，大于等于 0.5 的设置为平仓。按照 0.5 的阈值来划分预测结果，测试结果的准确率是 86.6%。

3.9.4 刻画资金曲线和回测分析

回测(backtesting)是依据股票历史交易数据以验证策略可行性和有效性的过程，可有效降低投资者在按照自己构建的策略在实盘时的盲目和风险，也是量化投资决策中的一个不能忽视的环节，回测广泛应用于股票、期货、期权、外汇等交易领域。通过回测，可以验证策略在历史上的盈利情况，从而为将来的交易决策提供参考依据。按照真实市场交易并不代表每次股票的买入和卖出都能成功，因此对算法策略的优劣评判不单纯根据算法预测的准确度进行衡量，算法质量的评估必须通过回测来评价，从而能够更可靠地分析所构建策略将带来的盈利和风险。

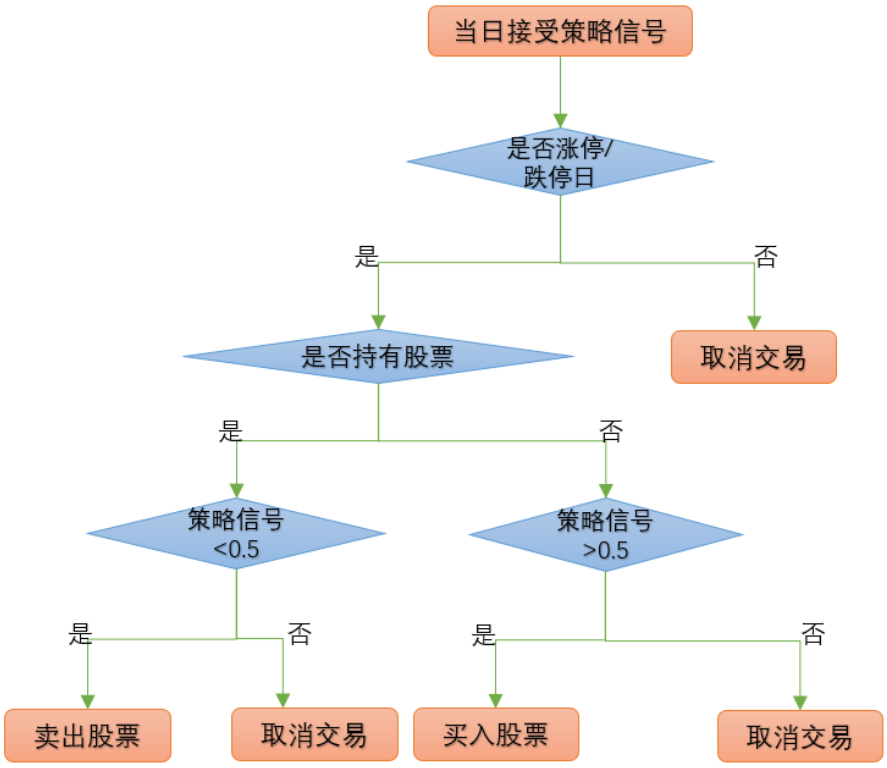

图 3.29 回测机制说明

图 3.29 为回测实现的框架。在发出信号的当根 K 线以收盘价买入，记第t日买入

股票的数量为$buynum_t$，设滑点为σ，手续费为$rate$，初始资金$initial$，第t日的收盘价C_t，除权收盘价为AC_t，则：

$$buynum_t = \frac{initial \cdot (1-rate)}{AC_{t-1}+\sigma} \tag{3.15}$$

记第t日卖出所有持有股票，记印花税为δ，则第t日的现金变动$\Delta CASH$为：

$$\Delta CASH = buynum_t \cdot (C_t - \sigma) \cdot (1 - rate - \delta) \tag{3.16}$$

模拟中国股票市场的 t+1 制度并且特殊情况例如涨跌停不能买入卖出股票。分析按照算法交易的每次下单买入股票数量、账户净值、账户净值累乘涨跌率（资金曲线刻画）、账户净值涨跌率，随着 2012 年 7 月 19 日至 2016 年 7 月 1 日的变化情况。

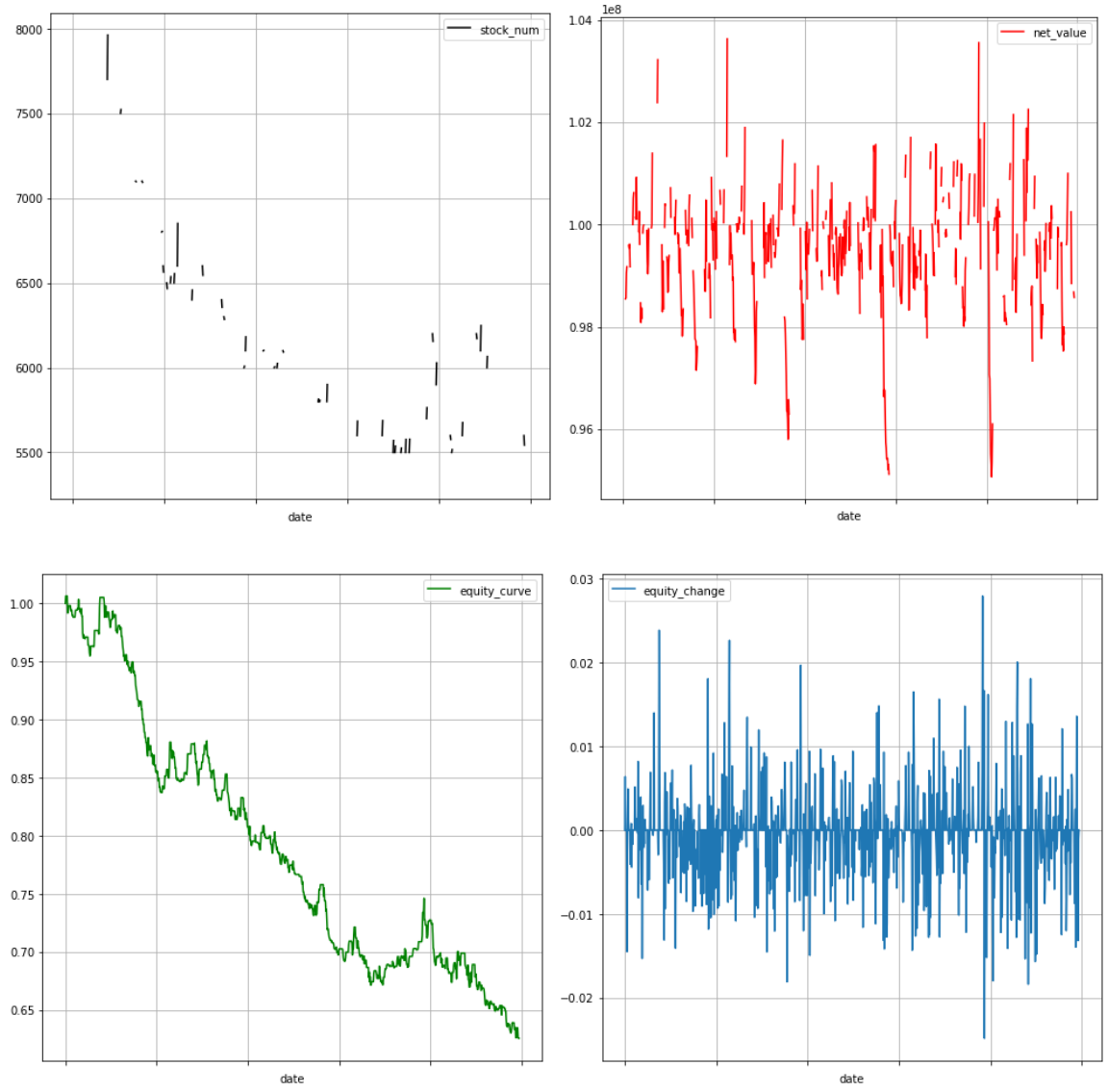

图 3.30 回测结果（每次买入股票数量、净资产、资金曲线、资金曲线的斜率）

此回测中，初始资金设置为 100000000 元，滑点设置 0.01（股票默认为 0.01 元，ETF 为 0.001 元），手续费默认为万分之 2.5，印花税默认为千分之 1。

为了充分评估策略的好坏，这里引入最大回撤率（max drawdown）。最大回撤率可以代表策略的最大风险，最大回撤率越大，策略的风险越高。投资者可以根据这一指标判断策略是否在他们的风险承受范围以内。最大回撤率的公式计算如下。

记账户当日净值为K，当日之前账户的最高净值为K_{max}，则最大回撤率τ计算公式为：

$$\tau = max(1 - \frac{K}{K_{max}}) \tag{3.17}$$

表 3.15 回测结果评估 1

回测评估指标	最大回撤率	收益率均值（天）	收益率方差	交易频率（四年总）
值(%)	38	-0.04	0.5	33.0（328 次）

图 3.30 是每次下单买入股票数量、账户净值、账户净值累乘涨跌率、账户净值涨跌率的情况。*stock_num*，*net_value*，*equity_curve*，*equity_change* 分别为每次下单买入股票数量、账户净值、账户净值累乘涨跌率、账户净值涨跌率。回测是在考虑滑点、手续费以及印花税并严格模拟真实市场交易的情况分析策略的可行性。依靠单纯的算法交易下单和撤单，买入股票（开仓）成功的次数 164 次等于卖出股票（平仓）的次数，交易频率为 32.96%，资金曲线（见图 3.30，*equity_curve*）最后一日为 0.625427，说明最后一天的净资产净值比原始资产净值跌了 0.375 左右。回测评估结果见表 3.15。可以看出即使能在高准确率的预测算法下，实盘交易任然是亏损的。表 3.16 是预测的混淆矩阵。

表 3.16 算法预测情况的混淆矩阵

	上涨	下跌
预测上涨	425	36
预测下跌	91	443

为何预测 AUC 分数达到 95%，未来四年预测准确率为 87.2%的算法构建的策略放到实盘亏损如此严重？由于较大的最大回撤率可以导致我们一次预测错误的亏损使盈利为负，通过最大回撤率可以看出（对比表 3.15 和表 3.17）,一部分原因是算法没有预测下一天的涨跌幅，可能导致的结果是，例如四天内，预测股价上涨预测对了三天，但是一次错误的预测导致最后一天跌的幅度比三天累积盈利的幅度还要大的多，这样就造成了我们亏损。通过分析回测账户净值、资金曲线以及其变化率，可以看出有部分股票是平仓失败和开仓失败导致的亏损，这也导致算法交易最终没有按照算法的实际预测情况进行操作，导致在实盘中亏损。另外亏损的原因还有，买进和卖出股票的次数总计达 396 次，手续费和印花税等原因也造成了相当的亏损；虽然印花税和

手续费比率较低，但是当交易次数达到一定频次之后以及交易金额大的情况下，这将是不容忽视的成本，然而往往算法交易的频次都要高于人工策略的频次。另外仅是预测第二天涨跌的可能对股价预测盈利本身问题描述也不太恰当，因此造成了重大的损失。另一方面，策略本身缺乏开仓策略和止损策略，每次交易都是买入和卖出最大的股票数量，导致了失误的下单交易无法弥补的极大亏损。

为了验证上述分析，我们除去所有手续费、印花税和滑点再此进行回测实验，假设每次交易都是成功的，并假设每次所有资金投入股票（不考虑每次由未买入的剩余资产储存在资金账户这种情况），资金曲线变化（如图 3.31 所示）资产曲线 B。这 4 年年化收益率为 8.85%，这四年除了偶尔亏损，四年的盈利率都是稳定的，净资产稳步上升，回测评估结果见表 3.17。

表 3.17 回测结果评估 2

回测评估指标	最大回撤率	收益率均值（天）	收益率方差	交易频率
值(%)	12	0.03	0.6	40.4(402 次)

只有在这种情况，算法交易才是有盈利的。由于市场的复杂和交易的不稳定性，不管算法的精确度有多高，建议量化投资者设置止损策略和开仓策略，这也将极大提高交易盈利的成功率，因此股市投资需谨慎。

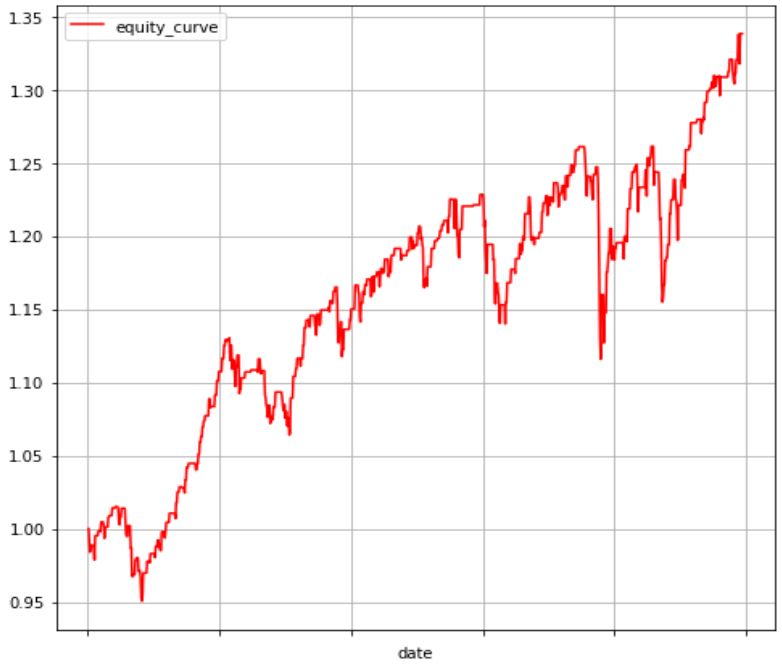

图 3.31 资产曲线 B

结　论

本文情感建模过程中，用的是显卡 1070Ti 加速训练。对于实证分析中的股票策略算法建模，用的是普通笔记本 CPU 训练的模型，本文实验的硬件条件见表 19，总训练时间为 240.27 秒，而且样本量非常小，仅在 995 个样本就可以达到 87.2%的准确率，所以实际应用能力较强，不需要很好的硬件条件支持即可完成模型的搭建和训练。

表 19 硬件条件

设备型号	ThinkPad x1 Carbon 5th Signature Edition
处理器	Intel(R) Core(TM) i7-7500 CPU @2.70GHz 2.90 GHz
已安装的 RAM	16.0GB
操作系统	Windows 10
系统类型	64 位操作系统，基于 x64 的处理器

通过 PCA 降维情感特征，可视化未来一天股票价格上涨和下跌的情感特征分布情况(见图 34)，初步得知，情感特征对于股价波动有一定影响。另外，通过无监督技术和有监督技术(基于 Attention-Based Bi-LSTM 的神经网络)提取新闻标题情感因子。利用 GBDT 算法进行交叉验证计算技术指标和情感特征的重要度。在通过基尼指数描述的特征贡献度中，构建的大部分情感特征的重要性要大于 Boll 指标和日均线指标，一定程度表明新闻头条的情感倾向会对股票价格波动会造成一定的影响。

通过对算法的评估，融合支持向量机、Logistic 和 GDBT 设计了一套集成算法模型，随机打乱数据划分训练集和测试集（5:5）训练模型，预测结果 AUC 分数高达 0.91，其中基线模型 SVM、Logistic、GBDT、Random Forest 得分最高（得分依次递减），通过 Blending 融合四个得分最高得基线模型训练前四年的股价历史数据，预测未来四年的股价，AUC 分数得分 0.95，准确率高达 87%。利用预测结果构建策略，合理设置初始资金、印花税、手续费和滑点，严格模拟中国股票市场 t+1 交易制度进行实盘模拟，涨停和跌停日都停止交易并以最大化的股票数量买入（总交易次数 328 次），回测的结果是最大回撤率 38%，总亏损 37.5%。算法预测精度高实盘盈利低的主要原因是手续费和印花税的成本，另外缺少止损策略和开仓策略，导致亏损不能及时平仓。为了验证分析结果，回测时除去手续费、印花税和滑点，为了尽可能确保策略的完全实施，假设每次交易都成功（总交易次数为 402 次），新的回测结果为最大回撤率 12%，年均收益率 8.85%，因此最总确定是手续费等成本造成了很大亏损。通过本文可得出结论，尽管算法交易的好处是不用人为干涉，不会受到人为心理因素干扰，并且预测精度高，但是一旦失误造成的风险是无法估量的。

参 考 文 献

[1] 理查德 A.德弗斯科, 丹尼斯 W.麦克利维. 量化投资分析[M]. 第三版 北京: 机械工业出版社, 2019.1：285-324

[2] 伊恩.古德费洛, 约书亚.本吉奥, 亚纶.库维尔. 深度学习[M]. 北京: 中信出版社，人民邮电出版社, 2017.8：201-244

[3] 郑泽宇, 顾思宇. TensorFlow 实战 Google 深度学习框架[M]. 北京: 中国工信出版集团, 电子工业出版社, 2017.3：199-227

[4] 涂铭, 刘祥, 刘树春. Python 自然语言处理实战核心技术和算法[M]. 北京: 机械工业出版社, 2018.8：85-105

[5] 约阿夫.戈尔德贝格. 基于深度学习的自然语言处理[M]. 北京: 机械工业出版社, 2018.12：119-124

[6] Ritter G. Machine Learning for Trading (August 8, 2017). Available at SSRN: https://ssrn.com /abstract =3015609 or http://dx.doi.org/10.2139/ssrn.3015609

[7] 张昊. 互联网财经新闻对股市影响效应的测度[D]. 浙江工商大学, 2015

[8] 朱昶胜. 基于 R 语言的网络舆情对股市影响研究[J]. 兰州理工大学学报, 2018,44(4):103-108

[9] 赵国顺. 基于时间序列分析的股票价格趋势预测研究[D]. 厦门大学, 2009

[10] 李航. 统计学习方法[M]. 北京: 清华大学出版社, 2012.3：95-134

[11] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning[M]. Second Edition Palo Alto: Springer, 2009.10：219-257

[12] Bengioy Y, Simardy P, Frasconiz P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE Transactions on Neural Network, 1994, 5(2):157-165

[13] Schuster M, Paliwal K K. Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing, 1997, 45(11): 2673-2681

[14] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Massachusetts Institute of Technology, 1997, 9(8): 1735-1780

[15] Srivastava N, Hinton G E, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of machine learning research, 2014, 5(1): 1929- 1958

[16] 任朝淦, 杨燕, 贾真, 唐慧佳, 喻琇瑛. 基于注意力机制的问句实体链接[J]. 模式识别与人工智能, 2018, 31(12):1127-1133

[17] 关鹏飞, 李宝安, 吕学强, 周建设. 注意力增强的双向 LSTM 情感分析[J]. 中文信息学报, 2019, 33(2):105-111

[18] 夏林旭, 刘茂福, 胡慧君. 基于 AM-BiLSTM 模型的情绪原因识别[J]. 武汉大学学报(理学版), 2019, 65(3):276-282

致　　谢

本项目其实耗时一年半左右，经历了模型设计、不断编程试验、调参的过程才得以完成。论文的完成得益于指导老师陈文娟还有其他老师对论文详细的批改和经验指导。前前后后也是不断得益于自己一些项目经历的总结和迁移，从而大大节约了本文程序调参的过程。本文借鉴的项目经历包括了 Kaggle 大数据平台激烈的国际竞赛，均得益于得益于杜传斌老师的指导，例如 Microsoft Malware Prediction·Competition，Kaggle Avito Demand Prediction Challenge 2018，Kaggle Santander Customer Transaction Prediction 2019 等总计获得两银三铜的成绩，本人也因此在 Kaggle 认证的 10 多万数据科学家中排名 1%左右，赢得 Kaggle Competition Experts 的荣誉。另外本文的算法方案设计和论文框架也得到了他宝贵的建议。回测实验和程序的设计和修正少不了孙钏锋老师的建议。还有这四年不管是技术还是思想的成长，不能没有张永祥老师和祁晓光老师大学四年的学业指导和陪伴以及大学期间各位辅导过我的老师。还有 MetLife China 总部(北京)的实习期间，上司 Cooper Kuo 对我实习期间的培训增强了我的 R 语言编程技术熟练度，很大程度帮助了我完成论文的实验设计。临近毕业，真的非常感激以上提到的所有人，都是我精神的支柱和灵魂的导师，希望以后的学术和职业生涯，不仅仅是研究能力和精神思想的，我都可以走向他们的高度。本文的结论和实验的实现主要通过 R 语言和 Python 实现，本文核心实验代码在本文附录中可以找到，完整代码存放在本人的 github 主页 https://github.com/chenxinye。

附录：程序代码

本文的代码实现主要是 R 语言和 Python，另外借助以 Tensorflow 为后端的 Keras API 对神经网络建模，本文的核心代码如下。

1. TF-IDF 计算

```
#TF-IDF
library(tcltk)
library(tidytext)
library(magrittr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(stringr)
library(scales)

data(stop_words)
Combined_News_DJIA <- read.csv("Combined_News_DJIA.csv")
head(Combined_News_DJIA)

tokens <- function(dfcol){
  #one token per document per row
  df_ = data.frame(line = 1:length(dfcol),text = dfcol)
  df_$text = as.character(df_$text)
  df_$text <- gsub("b'", "", df_$text)
  df_$text <- gsub("[0-9]", "", df_$text)
  df_ = df_%>% unnest_tokens(word, text)
  return(df_)
}

combine <- function(df1,df2,str1,str2){
  m_1 = mutate(df1,topic = str1)
  m_2 = mutate(df2,topic = str2)
  freq <- bind_rows(m_1,m_2)
  return(freq)
}

df <- Combined_News_DJIA
dt1 = df$Top1 %>% tokens() %>% anti_join(stop_words)
dt2 = df$Top2 %>% tokens() %>% anti_join(stop_words)
```

```
dt3 = df$Top3 %>% tokens() %>% anti_join(stop_words)
dt4 = df$Top4 %>% tokens() %>% anti_join(stop_words)
dt5 = df$Top5 %>% tokens() %>% anti_join(stop_words)
dt6 = df$Top6 %>% tokens() %>% anti_join(stop_words)
dt7 = df$Top7 %>% tokens() %>% anti_join(stop_words)
dt8 = df$Top8 %>% tokens() %>% anti_join(stop_words)
dt9 = df$Top9 %>% tokens() %>% anti_join(stop_words)
dt10 = df$Top10 %>% tokens() %>% anti_join(stop_words)
dt11 = df$Top11 %>% tokens() %>% anti_join(stop_words)
dt12 = df$Top12 %>% tokens() %>% anti_join(stop_words)
dt13 = df$Top13 %>% tokens() %>% anti_join(stop_words)
dt14 = df$Top14 %>% tokens() %>% anti_join(stop_words)
dt15 = df$Top15 %>% tokens() %>% anti_join(stop_words)
dt16 = df$Top16 %>% tokens() %>% anti_join(stop_words)
dt17 = df$Top17 %>% tokens() %>% anti_join(stop_words)
dt18 = df$Top18 %>% tokens() %>% anti_join(stop_words)
dt19 = df$Top19 %>% tokens() %>% anti_join(stop_words)
dt20 = df$Top20 %>% tokens() %>% anti_join(stop_words)
dt21 = df$Top21 %>% tokens() %>% anti_join(stop_words)
dt22 = df$Top22 %>% tokens() %>% anti_join(stop_words)
dt23 = df$Top23 %>% tokens() %>% anti_join(stop_words)
dt24 = df$Top24 %>% tokens() %>% anti_join(stop_words)
dt25 = df$Top25 %>% tokens() %>% anti_join(stop_words)

df_1 <- combine(dt1,dt2,'topic 1','topic 2')
df_2 <- combine(dt3,dt4,'topic 3','topic 4')
df_3 <- combine(dt5,dt6,'topic 5','topic 6')
df_4 <- combine(dt7,dt8,'topic 7','topic 8')
df_5 <- combine(dt9,dt10,'topic 9','topic 10')
df_6 <- combine(dt11,dt12,'topic 11','topic 12')
df_7 <- combine(dt13,dt14,'topic 13','topic 14')
df_8 <- combine(dt15,dt16,'topic 15','topic 16')
df_9 <- combine(dt17,dt18,'topic 17','topic 18')
df_10 <- combine(dt19,dt20,'topic 19','topic 20')
df_11 <- combine(dt21,dt22,'topic 21','topic 22')
df_12 <- combine(dt23,dt24,'topic 23','topic 24')
df_13 = mutate(dt25,topic = 'topic 25')

df = bind_rows(df_1,df_2)
df = bind_rows(df,df_3)
```

```
df = bind_rows(df,df_4)
df = bind_rows(df,df_5)
df = bind_rows(df,df_6)
df = bind_rows(df,df_7)
df = bind_rows(df,df_8)
df = bind_rows(df,df_9)
df = bind_rows(df,df_10)
df = bind_rows(df,df_11)
df = bind_rows(df,df_12)
df = bind_rows(df,df_13)

df_ = bind_rows(df_1,df_2)
df_ = bind_rows(df_,df_3)
df_ = bind_rows(df_,df_4)
df_ = bind_rows(df_,df_5)

return_words <- function(df){
   bw = df %>% count(topic,word,sort = T) %>% ungroup()
   tw = bw %>% group_by(topic) %>% summarize(total = sum(n))
   bc = left_join(bw,tw)
   print(bc)
   return(bc)}

bc <- return_words(df)

show <- function(bc){
   p <-ggplot(bc,aes(n/total,fill = topic)) + geom_histogram(show.legend = F) +
xlim(NA,0.0009) + facet_wrap(~topic,ncol = 2,scales = "free_y")
   print(p)
}

show(return_words(df_))

bc.tf_idf <- bc %>% bind_tf_idf(word,topic,n);bc.tf_idf%>%print()
print(bc.tf_idf[which(bc.tf_idf$idf != 0),])
tf <- bc.tf_idf %>% select(-total) %>% arrange(desc(tf_idf))
write.csv(tf,file = "count/tf_idf_count.csv")

bc.tf_idf_ <- return_words(df_) %>% bind_tf_idf(word,topic,n)
p <- bc.tf_idf_ %>% arrange(desc(tf_idf)) %>% mutate(word = factor(word,levels=
```

```
rev(unique(word)))) %>% group_by(topic) %>% top_n(5) %>% ungroup
p1 = p %>% ggplot(aes(word,tf_idf,fill=topic)) + geom_col(show.legend = F) + labs(x = NULL,y = 'tf-idf') + facet_wrap(~topic,ncol = 2,scales = "free") + coord_flip()
print(p1)
```

2. 情感分析

```
library(tcltk)
library(tidytext)
library(magrittr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(stringr)
library(scales)
library(reshape2)
library(wordcloud)

data(stop_words)
Combined_News_DJIA <- read.csv("Combined_News_DJIA.csv")
head(Combined_News_DJIA)

tokens <- function(dfcol){
  #one token per document per row
  df_ = data.frame(line = 1:length(dfcol),text = dfcol)
  df_$text = as.character(df_$text)
  df_$text <- gsub("b'", "", df_$text)
  df_$text <- gsub("[0-9]", "", df_$text)
  df_ = df_%>% unnest_tokens(word, text)
  return(df_)
}

combine <- function(df1,df2,str1,str2){
  m_1 = mutate(df1,topic = str1)
  m_2 = mutate(df2,topic = str2)
  freq <- bind_rows(m_1,m_2)
  return(freq)
}

df <- Combined_News_DJIA
```

```
dt1 = df$Top1 %>% tokens() %>% anti_join(stop_words)
dt2 = df$Top2 %>% tokens() %>% anti_join(stop_words)
dt3 = df$Top3 %>% tokens() %>% anti_join(stop_words)
dt4 = df$Top4 %>% tokens() %>% anti_join(stop_words)
dt5 = df$Top5 %>% tokens() %>% anti_join(stop_words)
dt6 = df$Top6 %>% tokens() %>% anti_join(stop_words)
dt7 = df$Top7 %>% tokens() %>% anti_join(stop_words)
dt8 = df$Top8 %>% tokens() %>% anti_join(stop_words)
dt9 = df$Top9 %>% tokens() %>% anti_join(stop_words)
dt10 = df$Top10 %>% tokens() %>% anti_join(stop_words)

emotion_w <- get_sentiments("bing")
df_1 <- combine(dt1,dt2,'topic 1','topic 2')
df_2 <- combine(dt3,dt4,'topic 3','topic 4')
df_3 <- combine(dt5,dt6,'topic 5','topic 6')
df_4 <- combine(dt7,dt8,'topic 7','topic 8')
df_5 <- combine(dt9,dt10,'topic 9','topic 10')

df = bind_rows(df_1,df_2)
df = bind_rows(df,df_3)
df = bind_rows(df,df_4)
df = bind_rows(df,df_5)

data1.2 <- df %>% inner_join(emotion_w)
#data1.2 <- data1.2%>% count(topic,index = line,sentiment) %>% spread(sentiment,n,fill =
0) %>% mutate(sentiment = 2*positive - negative)
#write.csv(data1.2,file = 'count/emotion_eval.csv')
data1.2 <- data1.2%>% count(topic,index = line,sentiment) %>% spread(sentiment,n,fill =
0) %>% mutate(sentiment = 2*positive - negative)
p <- ggplot(data1.2,aes(index,sentiment,fill = topic)) + geom_col(show.legend = F) +
facet_wrap(~topic,ncol = 2, scales = 'free_x')
print(p)

write.csv(data1.2,file = 'count/emotion_eval.csv')
dg = get_sentiments("bing") %>% count(sentiment)
print(paste("negative vs positive:",dg$n[1] /    dg$n[2]))

combine_topic <- function(Combined_News_DJIA,n){
   columns = names(Combined_News_DJIA)
   list_f = c()
```

```
  for (i in 1:length(Combined_News_DJIA[,1])){
    strl = ""
    for (col in columns[2:n]){
      strl = paste(strl,Combined_News_DJIA[col][i,], sep = "", collapse = NULL)
    }
    list_f = append(list_f,strl)
  }
  Combined_News_DJIA$combine_topic = list_f
  return(Combined_News_DJIA)
}

df_all <- combine_topic(Combined_News_DJIA,26)
df_all <- df_all$combine_topic %>% tokens() %>% anti_join(stop_words)
word_em_count <- df_all %>% inner_join(get_sentiments("bing")) %>%
count(word,sentiment,sort = T) %>% ungroup()

word_em_count %>% group_by(sentiment) %>% top_n(20) %>%
  ungroup() %>% mutate(word = reorder(word,n,fill=sentiment))%>%
  ggplot(aes(word,n,fill = sentiment)) + geom_col(show.legend = F) +
facet_wrap(~sentiment,scales = 'free_y') +
    labs(y = "contribution",x = NULL) + coord_flip()

df_all %>% inner_join(get_sentiments("bing")) %>% count(word,sentiment,sort = T) %>%
  acast(word~sentiment, value.var = "n",fill = 0) %>%
  comparison.cloud(colors=c("DarkRed","#FFA500"),max.words = 120,title.size = 1.5)

help(comparison.cloud)
```

3. LDA 主题建模

```
library(tcltk)
library(tidytext)
library(magrittr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(stringr)
library(scales)
library(reshape2)
library(wordcloud)
```

```
library(igraph)
library(ggraph)
library(widyr)
library(arules)
library(rattle)
library(arulesViz)
library(tm)
library(NLP)
library(tm)
library(topicmodels)

set.seed(200)
data(stop_words)
Combined_News_DJIA <- read.csv("Combined_News_DJIA.csv")

combine_topic <- function(Combined_News_DJIA,n){
  columns = names(Combined_News_DJIA)
  list_f = c()
  for (i in 1:length(Combined_News_DJIA[,1])){
    strl = ""
    for (col in columns[2:n]){
      strl = paste(strl,Combined_News_DJIA[col][i,], sep = "", collapse = NULL)
    }
    list_f = append(list_f,strl)}
  Combined_News_DJIA$combine_topic = list_f
  return(Combined_News_DJIA)}

tokens <- function(dfcol){
  #one token per document per row
  df_ = data.frame(line = 1:length(dfcol),text = dfcol)
  df_$text = as.character(df_$text)
  df_$text <- gsub("b'", "", df_$text)
  df_$text <- gsub("[0-9]", "", df_$text)
  df_ = df_%>% unnest_tokens(word, text)
  return(df_)}

DF = combine_topic(Combined_News_DJIA,26)
DF = DF$combine_topic %>% tokens() %>% anti_join(stop_words)
head(DF)
```

```
corpus <- Corpus(VectorSource(DF$word))
DTcorpus <- DocumentTermMatrix(corpus,control = list(wordLengths = c(1, Inf),bounds =
list(global = 5, Inf),removeNumbers = TRUE))

meancosine_caculate <- function(Documentmatrix){
  mean_similarity <- c();mean_similarity[1] = 1
  for(i in 2:20){
    control <- list(burnin = 500, iter = 3000, keep = 100)
    Gibbs <- LDA(Documentmatrix, k = i, method = "Gibbs", control = control)
    term <- terms(Gibbs, 50) ;word <- as.vector(term)    ;freq <- table(word) ;unique_word
<- names(freq)
    mat <- matrix(rep(0, i * length(unique_word)),nrow = i, ncol = length(unique_word))
    colnames(mat) <- unique_word
    for(k in 1:i){
      for(t in 1:50){mat[k, grep(term[t,k], unique_word)] <- mat[k, grep(term[t, k],
unique_word)] + 1}}
    p <- combn(c(1:i), 2);l <- ncol(p);top_similarity <- c()
    for(j in 1:l){
      x <- mat[p[, j][1], ];y <- mat[p[, j][2], ]
      top_similarity[j] <- sum(x * y) / sqrt(sum(x^2) * sum(y ^ 2))}
    mean_similarity[i] <- sum(top_similarity) / l;message("top_num ", i)}
  return(mean_similarity)}

#cosine.similarity <- meancosine_caculate(DTcorpus)

#par(mfrow = c(1, 1))
#plot(cosine.similarity, type = "l")

gibbs <- LDA(DTcorpus, k = 6, method = "Gibbs", control = list(burnin = 500, iter = 1000,
keep = 100))

termsl <- terms(gibbs, 50)

write.csv(termsl, "count/ldaterms.csv", row.names = FALSE)

plot_LDA <- function(terms,data = pdata.freq){
  len1 <- length(terms[,1]);len2 <- length(terms[1,])
  vec <- vector(mode = "logical",length = len2);vec[1] = 0.0
  for (i in 1:len2){count <- 0.0
  for (j in 1:len1){
```

```
      freq <- data[which(data$Var1 == terms[j,i]),c("Freq")]
      count <- count + freq}
   count %>% print();vec[i] <- count}
   dataf <- data.frame(vec = vec,topic = as.character(1:len2))
   myLabel = as.vector(dataf$topic)
   myLabel = paste("topic:",myLabel, "(", round(dataf$vec / sum(dataf$vec) * 100, 2), "%)",
sep = "")

   p = ggplot(dataf, aes(x = "", y = vec, fill = factor(topic))) +
      geom_bar(stat = "identity", width = 1) +
      coord_polar(theta = "y") +
      labs(x = "", y = "", title = "") +
      theme(axis.ticks = element_blank()) +
      theme(legend.title = element_blank(), legend.position = "top") +
      scale_fill_discrete(breaks = dataf$topic, labels = myLabel)
   print(p)}

picture_output <- function(pos_cos) {
   cosdf <- data.frame(x = 1:length(pos_cos),meancosine = pos_cos,x = rep("topic",10))
   p <- ggplot(cosdf,aes(x= x,y= meancosine))
   p <- p + stat_smooth(se = TRUE) + geom_point();print(p)}

freq <- DF %>% count(word,sort = T)
freq$n <- freq$n/sum(freq$n)
names(freq) = c("Var1","Freq")

plot_LDA(termsl,freq)
#picture_output(cosine.similarity)
```

4. 关联规则挖掘

```
library(tcltk)
library(tidytext)
library(magrittr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(stringr)
library(scales)
library(reshape2)
```

```
library(wordcloud)
library(igraph)
library(ggraph)
library(widyr)
library(arules)
library(rattle)
library(arulesViz)
set.seed(200)

data(stop_words)
Combined_News_DJIA <- read.csv("Combined_News_DJIA.csv")

combine_topic <- function(Combined_News_DJIA,n){
  columns = names(Combined_News_DJIA)
  list_f = c()
  for (i in 1:length(Combined_News_DJIA[,1])){
    strl = ""
    for (col in columns[2:n]){
      strl = paste(strl,Combined_News_DJIA[col][i,], sep = "", collapse = NULL)
    }
    list_f = append(list_f,strl)}
  Combined_News_DJIA$combine_topic = list_f
  return(Combined_News_DJIA)}

tokens <- function(dfcol){
  #one token per document per row
  df_ = data.frame(line = 1:length(dfcol),text = dfcol)
  df_$text = as.character(df_$text)
  df_$text <- gsub("b'", "", df_$text)
  df_$text <- gsub("[0-9]", "", df_$text)
  df_ = df_%>% unnest_tokens(word, text)
  return(df_)}

DF = combine_topic(Combined_News_DJIA,26)
DF = DF$combine_topic %>% tokens() %>% anti_join(stop_words)
head(DF)

WR = DF %>% group_by(word) %>% filter(n() >= 20) %>% pairwise_cor(word,line,sort
= T)
d = WR[which(WR$correlation <1),]
```

```
write.csv(d,file = "corr.csv")

top_word <- c("government","people","police","world","war","israel","u.s","killed","president")

d %>% filter(item1 %in% top_word) %>% group_by(item1) %>% top_n(6) %>% mutate(item2 = reorder(item2,correlation)) %>%
  ggplot(aes(item2,correlation)) + geom_bar(stat = "identity") + facet_wrap(~ item1,scales = "free") + coord_flip()

#d %>% filter(correlation > 0.8) %>% graph_from_data_frame() %>% ggraph(layout = "auto") +
 # geom_edge_link(aes(edge_alpha =correlation),show.legend = F) + geom_node_point(color = "lightblue",size = 5) +
  #geom_node_text(aes(label = name),repel = T) + theme_void()

return_summary <- function(comment_word,sup = 0.1,con = 0.1,boole = T){
  table_aprori <- as(split(comment_word$word,comment_word$line),"transactions")
  model_build <- apriori(table_aprori,parameter = list(support = sup,confidence=con))
  output <- inspect(sort(model_build,by = "confidence"));summary(model_build)
  output <- output[-which(output$lhs == "{}"),];head(output,30)
  if (boole == T){return (output)
  }else{return (model_build)}}

sc = return_summary(DF) # calculate support and confidence

return_writer <- function(df,string){
  write.csv(df, string, row.names = FALSE)}

return_writer(sc,"count/support and confidence.csv")

sc_ = return_summary(DF,sup = 0.1,con = 0.1,boole = F) # calculate support and confidence
output_neg <- head(sort(sc_,by = "confidence"),30)
plot(output_neg,method = "grouped",measure = "lift",shading ="support")
```

5. 基于 Attention-Based Bi-LSTM 模型预测情感分数

```
import numpy as np
import pandas as pd
import os
```

```
from tqdm import tqdm
tqdm.pandas()
print(os.listdir("../input"))
print(os.listdir("../input/text-combine"))

TEXT_COL = 'comment_text'
EMB_PATH = '../input/fasttext-crawl-300d-2m/crawl-300d-2M.vec'
train = pd.read_csv('../input/jigsaw-unintended-bias-in-toxicity-classification/train.csv', index_col='id')
test = pd.read_csv('../input/jigsaw-unintended-bias-in-toxicity-classification/test.csv', index_col='id')

def get_coefs(word,*arr): return word, np.asarray(arr, dtype='float32')

def load_embeddings(embed_dir=EMB_PATH):
    embedding_index = dict(get_coefs(*o.strip().split(" ")) for o in tqdm(open(embed_dir)))
    return embedding_index

def build_embedding_matrix(word_index, embeddings_index, max_features, lower = True, verbose = True):
    embedding_matrix = np.zeros((max_features, 300))
    for word, i in tqdm(word_index.items(),disable = not verbose):
        if lower:
            word = word.lower()
        if i >= max_features: continue
        try:
            embedding_vector = embeddings_index[word]
        except:
            embedding_vector = embeddings_index["unknown"]
        if embedding_vector is not None:
            # words not found in embedding index will be all-zeros.
            embedding_matrix[i] = embedding_vector
    return embedding_matrix

def build_matrix(word_index, embeddings_index):
    embedding_matrix = np.zeros((len(word_index) + 1,300))
    for word, i in word_index.items():
        try:
            embedding_matrix[i] = embeddings_index[word]
        except:
```

```
                embedding_matrix[i] = embeddings_index["unknown"]
    return embedding_matrix

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import gc

maxlen = 220
max_features = 100000
embed_size = 300
tokenizer = Tokenizer(num_words=max_features, lower=True) #filters = ''
#tokenizer = text.Tokenizer(num_words=max_features)
print('fitting tokenizer')
tokenizer.fit_on_texts(list(train[TEXT_COL]) + list(test[TEXT_COL]))
word_index = tokenizer.word_index
X_train = tokenizer.texts_to_sequences(list(train[TEXT_COL]))
y_train = train['target'].values
X_test = tokenizer.texts_to_sequences(list(test[TEXT_COL]))

X_train = pad_sequences(X_train, maxlen=maxlen)
X_test = pad_sequences(X_test, maxlen=maxlen)

gc.collect()

embeddings_index = load_embeddings()
embedding_matrix = build_matrix(word_index, embeddings_index)
del embeddings_index
gc.collect()

from keras import backend as K
from keras.engine.topology import Layer
from keras import initializers, regularizers, constraints, optimizers, layers

class Attention(Layer):
    def __init__(self, step_dim,
                 W_regularizer=None, b_regularizer=None,
                 W_constraint=None, b_constraint=None,
                 bias=True, **kwargs):
        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')
```

```
        self.W_regularizer = regularizers.get(W_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)

        self.W_constraint = constraints.get(W_constraint)
        self.b_constraint = constraints.get(b_constraint)

        self.bias = bias
        self.step_dim = step_dim
        self.features_dim = 0
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3

        self.W = self.add_weight((input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        self.features_dim = input_shape[-1]

        if self.bias:
            self.b = self.add_weight((input_shape[1],),
                                     initializer='zero',
                                     name='{}_b'.format(self.name),
                                     regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)
        else:
            self.b = None

        self.built = True

    def compute_mask(self, input, input_mask=None):
        return None

    def call(self, x, mask=None):
        features_dim = self.features_dim
        step_dim = self.step_dim
```

```
            eij = K.reshape(K.dot(K.reshape(x, (-1, features_dim)),
                                  K.reshape(self.W, (features_dim, 1))), (-1, step_dim))

            if self.bias:
                eij += self.b

            eij = K.tanh(eij)

            a = K.exp(eij)

            if mask is not None:
                a *= K.cast(mask, K.floatx())

            a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(), K.floatx())

            a = K.expand_dims(a)
            weighted_input = x * a
            return K.sum(weighted_input, axis=1)

        def compute_output_shape(self, input_shape):
            return input_shape[0],    self.features_dim

import keras.layers as L
from keras.models import Model
from keras.optimizers import Adam

def build_model(verbose = False, compile = True):
        sequence_input = L.Input(shape=(maxlen,), dtype='int32')
        embedding_layer = L.Embedding(len(word_index) + 1,
                                      300,
                                      weights=[embedding_matrix],
                                      input_length=maxlen,
                                      trainable=False)
        x = embedding_layer(sequence_input)
        x = L.SpatialDropout1D(0.2)(x)
        x = L.Bidirectional(L.CuDNNLSTM(64, return_sequences=True))(x)

        att = Attention(maxlen)(x)
        avg_pool1 = L.GlobalAveragePooling1D()(x)
        max_pool1 = L.GlobalMaxPooling1D()(x)
```

```
    x = L.concatenate([att,avg_pool1, max_pool1])

    preds = L.Dense(1, activation='sigmoid')(x)


    model = Model(sequence_input, preds)
    if verbose:
        model.summary()
    if compile:

model.compile(loss='binary_crossentropy',optimizer=Adam(0.005),metrics=['acc'])
    return model

from sklearn.model_selection import KFold
splits = list(KFold(n_splits=5).split(X_train,y_train))

from keras.callbacks import EarlyStopping, ModelCheckpoint
import keras.backend as K
import numpy as np
BATCH_SIZE = 2048
NUM_EPOCHS = 100

oof_preds = np.zeros((X_train.shape[0]))
test_preds = np.zeros((X_test.shape[0]))
for fold in [0,1,2,3,4]:
    K.clear_session()
    tr_ind, val_ind = splits[fold]
    ckpt = ModelCheckpoint(f'gru_{fold}.hdf5', save_best_only = True)
    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
    model = build_model()
    history_ = model.fit(X_train[tr_ind],
        y_train[tr_ind]>0.5,
        batch_size=BATCH_SIZE,
        epochs=NUM_EPOCHS,
        validation_data=(X_train[val_ind], y_train[val_ind]>0.5),
        callbacks = [es,ckpt])

    oof_preds[val_ind] += model.predict(X_train[val_ind])[:,0]
    test_preds += model.predict(X_test)[:,0]
```

```
test_preds /= 5

from sklearn.metrics import roc_auc_score
print(roc_auc_score(y_train>0.5,oof_preds))

import matplotlib.pyplot as plt
plt.plot(history_.history['loss'])
plt.plot(history_.history['val_loss'])
plt.plot(history_.history['acc'])
plt.plot(history_.history['val_acc'])
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(['loss', 'val_loss','acc','val_acc'], loc='upper')
plt.ylim(0,0.5)
plt.show()

def to_vec(data):
    data = tokenizer.texts_to_sequences(data)
    data = pad_sequences(data, maxlen=220)
    return(data)

_traincol = ['target', 'severe_toxicity', 'obscene', 'identity_attack', 'insult',
'threat','sexual_explicit']
text = pd.read_csv("../input/text-combine/text_combine.csv")
textvec = to_vec(text["combine_topic"])
predictions_text = model.predict(textvec, batch_size=1048)[1]

j = 0
for i in _traincol:
    text[i] = predictions_text[:,j]
    j += 1

text.to_csv('text_test.csv', index=False)

submission = pd.read_csv('../input/jigsaw-unintended-bias-in-toxicity-
classification/sample_submission.csv', index_col='id')
submission['prediction'] = test_preds
submission.reset_index(drop=False, inplace=True)

submission.to_csv('submission.csv', index=False)
```

6. 特征工程(特征规约和计算特征重要度)

```
#this lgbm is for the whole dataset
import pandas as pd
import numpy as np
from sklearn.metrics import roc_auc_score
import lightgbm as lgb
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import tree
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, AdaBoostClassifier
import xgboost as xgb
from sklearn.model_selection import train_test_split

em = pd.read_csv("emotion_ev\emotion_evaluation.csv")
em.columns = ['index', 'negative', 'positive', 'count', 'sentiment']
em_col = ["negative","positive","count","sentiment"]
traincol = ['sexual_explicit', 'severe_toxicity', 'obscene', 'identity_attack', 'insult', 'threat']

text = pd.read_csv("Combined_News_DJIA.csv")
text_em = pd.read_csv(r"text\text_test.csv")[traincol]
df_n = pd.read_csv(r"text\df_numeric.csv")
target = text[['Label']]

uls_train_col = ['date','open', 'high', 'low','rate_of_return','rate_of_return_change', 'rate_of_return_change_shift','change']

index_ = ['close',
      'cr','cr-ma1','cr-ma2','cr-ma3',
      'kdjk','kdjd','kdjj',
      'close_5_sma','close_10_sma',
      'macd','macds','macdh',
      'boll','boll_ub','boll_lb',
      'rsi_6','rsi_12',
```

```
        'wr_10','wr_6',
        'cci','cci_20',
        'tr','atr',
        'dma',
        'trix','trix_9_sma',
        'vr','vr_6_sma'
        ]

param1 = {
        'bagging_freq': 30,
        'bagging_fraction': 0.9,
        'boost_from_average':'false',
        'boost': 'gbdt',
        'feature_fraction': 0.91,
        'learning_rate': 0.01,
        'max_depth': -1,
        'metric':'auc',
        'min_data_in_leaf': 90,
        'min_sum_hessian_in_leaf': 15.0,
        'num_leaves': 27,
        'num_threads': 20,
        'tree_learner': 'serial',
        'objective': 'binary',
        'verbosity': -1
}

folds = StratifiedKFold(n_splits=5, shuffle=False, random_state=859)

df_nm = pd.concat([df_n[index_],em[em_col],text_em,text_em[traincol]],axis = 1)
oof = np.zeros(len(df_nm))
feature_importance_df = pd.DataFrame()

for fold_, (trn_idx, val_idx) in enumerate(folds.split(df_nm.values, target.values)):

        X_train, y_train = df_nm.iloc[trn_idx], target.iloc[trn_idx]
        X_valid, y_valid = df_nm.iloc[val_idx], target.iloc[val_idx]

        X_tr, y_tr = X_train, y_train
        X_tr = pd.DataFrame(X_tr)
```

```
        #print("Fold idx:{}".format(fold_ + 1))
        trn_data = lgb.Dataset(X_tr, label=y_tr)
        val_data = lgb.Dataset(X_valid, label=y_valid)

        clf = lgb.train(param1, trn_data, 2000, valid_sets = [val_data], verbose_eval=100000,
early_stopping_rounds = 200)
        oof[val_idx] = clf.predict(df_nm .iloc[val_idx],num_iteration=clf.best_iteration)
        #print("CV score: {:<8.5f}".format(roc_auc_score(target.loc[val_idx], oof[val_idx])))

        fold_importance_df = pd.DataFrame()
        fold_importance_df["feature"] = df_nm .columns
        fold_importance_df["importance"] = clf.feature_importance(importance_type='gain')
        fold_importance_df["fold"] = fold_ + 1
        feature_importance_df = pd.concat([feature_importance_df, fold_importance_df],
axis=0)

print("CV score: {:<8.5f}".format(roc_auc_score(target, oof)))
cols = (feature_importance_df[["feature",
"importance"]].groupby("feature").mean().sort_values(by="importance",
ascending=False)[:1000].index)
best_features = feature_importance_df.loc[feature_importance_df.feature.isin(cols)]
plt.figure(figsize=(10,10))
featurescore =
best_features.groupby("feature").mean().sort_values(by="importance",ascending=False).re
set_index()
sns.barplot(x="importance", y="feature", data=featurescore)
plt.title('GDBT Features (averaged over folds)')
plt.tight_layout()
plt.show()
```

7. 算法融合方案和策略信号产生

```
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.kernel_approximation import Nystroem
from sklearn.kernel_approximation import RBFSampler
```

```
from sklearn.pipeline import make_pipeline
import pydotplus    # you can install pydotplus with: pip install pydotplus
from IPython.display import Image
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import numpy as np
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn import tree
from sklearn import svm
import time
import warnings
warnings.filterwarnings('ignore')

SEED = 2019
df_nm = np.load("df_nm.npy")
target = np.load("target.npy")

xtrain, xtest, ytrain, ytest = train_test_split(df_nm,target,test_size=0.5,random_state=SEED)

def get_models():
    """Generate a library of base learners."""
    svc = svm.SVC(kernel='linear',probability = True)
    nn = MLPClassifier((40, 20), early_stopping=False, random_state=SEED)
    dt = DecisionTreeClassifier(max_depth=35, random_state=SEED)
    lr = LogisticRegression(C=90, random_state=SEED)
    rf = RandomForestClassifier(n_estimators=300, max_features=30,
random_state=SEED)
    xgm = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                    colsample_bytree=0.65, gamma=2, learning_rate=0.01,
max_delta_step=1,
                    max_depth=30, min_child_weight=2, missing=None, n_estimators=200,
                    n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=SEED,
                    silent=True, subsample=1)
```

```
    models = {'SVM':svc,
              'Neural Network':nn,
              'decisiontree':dt,
              'random forest': rf,
              'xgboost': xgm,
              'logistic': lr,
              }

    return models


def train_predict(model_list):
    """Fit models in list on training set and return preds"""
    P = np.zeros((ytest.shape[0], len(model_list)))
    P = pd.DataFrame(P)

    print("Fitting models.")
    cols = list()
    for i, (name, m) in enumerate(models.items()):
        print("%s..." % name, end=" ", flush=False)
        m.fit(xtrain, ytrain)
        P.iloc[:, i] = m.predict_proba(xtest)[:, 1]
        cols.append(name)
        print("done")

    P.columns = cols
    print("Done.\n")
    return P


def score_models(P, y):
    """Score model in prediction DF"""
    print("Scoring models.")
    for m in P.columns:
        score = roc_auc_score(y, P.loc[:, m])
        print("%-26s: %.3f" % (m, score))
    print("Done.\n")

models = get_models()
P = train_predict(models)
```

```
score_models(P, ytest)

# You need ML-Ensemble for this figure: you can install it with: pip install mlens
from mlens.visualization import corrmat

corrmat(P.corr(), inflate=False)
plt.show()


print("Ensemble ROC-AUC score: %.3f" % roc_auc_score(ytest, P.mean(axis=1)))

from sklearn.metrics import roc_curve

def plot_roc_curve(ytest, P_base_learners, P_ensemble, labels, ens_label):
    """Plot the roc curve for base learners and ensemble."""
    plt.figure(figsize=(10, 8))
    plt.plot([0, 1], [0, 1], 'k--')

    cm = [plt.cm.rainbow(i)
       for i in np.linspace(0, 1.0, P_base_learners.shape[1] + 1)]

    for i in range(P_base_learners.shape[1]):
        p = P_base_learners[:, i]
        fpr, tpr, _ = roc_curve(ytest, p)
        plt.plot(fpr, tpr, label=labels[i], c=cm[i + 1])

    fpr, tpr, _ = roc_curve(ytest, P_ensemble)
    plt.plot(fpr, tpr, label=ens_label, c=cm[0])

    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(frameon=False)
    plt.show()

class model_stacking:
    def __init__(self,train,target):
        self.train = train
        self.target = target
        self.x_train,    self.x_test,    self.y_train,    self.y_test    =
```

```
train_test_split(train,target,test_size=0.5,random_state=SEED)
            self.tr_pred,self.testpred,self.y_train = self.train_model()
            self.pred = self.stacking_train()

        def model_blend(self):
            svc = svm.SVC(kernel='linear',probability = True)
            nn = MLPClassifier((40, 20), early_stopping=False, random_state=SEED)
            dt = DecisionTreeClassifier(max_depth=35, random_state=SEED)
            lr = LogisticRegression(C=90, random_state=SEED)
            rf = RandomForestClassifier(n_estimators=300, max_features=30,
random_state=SEED)
            xgm = xgb.XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1,
                                    colsample_bytree=0.65, gamma=2, learning_rate=0.01,
max_delta_step=1,
                                    max_depth=30, min_child_weight=2, missing=None,
n_estimators=200,
                                    n_jobs=1, nthread=None, objective='binary:logistic',
random_state=0,
                                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=SEED,
                                    silent=True, subsample=1)

            model_list = [svc,nn,dt,lr,rf,xgm]

            return model_list

        def train_model(self):
            model_list = self.model_blend()
            print("begin")
            x_train,y_train = self.x_train,self.y_train
            for model in model_list:
                print(model)
                if model == model_list[0]:
                    model.fit(x_train,y_train)
                    pred = model.predict_proba(x_train)[:,1].reshape(-1,1)
                    testpred = model.predict_proba(self.x_test.values)[:,1].reshape(-1,1)
                else:
                    model.fit(x_train,y_train)
                    pre = model.predict_proba(x_train)[:,1]
                    pred = np.hstack((pred,pre.reshape(-1,1)))
```

```
                testpre = model.predict_proba(self.x_test.values)[:,1]
                testpred = np.hstack((testpred,testpre.reshape(-1,1)))
        print("done!")

        return (pred,testpred,y_train)

    def stacking_train(self):
        mnb = MultinomialNB()
        #log = LogisticRegression(C=100, random_state=42)
        mnb.fit(self.tr_pred,self.y_train)
        #log.fit(tr_pred,y_train)
        test_pred = mnb.predict_proba(self.testpred)[:,1]
        print("stacking AUC:",roc_auc_score(self.y_test,test_pred))
        return test_pred

def log_meta(self):
    log = LogisticRegression(C=100, random_state=42)
    log.fit(self.tr_pred,self.y_train)
    test_pred = log.predict_proba(self.testpred)[:,1]
    print("stacking AUC:",roc_auc_score(self.y_test,test_pred))
    return test_pred

def xgb_meta(self):
    xgm = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                    colsample_bytree=0.65, gamma=2, learning_rate=0.01,
max_delta_step=1,
                    max_depth=4, min_child_weight=2, missing=None,
n_estimators=200,
                    n_jobs=1, nthread=None, objective='binary:logistic',
random_state=0,
                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=SEED,
                    silent=True, subsample=1)
    xgm.fit(self.tr_pred,self.y_train)
    test_pred = xgm.predict_proba(self.testpred)[:,1]
    print("stacking AUC:",roc_auc_score(self.y_test,test_pred))
    return test_pred

pipeline_stacking = model_stacking(df_nm,target)
print("gau model:")
p = pipeline_stacking.stacking_train()
```

```
print("logistic model:")
p1 = log_meta(pipeline_stacking)
print("xgb model:")
p2 = xgb_meta(pipeline_stacking)
plot_roc_curve(ytest, P.values, P.mean(axis=1), list(P.columns), "ensemble")

#==================
#产生信号
#==================
x_train, x_test, y_train, y_test = df_nm[0:int(0.5*len(df_nm))],df_nm[int(0.5*len(df_nm)):],target[0:int(0.5*len(df_nm))],target[int(0.5*len(df_nm)):]


def final_models():
    """Generate a library of base learners."""
    svc = svm.SVC(kernel='linear',probability = True)
    lr = LogisticRegression(C=90, random_state=SEED)
    rf = RandomForestClassifier(n_estimators=300, max_features=30, random_state=SEED)
    xgm = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                    colsample_bytree=0.65, gamma=2, learning_rate=0.01, max_delta_step=1,
                    max_depth=25, min_child_weight=2, missing=None, n_estimators=500,
                    n_jobs=1, nthread=None, objective='binary:logistic', random_state=SEED,
                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=SEED,
                    silent=True, subsample=1)

    models = {'SVM':svc,
                'random forest': rf,
                'xgboost': xgm,
                'logistic': lr,
                }

    return models

def final_train_predict(model_list):
    """Fit models in list on training set and return preds"""
    P = np.zeros((y_test.shape[0], len(model_list)))
```

```
    P = pd.DataFrame(P)

    print("Fitting models.")
    cols = list()
    for i, (name, m) in enumerate(model_list.items()):
        print("%s..." % name, end=" ", flush=False)
        m.fit(x_train, y_train)
        P.iloc[:, i] = m.predict_proba(x_test)[:, 1]
        cols.append(name)
        print("done")

    P.columns = cols
    print("Done.\n")
    return P


def score_models(P, y):
    """Score model in prediction DF"""
    print("Scoring models.")
    for m in P.columns:
        score = roc_auc_score(y, P.loc[:, m])
        print("%-26s: %.3f" % (m, score))
    print("Done.\n")

start = time.time()
final_model = final_models()
fP = final_train_predict(final_model)
score_models(fP, y_test)
end = time.time()
print("Ensemble ROC-AUC score: %.3f" % roc_auc_score(y_test, fP.mean(axis=1)))

print('time consumption:',end - start)

df_final_result = pd.read_csv("text\df_numeric.csv")[int(0.5*len(df_nm)):].reset_index()

df_final_result['finalsig'] = fP.mean(axis=1)

print("evaluation:\n", classification_report(y_test,np.round(fP.mean(axis=1))))

df_final_result.to_csv("src\output\signal.csv",index = 0)
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(np.array(y_test),np.round(fP.mean(axis=1)))
from sklearn.metrics import accuracy_score
accuracy_score(np.array(y_test),np.round(fP.mean(axis=1)))

def search_best(yt,yp):
    acc = accuracy_score(yt,np.round(yp))
    bst = 0.5
    for i in np.arange(0,1,0.01):
        y = yp.copy()
        y[y>i] = 1
        y[y<=i] = 0
        acc_ = accuracy_score(yt,y)
        print(acc_)
        if acc_> acc:
            bst = i
    yp[yp>bst] = 1
    yp[yp<bst] = 0
    return bst,yp

bst,y = search_best(y_test,fP.mean(axis=1))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(np.array(y_test),y))
from sklearn.metrics import accuracy_score
print(accuracy_score(np.array(y_test),y))
```

8. 策略信号调整

```
import pandas as pd
from decimal import Decimal, ROUND_HALF_UP
pd.set_option('expand_frame_repr', False)  # 当列太多时不换行
pd.set_option('display.max_rows', 5000)  # 最多显示数据的行数

# =====读入股票数据 output
df = pd.read_csv("output/signal.csv")
# 任何原始数据读入都进行一下排序、去重，以防万一
df.sort_values(by=['date'], inplace=True)
```

```
df.drop_duplicates(subset=['date'], inplace=True)
df.reset_index(inplace=True, drop=True)

# =====计算后复权价
df['最高价'] = df['high']
df['最低价'] = df['low']
df['收盘价'] = df['close']
df['开盘价'] = df['open']
df['前收盘价'] = df['收盘价'].shift(-1)
df['涨跌幅'] = df['收盘价'] / df['前收盘价'] - 1
df['复权因子'] = (1 + df['涨跌幅']).cumprod()
df['收盘价_复权'] = df['复权因子'] * (df.iloc[0]['收盘价'] / df.iloc[0]['复权因子'])
df['开盘价_复权'] = df['开盘价'] / df['收盘价'] * df['收盘价_复权']
df['最高价_复权'] = df['最高价'] / df['收盘价'] * df['收盘价_复权']
df['最低价_复权'] = df['最低价'] / df['收盘价'] * df['收盘价_复权']
df.drop(['复权因子'], axis=1, inplace=True)


# =====计算涨跌停价格
df['涨停价'] = df['前收盘价'] * 1.1
df['跌停价'] = df['前收盘价'] * 0.9
#  四舍五入
# print(round(3.5), round(4.5))   # 银行家舍入法：四舍六进，五，奇进偶不进
df['涨停价'] = df['涨停价'].apply(lambda x: float(Decimal(x*100).quantize(Decimal('1'),
rounding=ROUND_HALF_UP) / 100))
df['跌停价'] = df['跌停价'].apply(lambda x: float(Decimal(x*100).quantize(Decimal('1'),
rounding=ROUND_HALF_UP) / 100))

df['signal'] = df['finalsig']
df.loc[df.signal > 0.5,'signal'] = 1
df.loc[df.signal <= 0.5,'signal'] = 0
df.to_hdf('signals.h5', key='df', mode='w')
```

9. 产生实际持仓

```
import pandas as pd
pd.set_option('expand_frame_repr', False)
pd.set_option('display.max_rows', 5000)
```

```
# ===导入数据
df = pd.read_hdf('signals.h5', key='df')

# ===由 signal 计算出实际的每天持有仓位
# 在产生 signal 的 k 线结束的时候，进行买入
df['signal'].fillna(method='ffill', inplace=True)
df['signal'].fillna(value=0, inplace=True)  # 将初始行数的 signal 补全为 0
df['pos'] = df['signal'].shift()
df['pos'].fillna(value=0, inplace=True)  # 将初始行数的 pos 补全为 0

# ===对涨跌停无法买卖做出相关处理。
# 找出收盘价无法买入的 K 线
cannot_buy_condition = df['收盘价'] >= df['涨停价']
# 将找出上一周期无法买入的 K 线、并且 signal 为 1 时，的'pos'设置为空值
df.loc[cannot_buy_condition.shift() & (df['signal'].shift() == 1), 'pos'] = None  # 2010-12-22

# 找出收盘价无法卖出的 K 线
cannot_sell_condition = df['收盘价'] <= df['跌停价']
# 将找出上一周期无法卖出的 K 线、并且 signal 为 0 时的'pos'设置为空值
df.loc[cannot_sell_condition.shift() & (df['signal'].shift() == 0), 'pos'] = None

# pos 为空的时，不能买卖，只能和前一周期保持一致。
df['pos'].fillna(method='ffill', inplace=True)

df.drop(['signal'], axis=1, inplace=True)

df.to_hdf('pos.h5', key='df', mode='w')
```

10. 计算资金曲线：模拟真实市场回测

```
import pandas as pd
import numpy as np

df = pd.read_hdf('pos.h5', key='df')


#df = df[df['date'] >= pd.to_datetime('20070101')]
```

```
#开仓、平仓条件
condition1 = df['pos'] != 0
condition2 = df['pos'] != df['pos'].shift(1)
open_pos_condition = condition1 & condition2

condition1 = df['pos'] != 0
condition2 = df['pos'] != df['pos'].shift(-1)
close_pos_condition = condition1 & condition2

df.loc[open_pos_condition, 'start_time'] = df['date']
df['start_time'].fillna(method='ffill', inplace=True)
df.loc[df['pos'] == 0, 'start_time'] = pd.NaT


# 计算资金曲线
# ===基本参数
initial_cash = 100000000  # 初始资金，默认为 1000000 元
slippage = 0.01  # 滑点，股票默认为 0.01 元
c_rate = 2.5 / 10000  # 手续费，默认为万分之 2.5
t_rate = 1.0 / 1000  # 印花税，默认为千分之 1

# ===在买入的 K 线
# 在发出信号的当根 K 线以收盘价买入
df.loc[open_pos_condition, 'stock_num'] = initial_cash * (1 - c_rate) / (df['前收盘价'] +
slippage)

# 实际买入股票数量
df['stock_num'] = np.floor(df['stock_num'] / 100.0) * 100

# 买入股票之后剩余的钱，扣除了手续费
df['cash'] = initial_cash - df['stock_num'] * (df['前收盘价'] + slippage) * (1 + c_rate)

# 收盘时的股票净值
df['stock_value'] = df['stock_num'] * df['收盘价']

# ===在买入之后的 K 线
# 买入之后现金不再发生变动
```

```
df['cash'].fillna(method='ffill', inplace=True)
df.loc[df['pos'] == 0, ['cash']] = None

# 股票净值随着涨跌幅波动
group_num = len(df.groupby('start_time'))
if group_num > 1:
    t = df.groupby('start_time').apply(lambda x: x['收盘价_复权'] / x.iloc[0]['收盘价_复权'] * x.iloc[0]['stock_value'])
    t = t.reset_index(level=[0])
    df['stock_value'] = t['收盘价_复权']
elif group_num == 1:
    t = df.groupby('start_time')[['收盘价_复权', 'stock_value']].apply(lambda x: x['收盘价_复权'] / x.iloc[0]['收盘价_复权'] * x.iloc[0]['stock_value'])
    df['stock_value'] = t.T.iloc[:, 0]

# ===在卖出之后的 K 线
# 股票数量变动
df.loc[close_pos_condition, 'stock_num'] = df['stock_value'] / df['收盘价']

# 现金变动
df.loc[close_pos_condition, 'cash'] += df.loc[close_pos_condition, 'stock_num'] * (df['收盘价'] - slippage) * (1 - c_rate - t_rate)

# 股票价值变动
df.loc[close_pos_condition, 'stock_value'] = 0

# ===账户净值
df['net_value'] = df['stock_value'] + df['cash']

# ===计算资金曲线
df['equity_change'] = df['net_value'].pct_change(fill_method=None)
df.loc[open_pos_condition, 'equity_change'] = df.loc[open_pos_condition, 'net_value'] / initial_cash - 1  # 开仓日的收益率
df['equity_change'].fillna(value=0, inplace=True)
df['equity_curve'] = (1 + df['equity_change']).cumprod()

print(df['equity_curve'].head(5))
print(df['date'].head(5))
```

```
df.plot(figsize = (8,8),grid = True, use_index = True,x = 'date',y = "stock_num",color = 'black')
df.plot(figsize = (8,8),grid = True, use_index = True,x = 'date',y = "net_value",color = 'r')
df.plot(figsize = (8,8),grid = True, use_index = True,x = 'date',y = "equity_curve",color = 'g')
df.plot(figsize = (8,8),grid = True, use_index = True,x = 'date',y = "equity_change",color = 'b')

print("the frequency of buy or sell:",len(df.loc[open_pos_condition + close_pos_condition, 'stock_num']))
max_net = df['net_value'].max()

remains = df['equity_curve']/df['equity_curve'].expanding().max()

print("最大回撤率为：",round((1 - remains.min()),2))
print("收益率均值为：",df['equity_change'].mean())
print("收益率标准差为：",df['equity_change'].std())
print("交易频率为：",len(df.loc[open_pos_condition + close_pos_condition, 'stock_num'])/len(df))
```

11. 不考虑手续费，印花税，滑点等回测

```
import pandas as pd
import numpy as np

df = pd.read_csv("output/signal.csv")

def test(df,Commission = 0):
    #bid = 0 说明买入，否则卖出，account 是入账金额
    capital_set = 100000000
    account = pd.Series(np.zeros((len(df))))
    capital_set_d = pd.Series(np.zeros((len(df))))
    capital_set_d[0] = 100000000
    bid = 1
    num = 0
    for i in [j + 1 for j in range(len(df)-1)]:

        change = (df['close'][i] - df['close'][i - 1])/df['close'][i - 1]

        #print(change)
```

```
        if (df['finalsig'][i] < 0.5):
            #buy
            if bid == 1:
                account[i] = capital_set - Commission*capital_set
                capital_set = 0
                capital_set_d[i] = 0
                bid = 0
                num = num + 1
            else:
                account[i] = account[i - 1]*(1 + change)

        elif (df['finalsig'][i] > 0.5):
            #sell
            if bid == 0:
                account[i] = account[i - 1]*(1 + change)
                capital_set    = account[i] - Commission*account[i]
                capital_set_d[i+1] = account[i] - Commission*account[i]
                account[i + 1] = 0
                bid = 1
                num = num + 1
            else:capital_set_d[i+1] = capital_set_d[i]
        else:
            account[i] = account[i - 1]*(1 + change)
            capital_set_d[i+1] = capital_set_d[i]

        if (i == len(df) - 1) and (bid == 0):
            #when the last day come, sell all of the capital in your account
            account[i] = account[i - 1]*(1 + change)
            capital_set    = account[i] - Commission*account[i]
            capital_set_d[i+1] = account[i] - Commission*account[i]
            bid = 1
            num = num + 1

    df['投入资金'] = account
    df['资金账户'] = capital_set_d
    df['net_value'] = account + capital_set_d
    df['equity_change'] = df['net_value'].pct_change(fill_method=None)
    df['equity_curve'] = (1 + df['equity_change']).cumprod()
    profit = (capital_set - 100000000)/100000000
    print("佣金率:%s"%str(Commission) + " 这 4 年年化收益率为：%s"%str(profit/4))
```

```
    return(df,capital_set,capital_set_d,num)

df,capital_set,capital_set_d,num = test(df,Commission = 0)
df.to_csv("output/result_1.csv",index = False)



#plot
df.plot(figsize = (8,8),grid = True, use_index = True,x = 'date',y = "equity_curve",color =
'red')
plt.show()

remains = df['equity_curve']/df['equity_curve'].expanding().max()

print("最大回撤率为：",round((1 - remains.min()),2))
print("收益率均值为：",df['equity_change'].mean())
print("收益率标准差为：",df['equity_change'].std())
print("交易频率为：",num/len(df))
```