文献梳理、研究框架、创新点与描述性统计

一、梳理Financial distress方面相关文献进行参考

用于预测财务困境的数据可大致分为两类：结构化数据（如市场数据和财务数据）和非结构化数据（如文本、图像和网络）。

1. 利用结构化数据预测财务困境

结构化数据通常涉及使用财务指标作为特征。近年来，学者们还对结构化数据的其他方面的贡献进行了研究。

结构化数据来预测公司困境。这又可分为公司内部数据、公司外部数据和混合数据。

公司内部数据包括公司环境、创新、资本结构和公司治理等因素。Pham等人分析了41个国家的公司困境，发现较好的工作环境与较高的财务稳定性之间存在相关性。Bai和Tian在对美国上市公司的破产预测研究中发现，创新绩效与破产概率之间存在负相关关系。Ji等人发现，数字金融发展水平是预测中国A股上市公司破产风险的有效指标。在资本结构方面，女性董事比例、银行对公司的控制权以及杠杆收购交易导致的资本结构急剧变化都是预测公司破产的有效指标。Liang等人通过逐步判别分析发现，选择六个具体的公司治理指标可以提高财务困境的预测性能，公司是否遵守公司治理准则也会影响预测性能。此外，环境、社会和治理评分、新任高管的初始报酬以及公司的避税程度都能提高预测公司财务失败的准确性。

外部数据主要包括市场和宏观经济因素。基于市场的违约距离模型通常是预测下一年财务困境的良好指标。政府效率高、企业活动多、腐败控制力强的国家可以降低企业破产的概率。混合数据是指整合不同方面的指标，以获得比单独使用单个指标更高的预测准确性。一种方法是将公司外部数据和内部数据结合起来，如将市场因素或公司治理指标与财务指标和宏观经济变量混合起来。

2. 利用非结构化数据预测财务困境

非结构化数据一般指文本数据，因为公司披露的财务报告文本具有官方性和权威性，因此受到大多数研究人员的青睐。Ashraf等人利用财务报告的质量作为特征。Wang等人利用一种新的随机子空间方法，结合情感和文本信息进行财务困境预测。Li等人使用自建的金融领域相关词典，从财务报告文本中的正面和负面词语中统计提取情感和语气，作为预测公司倒闭的特征。Mai等人使用Word2Vec将财务报告中的管理讨论与分析（MD&A）文本转化为词向量矩阵，并使用平均加权法获得文档向量作为模型输入。Matin等人进一步利用卷积神经网络从词向量矩阵中提取特征作为输入。Jiang等人提出了一个利用当前报告预测非上市公众公司财务困境的框架。Nguyen和Huynh使用金融情感词典和基于规则的情感分析来提高企业破产预测模型的分类性能。在除了官方财务报告文本，一些学者还使用其他类型的文本数据作为特征。Fedorova等人测量了Twitter文章中相关关键词的频率，以此作为评估经济政策不确定性的指标。Zhao等人使用LM词表作为特征，从在线股票论坛的评论中提取情感基调。除文本数据外，还有利用图像和网络进行预测的研究。Hosaka从财务报表中提取财务比率并将其表示为灰度图像，然后使用基于GoogLeNet的卷积神经网络对日本上市公司进行破产预测。Kou等人证实了基于交易数据和付款的网络变量对破产预测的重要性。类似的研究还包括使用专利文本信息、分析师报告和问答文本信息。

从已有的研究中可以发现，目前大多数研究集中于分析某些文本处理方法，如Word2Vec和BERT对财务困境预测的影响。同时存在word2vec和BERT的对比分析，以及基于word2vec和BERT的改进方法在财务困境预测中的有效性研究。有研究提出了四种模型：Word2Vec-平均模型、Word2Vec-加权模型、BERT-单词模型和BERT-句子模型来检验财务文本信息对财务困境预测的影响。研究以中国上市公司数据为样本进行模型验证，并通过交叉验证和滑动时间窗分析验证了向量化文本信息在四种不同预测期限下的预测性能。

二、结合违规披露预测上市公司退市的研究框架、创新点

以往的文献利用非结构化数据，一是从文本提取特定的变量，如情绪，二是结合深度学习中提取原始特征，将其与数值特征进行合并。后者的研究较少，且大多数是通过简单的特征拼接达到特征融合的目的。后者模型框架如图：

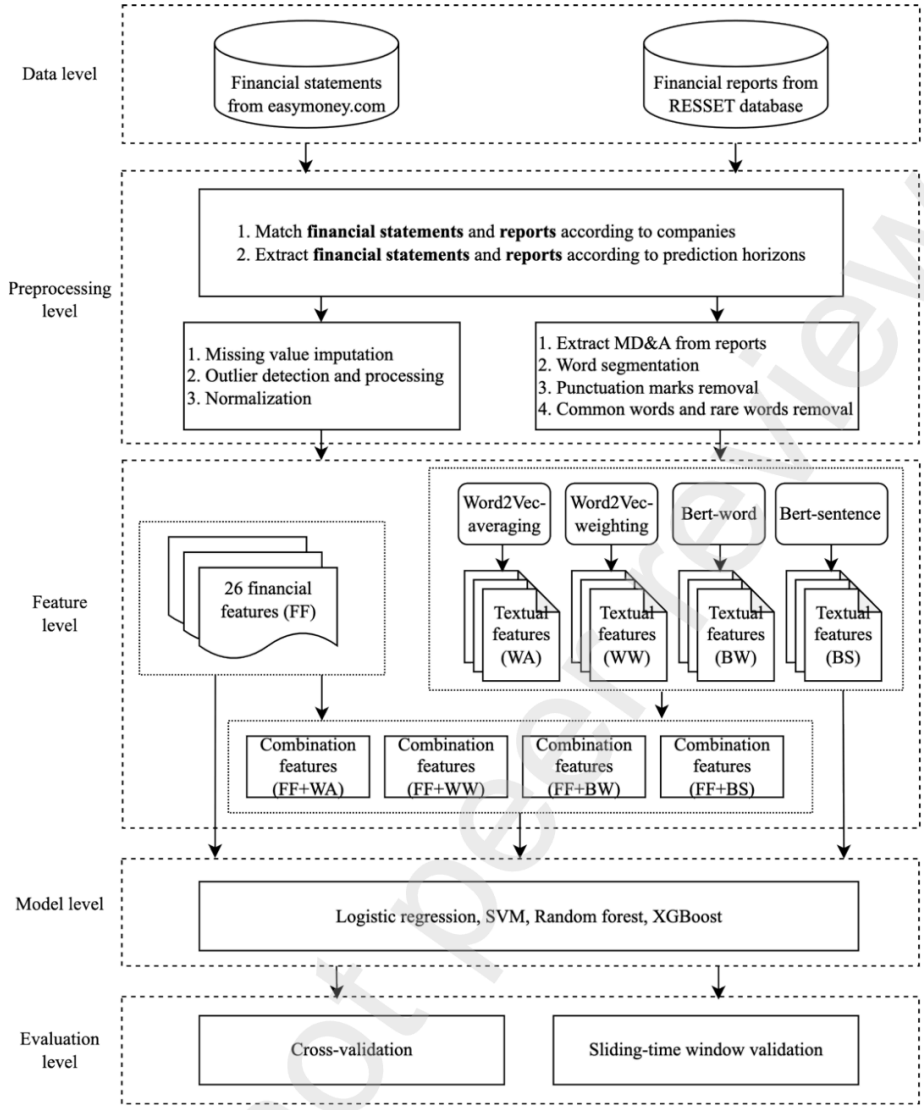
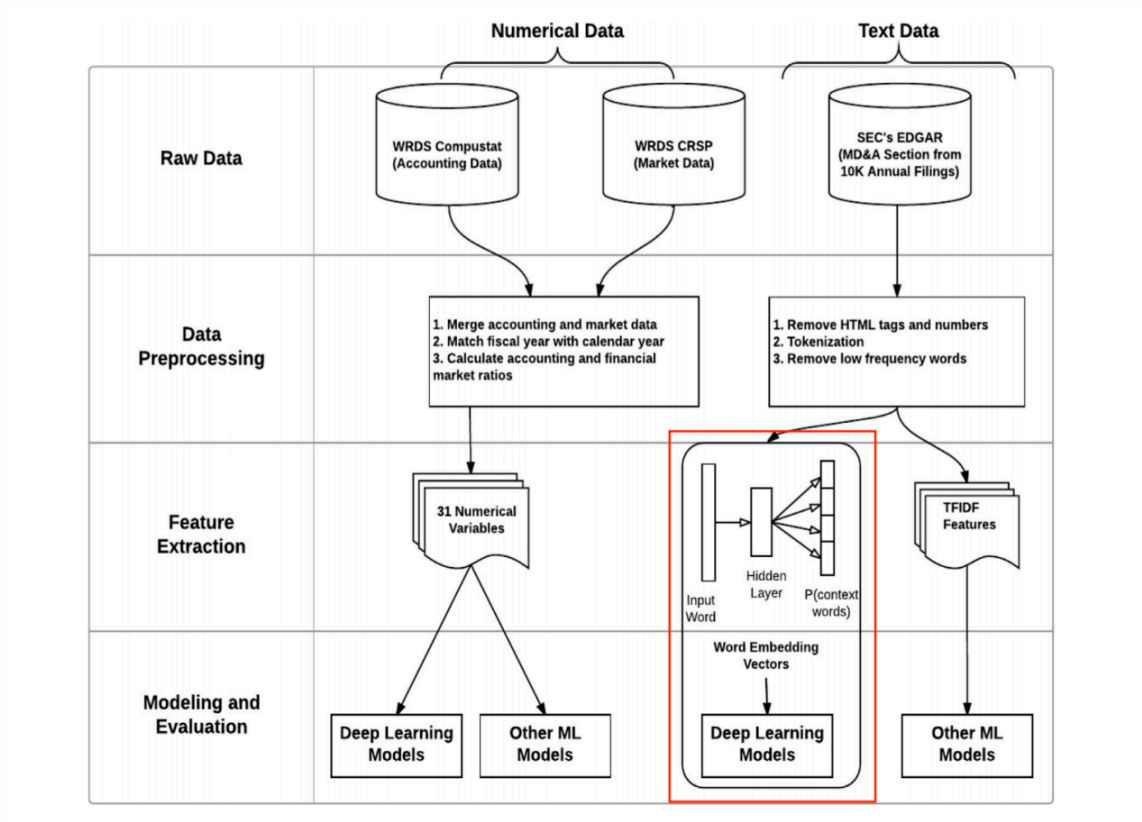


Figure 1 Framework and architecture

这样的思路的一个特点就是，将不同种类features合并的模型与最终做出预测的模型不同。比如示例中，feature level运用了较为先进的bert进行预处理和特征提取，而model level则使用的是机器学习将拼接的特征进行预测。另外我们观察到，Deep learning models for bankruptcy prediction using textual disclosures的文章中预测公司破产没有简单的特征拼接，而是将处理numeric feature的CNN与处理text的CNN在隐藏层进行合并，即模型并融，直接进行分类任务。如图标红部分：



以上面两篇文献为启发，我们试图将bert与深度学习模型进行模型并融，实现在模型内部层面进行合并，并直接做出预测。这样的模型范式目前在AI领域也十分热门，因为这样的模式可以让模型处理多种类型的数据，模型层面合并然后进行分类，包括视频、音频、图片以及文字等，即multimodal learning。研究主要围绕情感识别为核心进行分类任务。

以下是对multimodal learning 的文献梳理： 情感分析（SA）在人工智能（AI）和自然语言领域备受瞩目。自然语言处理（NLP）。自动分析用户对产品或服务的情感的需求日益增长。

越来越多的人在网上以视频的形式分享观点，而不仅仅是文字。这使得使用多种模式的情感分析（称为多模式情感分析（MSA））成为一个重要的研究领域。MSA在不同阶段利用机器学习和深度学习的最新进展，包括多模态特征提取和融合以及情感极性检测，旨在最大限度地降低错误率并提高性能。MSA架构的最新发展分为十类，即早期融合、晚期融合、混合融合、模型级融合、张量融合、分层融合、双模态融合、基于注意力的融合、基于量子的融合和词级融合。

BERT现在是许多实际应用中的主要模型，采用了大规模的语料库，包括无监督的语料库和有监督的语料库。通过无监督预训练，BERT学习了语言的结构和语义信息；通过有监督预训练，BERT进一步学习了特定的NLP任务，如情感分析、文本分类等。这种预训练方式使BERT能够更好地理解语义信息，从而提高其在各种NLP任务中的性能。

在多模态学习模型中，主要侧重于从听觉、视觉和文本等感官模态中学习，在多模态学习中，有几个研究分支。卡内基梅隆大学的 MultiComp 实验室提供了出色的分类法。我们的问题属于所谓的多模态融合——将来自两个或多个模态的信息连接起来进行预测。 由于文本数据是我们的主要模态，因此我们的综述集中在将文本视为主要模态的文献上，并介绍了利用转换器架构的模型。

在图像和文本的转换方面，Kiela 等人（2019 年）的 Supervised Multimodal Bitransformers for Classifying Images and Text 分别在单模态图像和文本上使用预训练的 ResNet 和预训练的 BERT 特征，并将其馈送到双向转换器中。关键的创新是将图像特征作为 transformer 模型的附加令牌进行调整。此外，还有一些模型——ViLBERT（Lu 等人，2019 年）和 VLBert（Su 等人，2020 年）——定义了图像和文本的预训练任务。这两个模型都在 Conceptual Captions 数据集上进行预训练，该数据集包含大约 330 万个图像-标题对（带有替代文本标

题的 Web 图像）。在这两种情况下，对于任何给定的图像，预训练的目标检测模型（如 Faster R-CNN）都会获取图像区域的向量表示，这些区域算作转换器模型的输入标记嵌入。另外，还有 LXMERT （Tan and Mohit 2019），这是另一个预训练的 transformer 模型，从 Transformers 版本 3.1.0 开始，作为库的一部分实现。LXMERT 的输入与 ViLBERT 和 VLBERT 相同。但是，LXMERT 在聚合数据集上进行预训练，其中还包括视觉问答数据集。LXMERT 总共对 918 万个图像文本对进行了预训练。

在音频、视频以及文本的转换方面，包括 MulT、用于未对齐多模态语言序列的多模态转换器（Tsai 等人，2019 年）和将多模态信息集成到大型预训练转换器中的多模态自适应门（MAG）（Rahman 等人，2020 年）。Wasifur Rahman等人（2020年）提出了BERT和XLNet的附件，称为多模态自适应门(MAG)。MAG允许BERT和XLNet在微调期间接受多模态非语言数据。它通过生成BERT和XLNet的内部表示的转换来实现这一点；以视觉和听觉模式为条件的转变。实验中，研究了常用的CMUMOSI和CMU-MOSEI数据集用于多模态情感分析。与之前的基线相比，微调MAGBERT和magg -XLNet以及仅对BERT和XLNet进行语言微调显著提高了情感分析性能。在CMUMOSI数据集上，MAG-XLNet在NLP领域首次实现了人类级别的多模态情感分析性能。MulT 类似于 ViLBert，其中在成对的模态之间使用共同注意力。MAG通过门控机制在某些变压器层注入其他模态信息。

在文本和知识图谱嵌入方面， 使用用于文档分类的知识图谱嵌入来丰富BERT（Ostendorff et al. 2019）除了用于书籍类别分类的元数据功能外，还使用维基数据知识图谱中作者实体的特征。ERNIE（Zhang 等人，2019 年）将输入文本中的标记与知识图谱中的实体进行匹配。它们将这些嵌入融合在一起，以生成具有此匹配的实体感知文本嵌入和文本感知实体嵌入。

*特别的，与本论文更加相关的多模态模型： Ken Gu等人（2021年）提供MultimodalToolkit,1一个开源Python包，用于将文本和表格(分类和数字)数据与下游应用程序的transformer合并。其工具包与hug Face现有的API集成得很好，如tokenization和model hub2，允许轻松下载不同的预训练模型。

在已有文献中我们找到了现有的多种模型可实现特征融合，其中模型层面并融如图标红：

Combine Feature Method	**Equation**
Text only	$\boldsymbol{m} = \boldsymbol{x}$
Concat	$\boldsymbol{m} = \boldsymbol{x}\|\|\boldsymbol{c}\|\|\boldsymbol{n}$
Individual MLPs on categorical and-numerical features then concat (MLP + Concat)	$\boldsymbol{m} = \boldsymbol{x}\|\|\text{MLP}(\boldsymbol{c})\|\|\text{MLP}(\boldsymbol{n})$
MLP on concatenated categorical and numerical features then concat (Concat + MLP)	$\boldsymbol{m} = \boldsymbol{x}\|\|\text{MLP}(\boldsymbol{c}\|\|\boldsymbol{n})$
Attention on categorical and numerical features (Attention)	$\boldsymbol{m} = \alpha_{x,x}\boldsymbol{W}_x\boldsymbol{x} + \alpha_{x,c}\boldsymbol{W}_c\boldsymbol{c} + \alpha_{x,n}\boldsymbol{W}_n\boldsymbol{n}$ $\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\boldsymbol{a}^T[\boldsymbol{W}_i\boldsymbol{x}_i\|\|\boldsymbol{W}_j\boldsymbol{x}_j]))}{\sum_{k\in\{x,c,n\}}\exp(\text{LeakyReLU}(\boldsymbol{a}^T[\boldsymbol{W}_i\boldsymbol{x}_i\|\|\boldsymbol{W}_k\boldsymbol{x}_k]))}$
Gating on categorical and numerical features and then sum (Rahman et al., 2020) (Gating)	$\boldsymbol{m} = \boldsymbol{x} + \alpha\boldsymbol{h}$ $\boldsymbol{h} = \boldsymbol{g}_c \odot (\boldsymbol{W}_c\boldsymbol{c}) + \boldsymbol{g}_n \odot (\boldsymbol{W}_n\boldsymbol{n}) + b_h$ $\alpha = \min(\frac{\|\|\boldsymbol{x}\|\|_2}{\|\|\boldsymbol{h}\|\|_2} * \beta, 1)$ $\boldsymbol{g}_i = \text{R}(\boldsymbol{W}_{g_i}[\boldsymbol{i}\|\|\boldsymbol{x} + b_i)$ where β is a hyperparameter and R is an activation function
Weighted feature sum on text, categorical, and numerical features (Weighted Sum)	$\boldsymbol{m} = \boldsymbol{x} + \boldsymbol{w}_c \odot \boldsymbol{W}_c\boldsymbol{c} + \boldsymbol{w}_n \odot \boldsymbol{W}_n\boldsymbol{n}$

Table 1: The included combining methods in the combining module. Uppercase bold letters represent 2D matrices, lowercase bold letters represent 1D vectors. b is a scalar bias, $\boldsymbol{W}$ represents a weight matrix, and $||$ is the concatenation operator. Please see Rahman et al. (2020) for details on the gating mechanism.

按照文献梳理结果，其中以Wasifur Rahman多模态自适应门(MAG)等效果最佳。因此我们初步确定，用以上七种方法进行下一步实操，进行对比试验。

总而言之，multimodal learning的核心创新在于将不同类型的变量在模型层面进行合并处理，未来不仅仅是numeric categoric text三种类别的特征并融，还可以添加音频、视频、图片等信息，加入模型中，可以为解决变量缺失、不平衡等问题提供思路。并且，当前选择的

Wasifur Rahman多模态自适应门(MAG)等模型具有该适应性。

在实操方面，最近研究运用上述方法进行了三个关于性别歧视的分类任务，对我们的研究在技术上具有较强的相关度(Vallecillo-Rodríguez, del Arco, Ureña-López, Martín-Valdivia, & Montejo-Ráez, 2023)。其应用的同样是Ken Gu等人（2021年）提供的MultimodalToolkit。

1. 系统概述

上市公司退市是一项分类任务。我们提出了两种不同的架构，即基础架构（BA）和数字变量整合。数字信息解决方案考虑了与文本相关的信息，目的是让系统考虑文本的语义以及数字信息。

1.1 基础结构

基本架构是用于文本分类的 Transformers架构。在该架构中，我们对文本进行标记化， 然后将其传递给模型。模型对输入进行预处理并生成输出。为了进行文本分类，我们从模型 的最后一个隐藏状态获取分类标记。该标记旨在对整个输入序列进行编码。然后将该标记作 为输入传递给前馈网络，由该网络对文本进行分类，并根据每个特定任务所需的类数生成分 类结果。我们提出的两个实验分别称为 "基线"和 "带数据增强的基线"，都采用了这种架构。

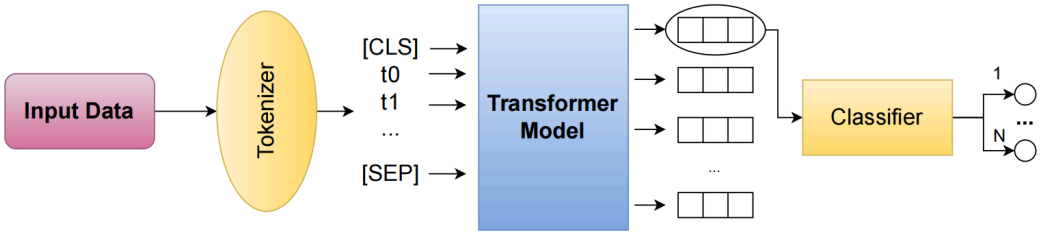


Figure 1: Base architecture proposed for EXIST shared task. N represents the number of output nodes and depends on the task because it corresponds to the number of labels to classify.

1.2 整合数字信息

这种设计是一种多模式架构，称为表格式转换器（Transformer With Tabular），它接收 一个数据帧作为输入。数据帧经过预处理，提取出代表要分类的文本、分类信息和数字信息 的列。然后，对文本特征进行标记化，对分类特征进行编码，并将数字特征转换为适当的格式。然后，将文本特征传递给转换器模型。Transformer 模型的输出会传递给一个组合模块，该模块会将数字特征和分类特征结合起来，生成一个独特的张量，然后传递给分类器，从而生成最终的预测结果。在图中这个案例里，没有数字特征，因此图 2 中没有显示这种架构。名为 "Proposed architecture to incorporate annotator information for EXIST shared task"。

该架构的组合模块代表了组合不同特征的策略。在这种情况下，其探索了以下方法：

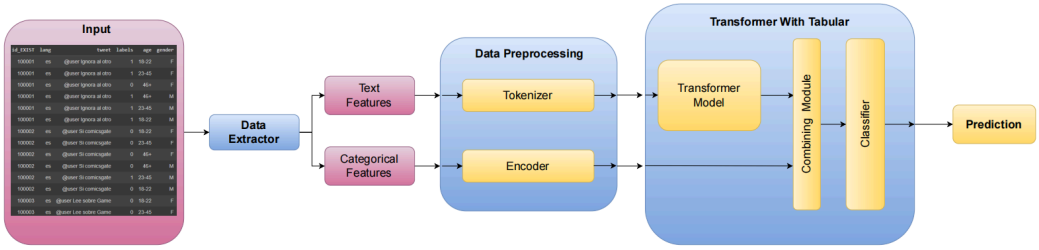


Figure 2: Proposed architecture to incorporate annotator information for EXIST shared task.

- **Concat**. Concatenate transformer model output and categorical features before the classifier.
- **Multi Layer Perceptron (MLP) on categorical features then concat**. We apply MLP

2. 数据选择

数字信息和文本信息结合。

2.1 数字信息部分：财务数据、市场数据

根据Mai 2020 中的总结，我们初步以此为标准：

Description of numeric variables.

Variable	Description	Variable	Description
ACTLCT	Current Assets/Current Liabilities	LTMTA	Total Liabilities/(Market Equity + Total Liabilities)
APSALE	Accounts Payable/Sales	LOG(AT)	Log(Total Assets)
CASHAT	Cash and Short-term Investment/Total Assets	LOG(SALE)	Log(Sale)
CASHMTA	Cash and Short-term Investment/(Market Equity + Total Liabilities)	MB	Market-to-Book Ratio
CHAT	Cash/Total Assets	NIAT	Net Income/Total Asset
CHLCT	Cash/Current Liabilities	NIMTA	Net Income/(Market Equity + Total Liabilities)
(EBIT+DP)/AT	(Earnings before Interest and Tax + Amortization and Depreciation)/Total Asset	NISALE	Net Income/Sales
EBITAT	Earnings before Interest and Tax/Total Asset	OIADPAT	Operating Income/Total Asset
EBITSALE	Earnings before Interest and Tax/Sales	OIADPSALE	Operating Income/Sales
EXCESS RETURN	Excess Return Over S&P 500 Index	PRICE	Log(Price)
FAT	Total Debts/Total Assets	QALCT	Quick Assets/Current Liabilities
INVCHINVT	Growth of Inventories /Inventories	REAT	Retained Earnings/Total Asset
INVTSALE	Inventories/Sales	RELCT	Retained Earnings/Current Liabilities
(LCT-CH)/AT	(Current Liabilities – Cash)/Total Asset	RSIZE	Log(Market Capitalization)
LCTAT	Current Liabilities/Total Asset	SALEAT	Sales/Total Assets
LCTLT	Current Liabilities/Total Liabilities	SEQAT	Equity/Total Asset
LCTSALE	Current Liabilities/Sales	SIGMA	Stock Volatility
LTAT	Total Liabilities/Total Assets	WCAPAT	Working Capital/Total Assets

Note: The table provides the description of the 36 numerical bankruptcy predictors.

2.2 文本信息

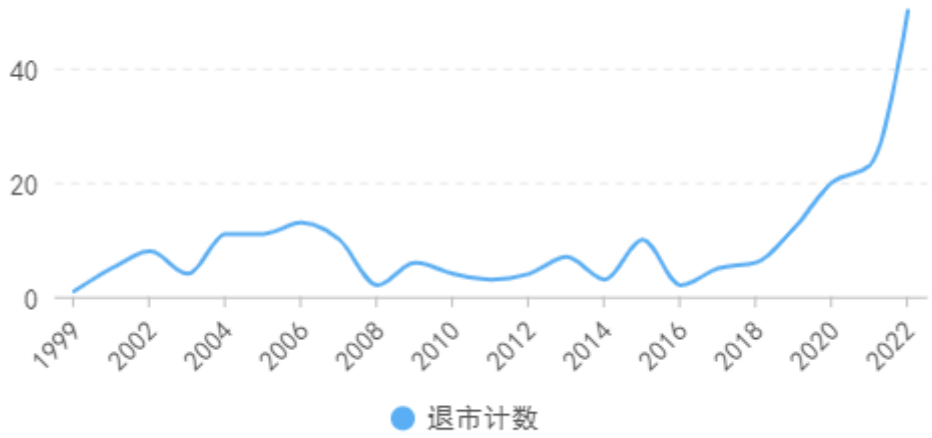
违规数据为现有数据集中的categorical and numeric变量以及违规行为中的text。而对于text这个部分，由于与通常文献使用年报中的MD&A文本不同，我们做了较为全面的描述性统计，其中我们特意利用financial-news-Chinese微调过的bert模型将文本情绪提取出来，并加入描述性统之中，以此增进我们对于该数据集文本内容的理解。

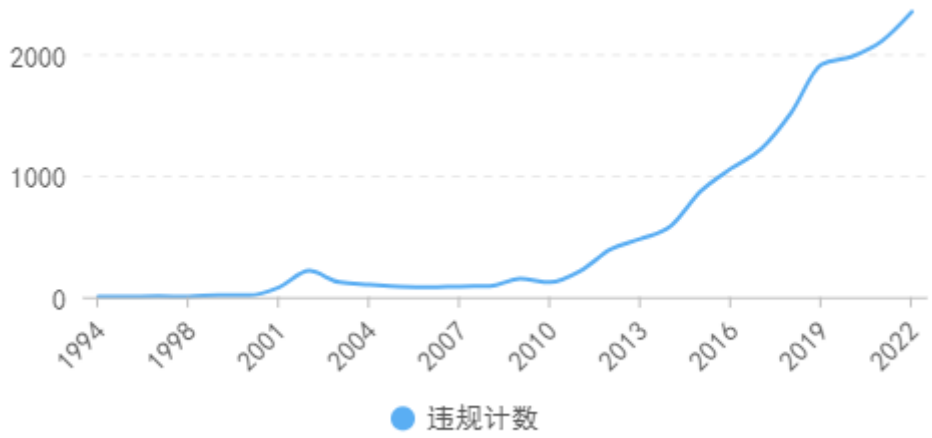
三、 描述性统计

In [22]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import jieba
import string
from collections import Counter
plt.rc("font",family='YouYuan')
```

In [23]:
```
df = pd.read_excel("C:/Users/刘成鑫/Desktop/论文---退市预警/数据集/可视化/可视化202
```

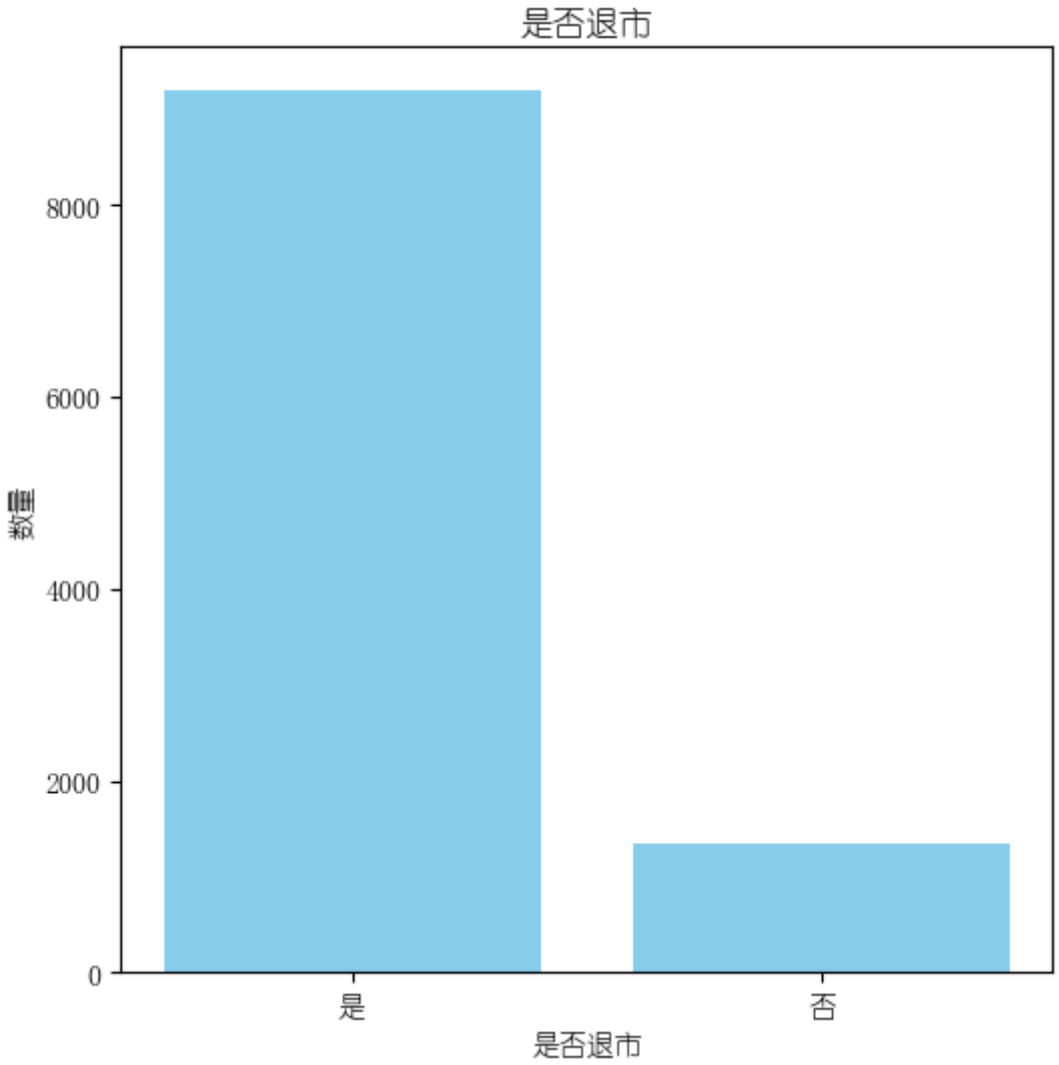
先看看整体的退市、违规、罚款之间的关系：发现都呈现上升趋势：



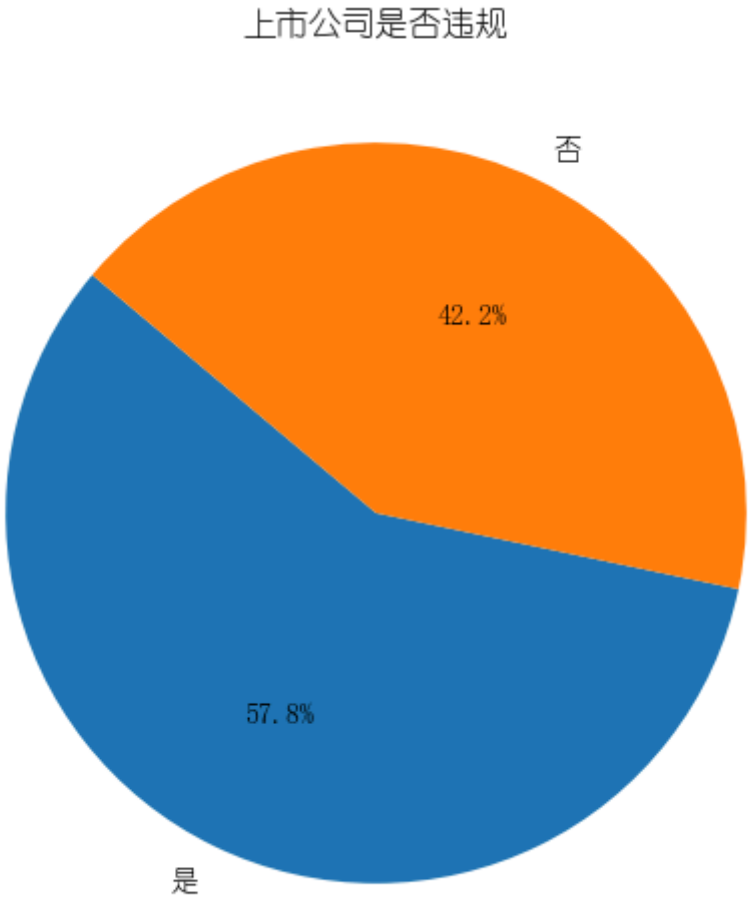


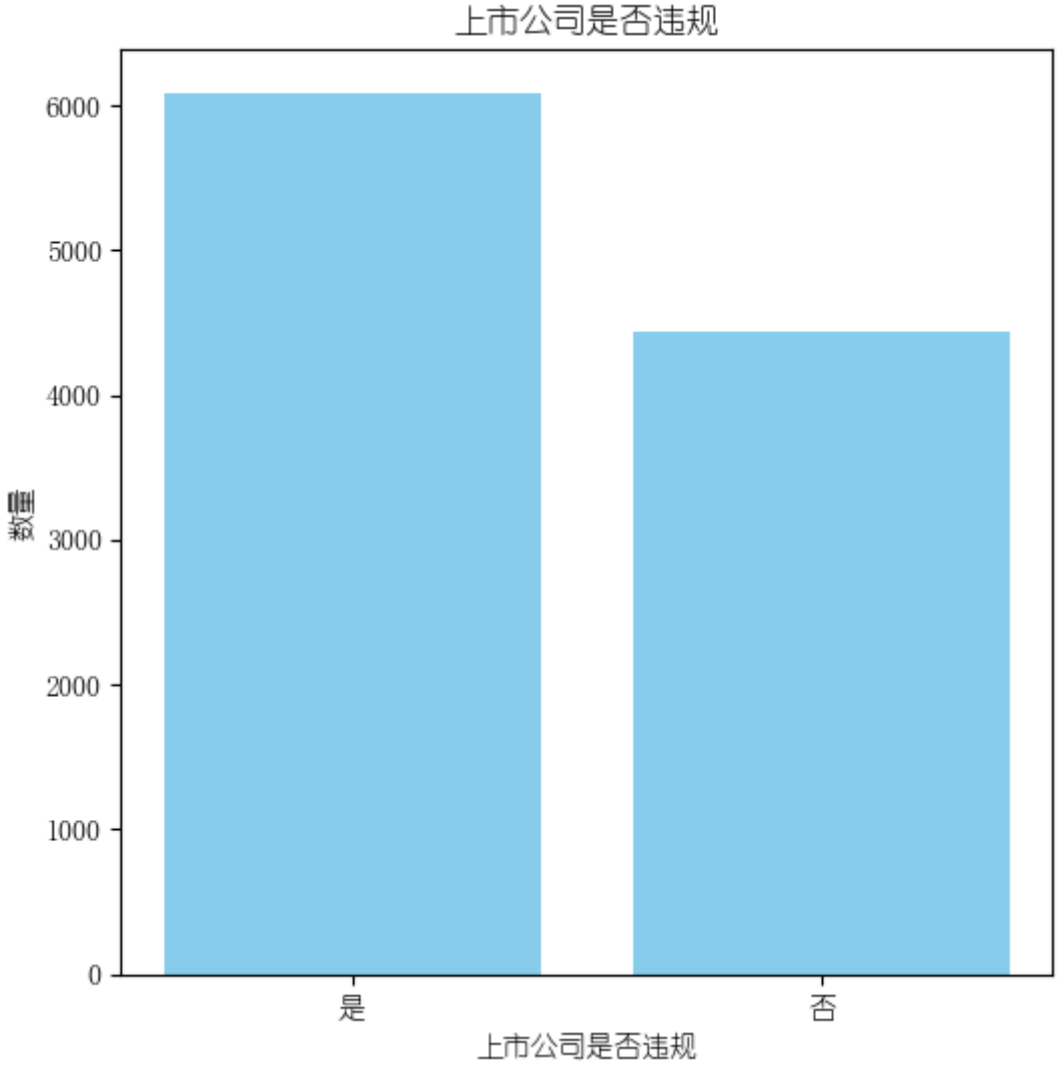
从总频数的角度统计一下：

In [3]:
```
#plot “是否退市”
counts = df['最终是否退市'].value_counts()

# plt.figure(figsize=(6, 6))
# plt.pie(counts, labels=['是', '否'], autopct='%1.1f%%', startangle=140)
# plt.title('是否退市')
# plt.show()

plt.figure(figsize=(6, 6))
bars = plt.bar(['是', '否'], counts, color='skyblue')
# plt.xticks(rotation=45, ha='right')
plt.title('是否退市')
plt.xlabel('是否退市')
plt.ylabel('数量')
plt.show()
```



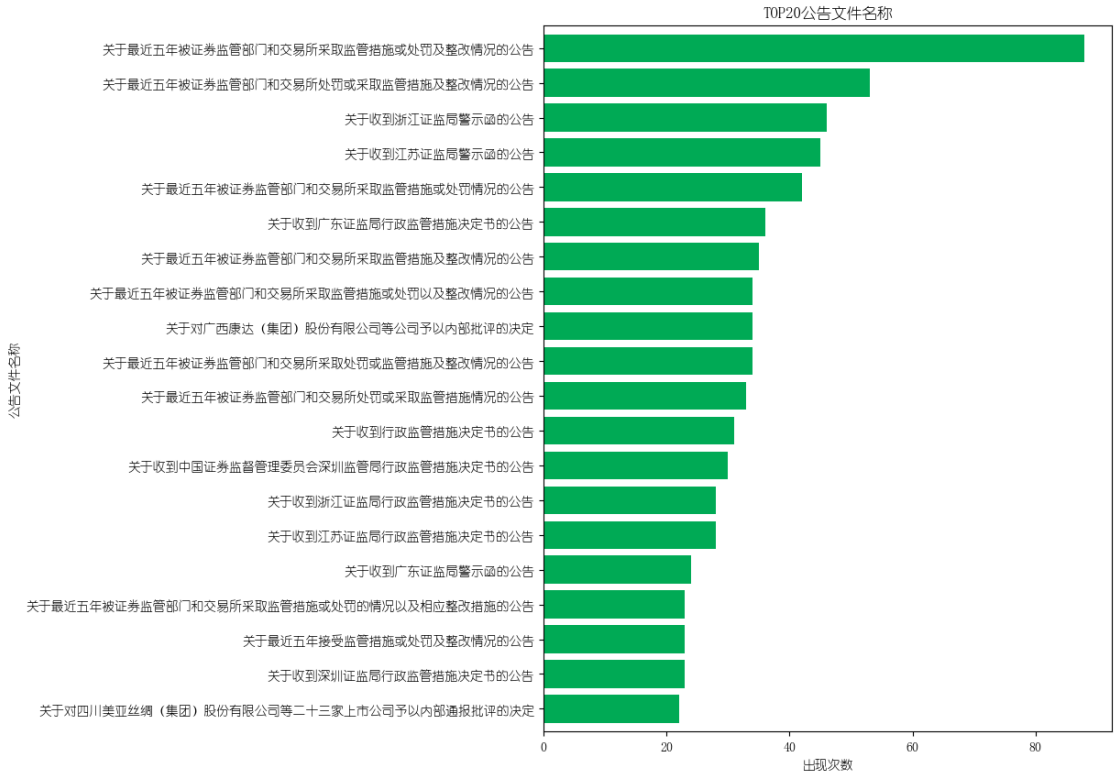
In [4]:
```
#plot “是否违规”
counts = df['上市公司是否违规'].value_counts()

plt.figure(figsize=(6, 6))
plt.pie(counts, labels=['是', '否'], autopct='%1.1f%%', startangle=140)
plt.title('上市公司是否违规')
plt.show()

plt.figure(figsize=(6, 6))
bars = plt.bar(['是', '否'], counts, color='skyblue')
# plt.xticks(rotation=45, ha='right')
plt.title('上市公司是否违规')
plt.xlabel('上市公司是否违规')
plt.ylabel('数量')
plt.show()
```





以下正式围绕违规数据集中的变量进行相关可视化，文本中消极情绪指标用bert提取：

In [5]:
```
# plot “公告文件名称”top20
file_name_column = '公告文件名称'
text_values = df[file_name_column].astype(str)

word_counts = text_values.value_counts().head(20)

plt.figure(figsize=(8, 10))
word_counts.plot(kind='barh', width=0.8, color='#00AA55')
plt.title('TOP20公告文件名称')
plt.xlabel('出现次数')
plt.ylabel('公告文件名称')
plt.gca().invert_yaxis()
plt.show()
```

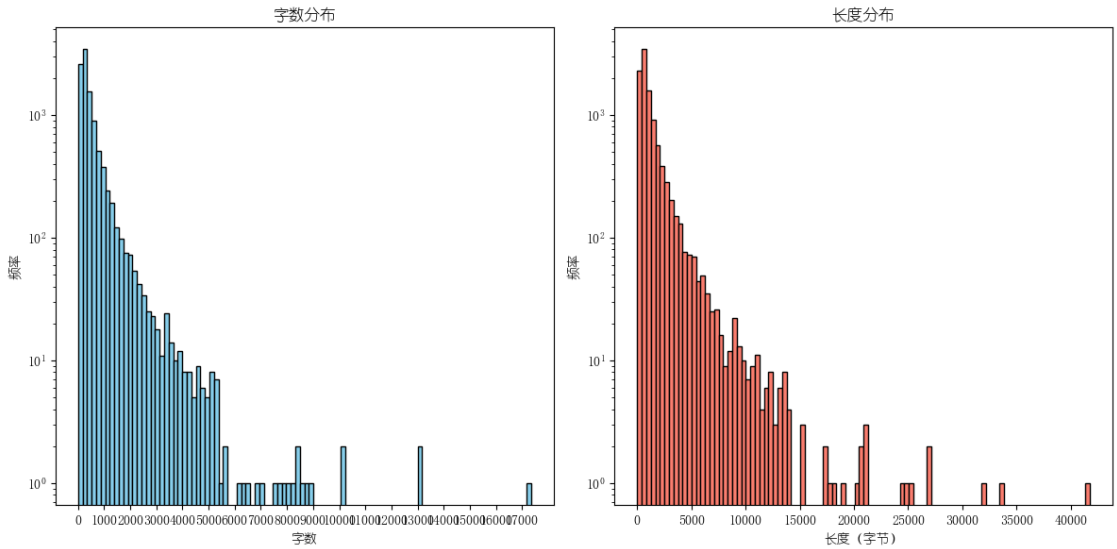


In [6]:

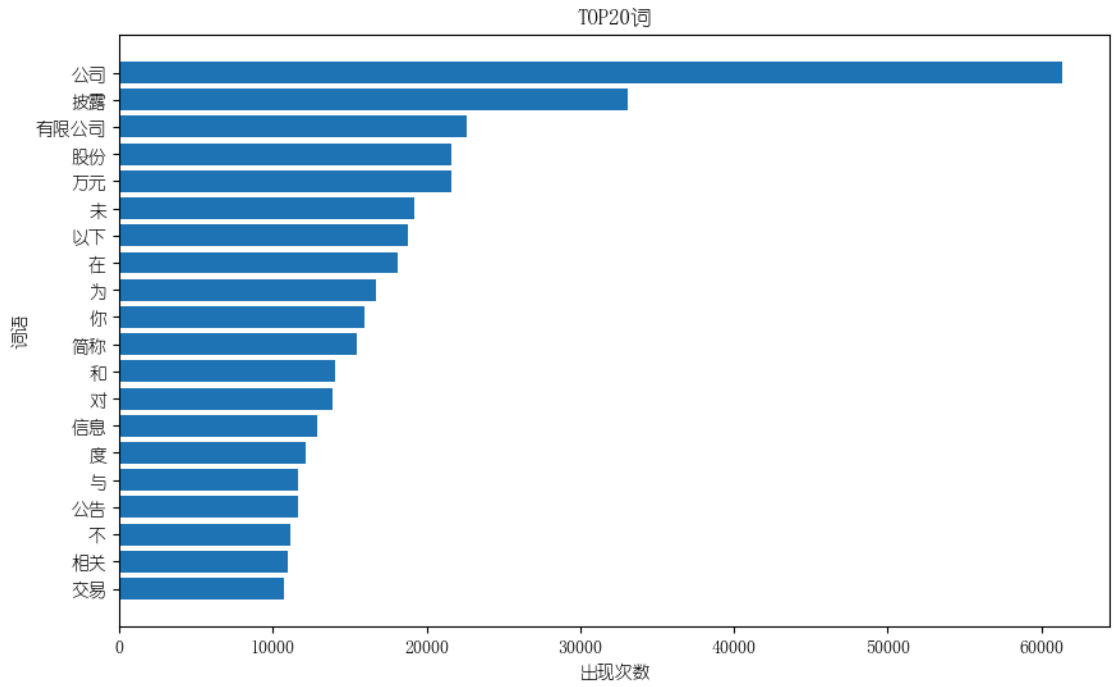
```
# plot 违规行为字数、长度、词频
text_column = '违规行为'

def preprocess(ReviewText):
    ReviewText = ReviewText.str.replace("(<br/>)", "")
    ReviewText = ReviewText.str.replace('(<a).*(>).*(</a>)', '')
    ReviewText = ReviewText.str.replace('(&amp)', '')
    ReviewText = ReviewText.str.replace('(&gt)', '')
    ReviewText = ReviewText.str.replace('(&lt)', '')
    ReviewText = ReviewText.str.replace('(\xa0)', '')
    ReviewText = ReviewText.str.replace('\n', '')
    return ReviewText

df['违规行为'] = preprocess(df['违规行为'])

texts = df[text_column].astype(str)

df['字数'] = texts.apply(lambda x: len(x))
df['长度'] = texts.apply(lambda x: len(x.encode('utf-8')))

max_text_length = df['长度'].max()
max_word_count = df['字数'].max()

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(df['字数'], bins=100, color='skyblue', edgecolor='black')
plt.title('字数分布')
plt.xlabel('字数')
plt.ylabel('频率')
plt.xticks(np.arange(0, max_word_count, 1000))
plt.yscale('log')

plt.subplot(1, 2, 2)
plt.hist(df['长度'], bins=100, color='salmon', edgecolor='black')
plt.title('长度分布')
plt.xlabel('长度（字节）')
plt.ylabel('频率')
plt.xticks(np.arange(0, max_text_length, 5000))
plt.yscale('log')
plt.tight_layout()
plt.show()

def chinese_segmentation(text):
    return jieba.cut(text)

def remove_punctuation(text):
    translator = str.maketrans("", "", string.punctuation)
    return text.translate(translator)

#删点不重要的词
texts = texts.str.replace('的', '')
texts = texts.str.replace('年', '')
texts = texts.str.replace('月', '')
texts = texts.str.replace('日', '')
# texts = texts.str.replace('在', '')
# texts = texts.str.replace('未', '')
# texts = texts.str.replace('万元', '')
# texts = texts.str.replace('为', '')
# texts = texts.str.replace('你', '')
texts = texts.str.replace('《', '')
texts = texts.str.replace('》', '')
```

```
texts = texts.str.replace('“', '')
texts = texts.str.replace('”', '')
texts = texts.str.replace('。', '')
texts = texts.str.replace(',', '')
texts = texts.str.replace('、', '')

word_counts = Counter()

for text in texts:
    text = remove_punctuation(text)
    words = chinese_segmentation(text)
    word_counts.update(words)

words, counts = zip(*word_counts.most_common(20))  # 取top20
plt.figure(figsize=(10, 6))
plt.barh(words, counts)
plt.xlabel('出现次数')
plt.ylabel('词语')
plt.title('TOP20词')
plt.gca().invert_yaxis()
plt.show()
```

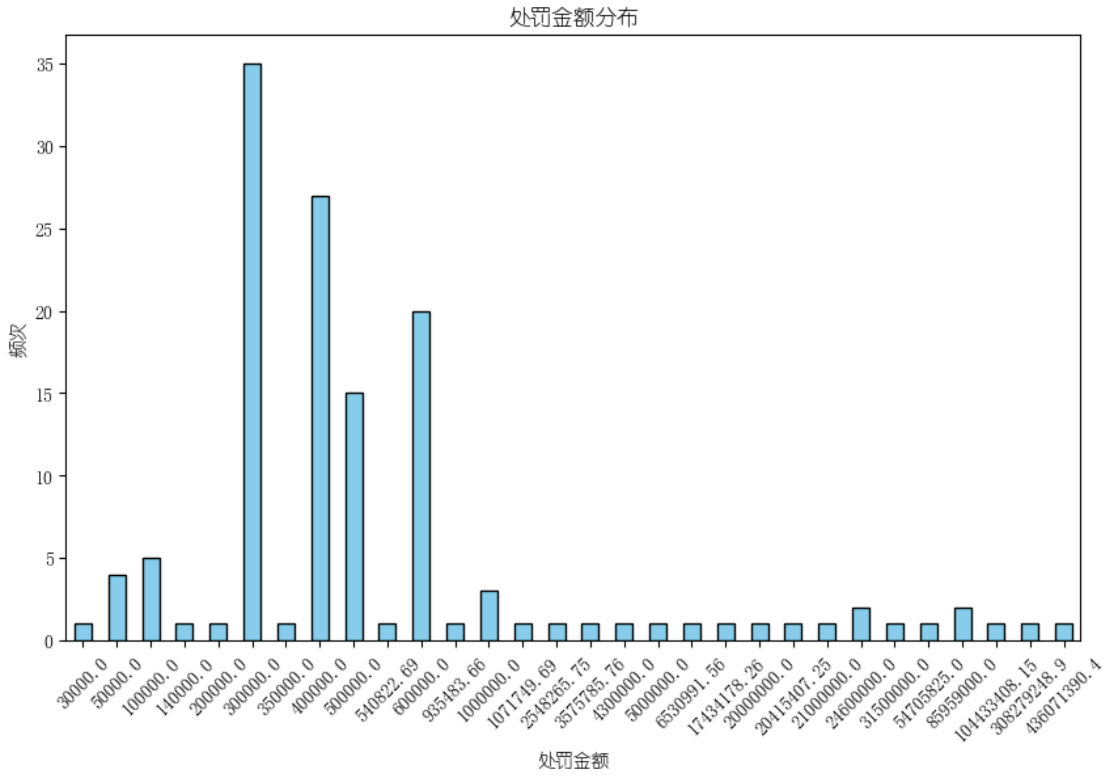
```
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:5: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace("(<br/>)", "")
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:6: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace('(<a).*(>).*(</a>)', '')
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:7: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace('(&amp)', '')
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:8: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace('(&gt)', '')
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:9: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace('(&lt)', '')
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:10: FutureWarnin
g: The default value of regex will change from True to False in a future versio
n.
  ReviewText = ReviewText.str.replace('(\xa0)', '')
```



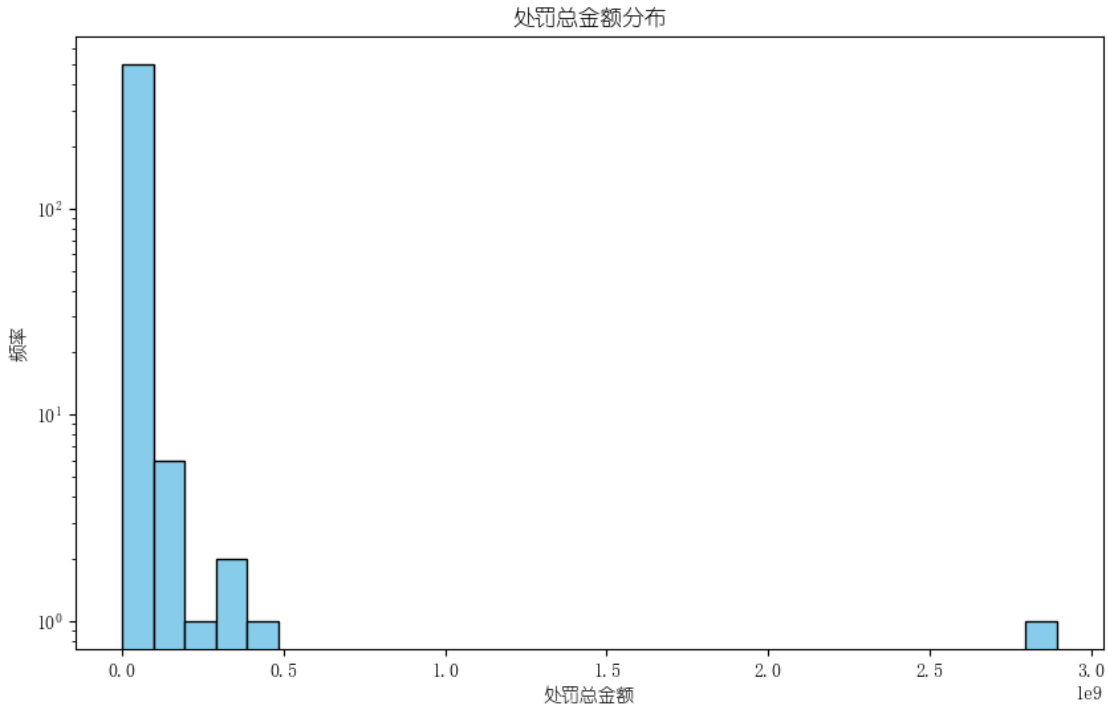

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\17629\AppData\Local\Temp\jieba.cache
Loading model cost 0.748 seconds.
Prefix dict has been built successfully.
```



In [7]:
```
# plot “处罚金额-上市公司”分布
punishment_amount_column = '处罚金额-上市公司'
numeric_values = pd.to_numeric(df[punishment_amount_column], errors='coerce').drop

plt.figure(figsize=(10, 6))
counts = numeric_values.value_counts().sort_index()
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('处罚金额分布')
plt.xlabel('处罚金额')
plt.ylabel('频次')
plt.xticks(rotation=45, ha='center')
plt.show()
```

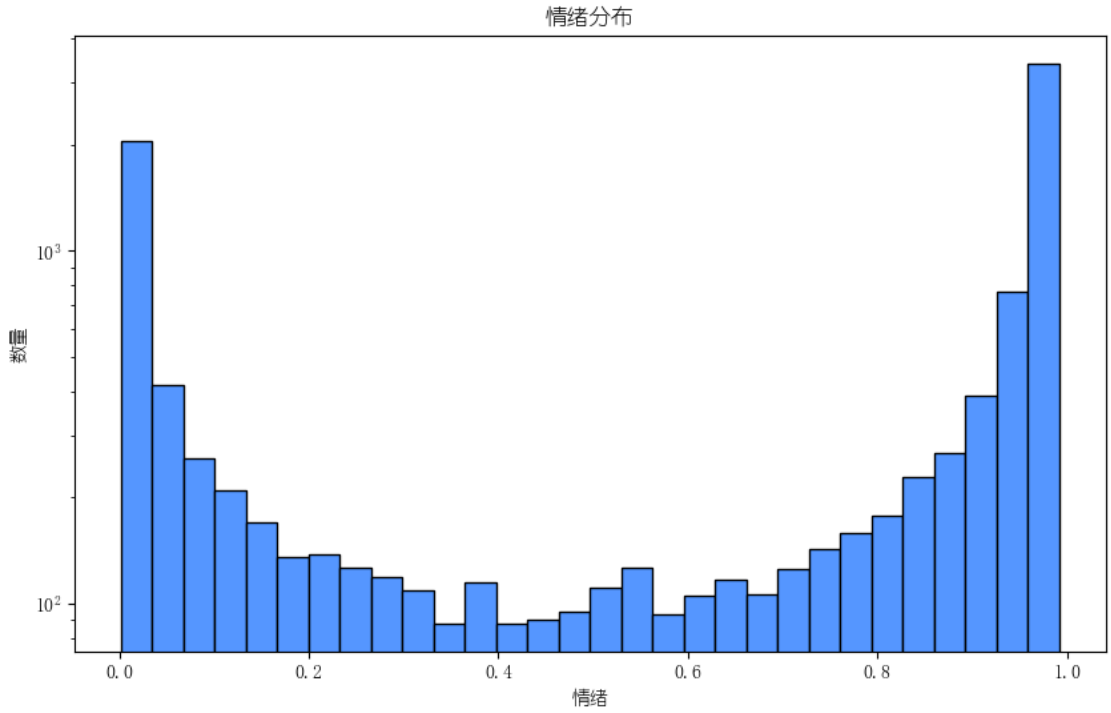


In [8]:
```
# plot “处罚总金额-上市公司”分布
punishment_column = '处罚总金额'
numeric_values = pd.to_numeric(df[punishment_column], errors='coerce').dropna()

plt.figure(figsize=(10, 6))
plt.hist(numeric_values, bins=30, color='skyblue', edgecolor='black')
plt.title('处罚总金额分布')
plt.xlabel('处罚总金额')
plt.ylabel('频率')
plt.yscale('log')
plt.show()
```



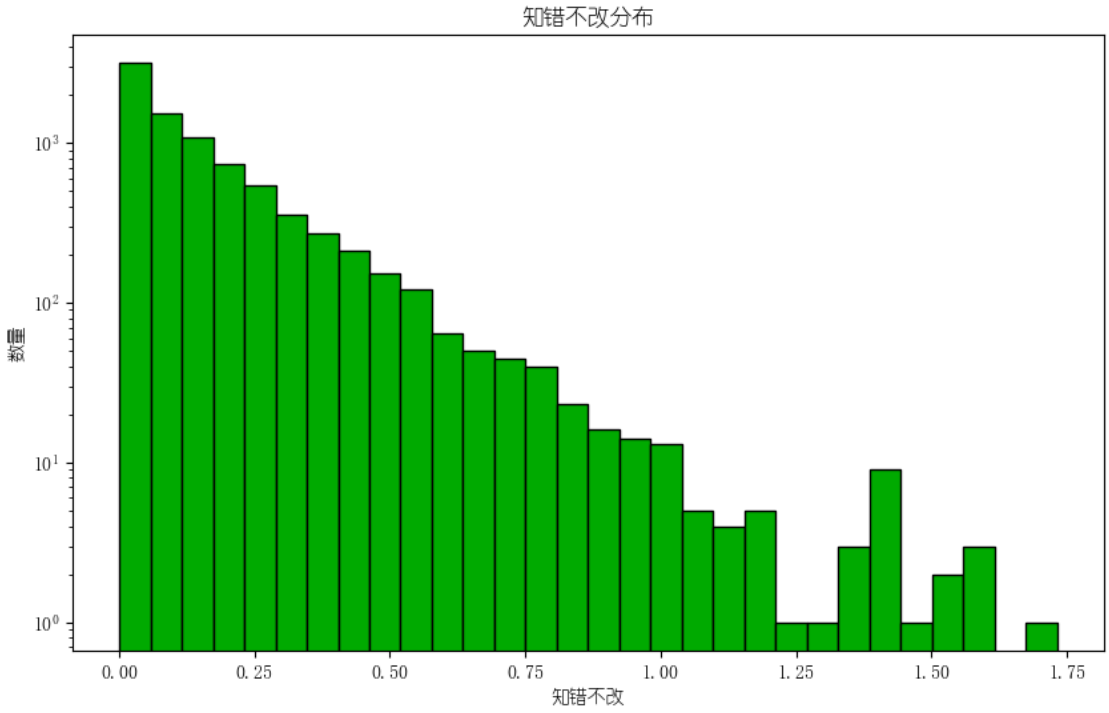
情绪变量设置在（0，1），默认全为negative，数值=0时，情绪为中性

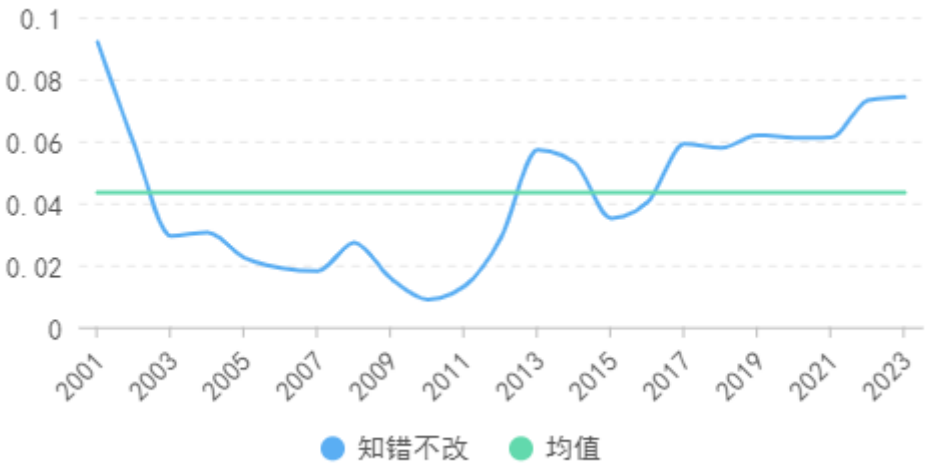
In [9]:
```
#plot sentiment分布
punishment_column = 'Adjusted_Sentiment_Score'
numeric_values = pd.to_numeric(df[punishment_column], errors='coerce').dropna()

plt.figure(figsize=(10, 6))
plt.hist(numeric_values, bins=30, color='#5599FF', edgecolor='black')
plt.title('情绪分布')
plt.xlabel('情绪')
plt.ylabel('数量')
plt.yscale('log')
plt.show()
```



知错不改定义：公司在某一时间点累计重复某一特定种类的违规类型次数越多，占所有涉及的违规类型中比率越大，此数值越大。

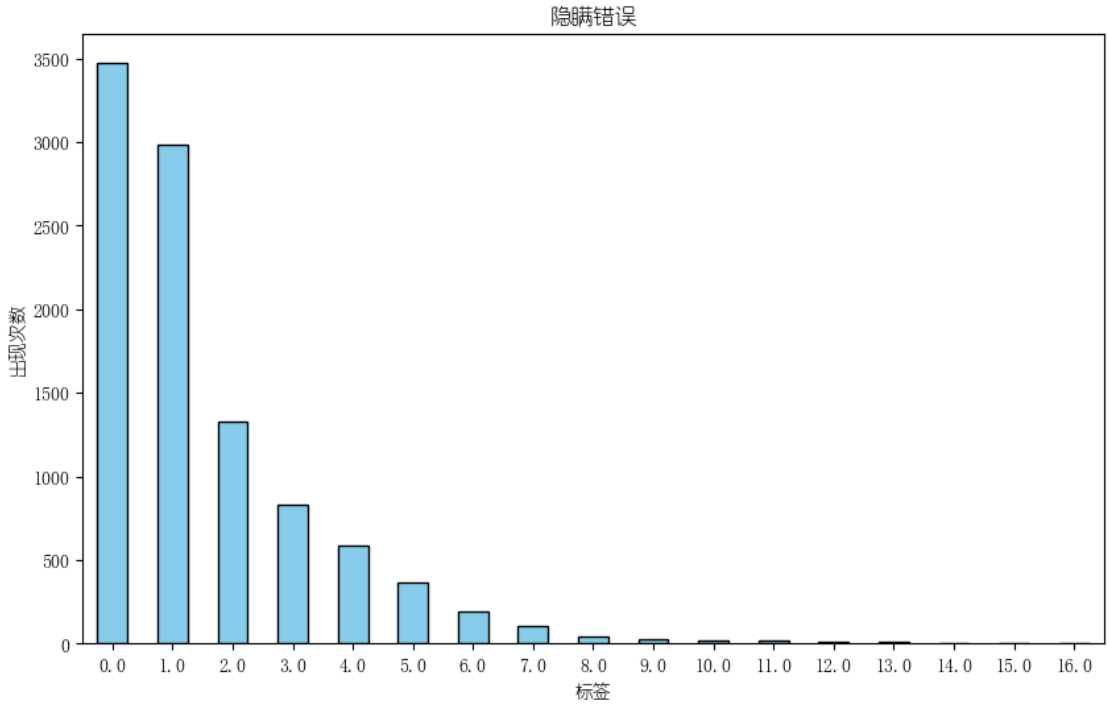
In [10]:
```
#plot 知错不改分布
punishment_column = '知错不改transformed_sqrt'
numeric_values = pd.to_numeric(df[punishment_column], errors='coerce').dropna()

plt.figure(figsize=(10, 6))
plt.hist(numeric_values, bins=30, color='#00AA00', edgecolor='black')
plt.title('知错不改分布')
plt.xlabel('知错不改')
plt.ylabel('数量')
plt.yscale('log')
plt.show()
```

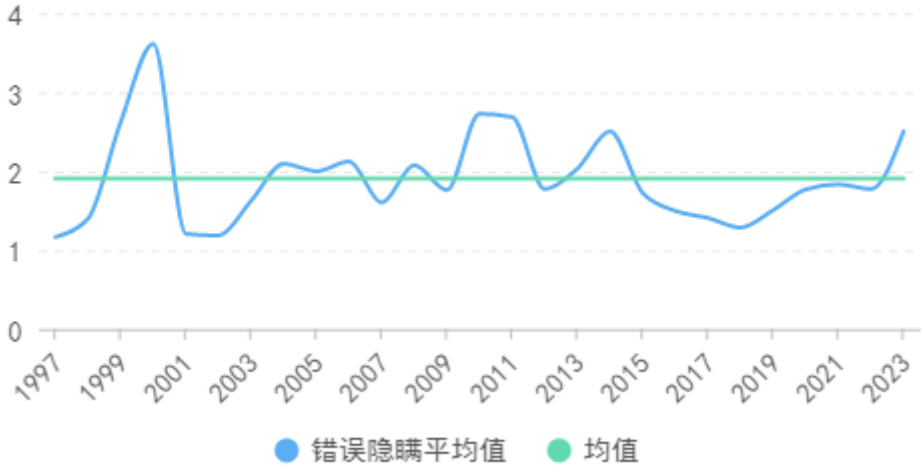




隐瞒错误定义：做出违规行为的实际时间和被官方披露的时间差（年）

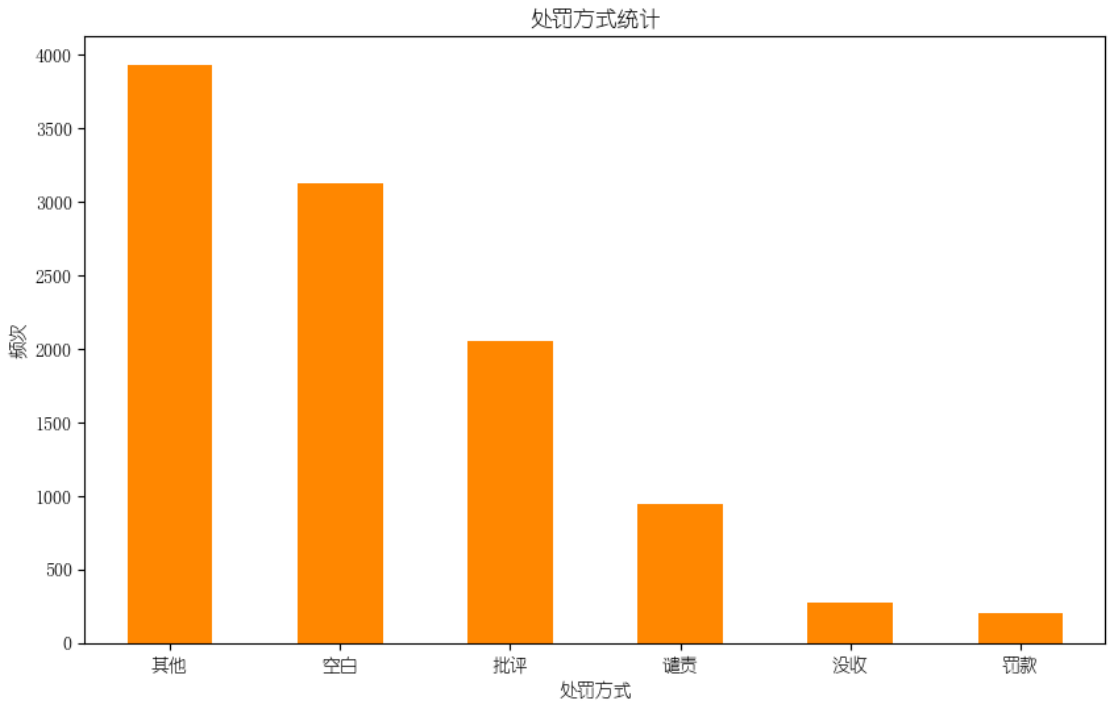
In [11]:
```
# plot “隐瞒错误”

label_column = '隐瞒错误'
positive_integers = df[label_column][(df[label_column] >= 0) & (df[label_column] %

integer_counts = positive_integers.value_counts()

plt.figure(figsize=(10, 6))
integer_counts.sort_index().plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('隐瞒错误')
plt.xlabel('标签')
plt.ylabel('出现次数')
plt.xticks(rotation='horizontal',ha='center')
plt.xticks(integer_counts.index.astype(int))
plt.show()
```

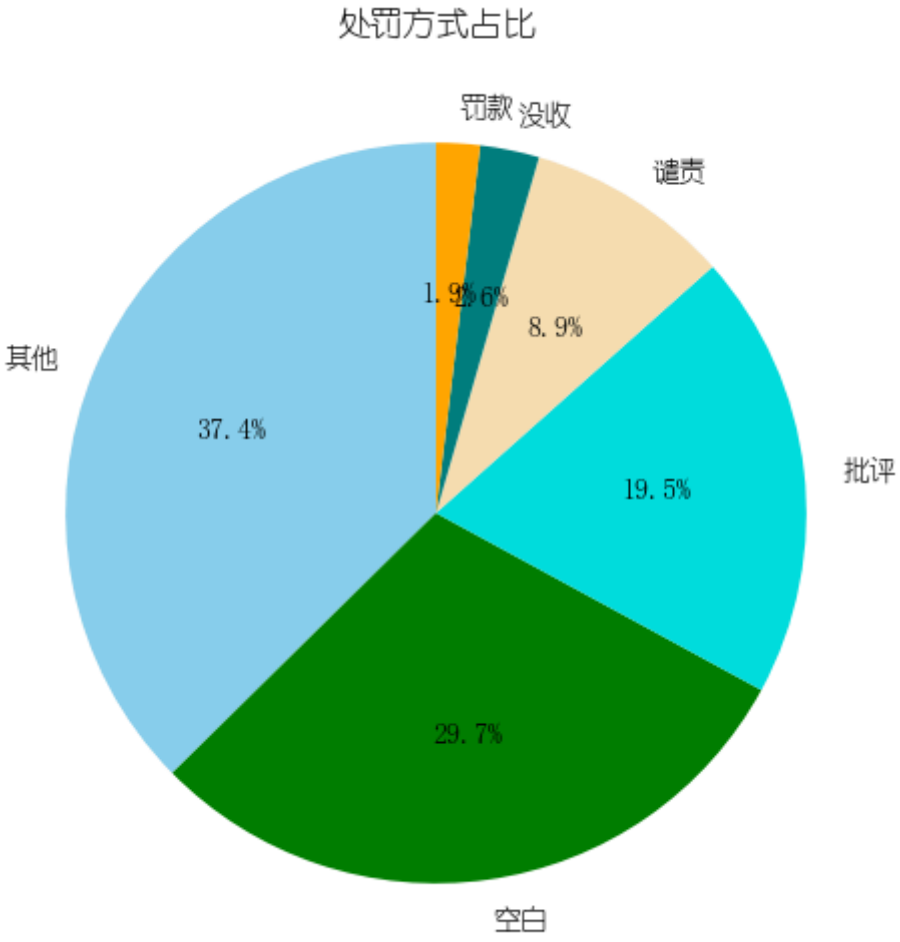




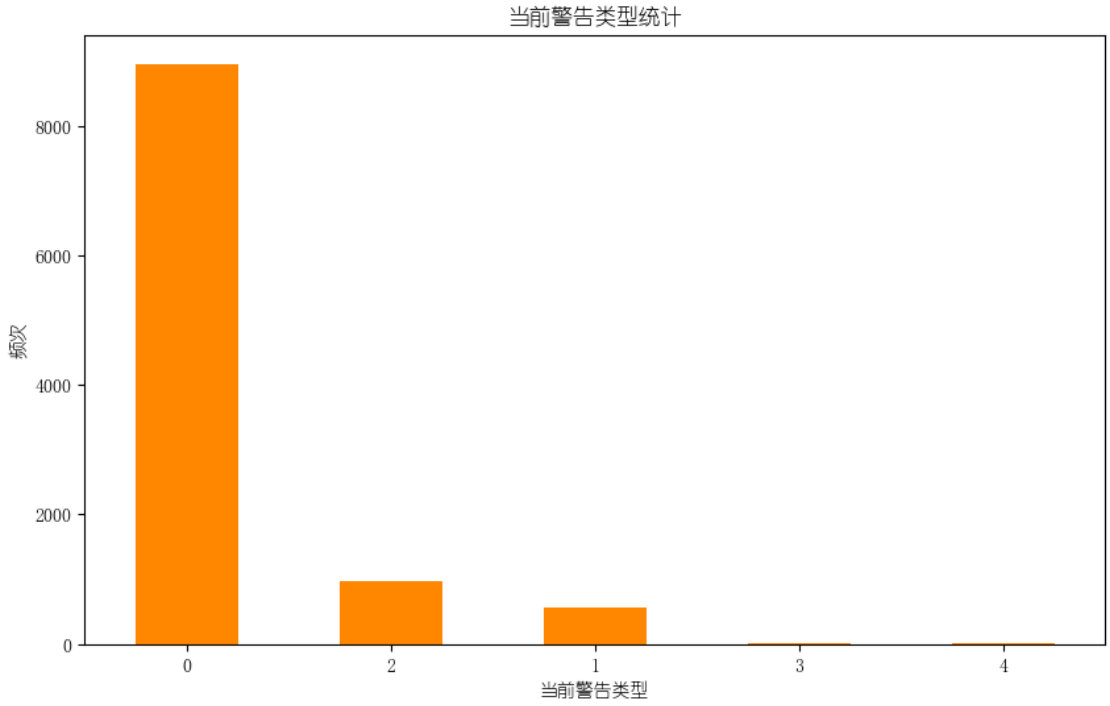
在两年周围波动，也就是所有公司平均而言，历史上发生违规行为和被监管部门处理的时间差恒定在2年附近

In [12]:
```
# plot “处罚方式”
punishment_column = '处罚方式（总）'

df[punishment_column].fillna('空白', inplace=True)

text_counts = df[punishment_column].value_counts()

plt.figure(figsize=(10, 6))
text_counts.plot(kind='bar', color='#FF8800')
plt.title('处罚方式统计')
plt.xlabel('处罚方式')
plt.ylabel('频次')
plt.xticks(rotation=0, ha='center')
plt.show()

colorpie = ['skyblue','green','#00DDDD','wheat','teal','orange']
plt.figure(figsize=(6, 6))
text_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=colorpie)
plt.title('处罚方式占比')
plt.ylabel('')
plt.show()
```

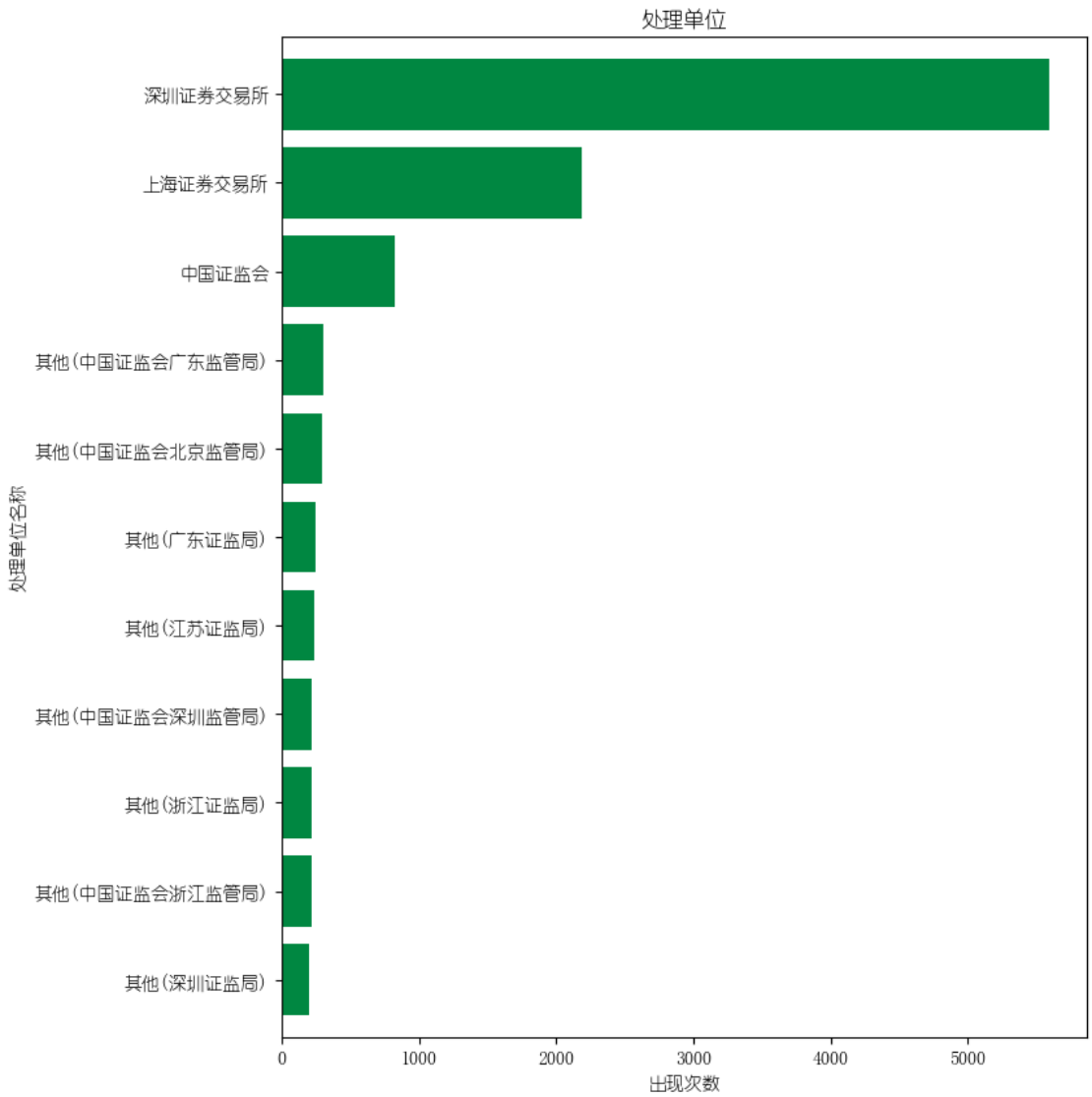



处罚方式占比
罚款
没收
谴责
批评
空白
其他
1.9%
2.6%
8.9%
19.5%
29.7%
37.4%

In [31]:
```
# plot “当前警告类型”
# 无警示状态=0，ST=1,*ST=2,S*ST=3,PT=4
punishment_column = '当前警告类型'

df[punishment_column].fillna(1 , inplace=True) #把空着的当成0

text_counts = df[punishment_column].value_counts()

plt.figure(figsize=(10, 6))
text_counts.plot(kind='bar', color='#FF8800')
plt.title('当前警告类型统计')
plt.xlabel('当前警告类型')
plt.ylabel('频次')
plt.xticks(rotation=0, ha='center')
plt.show()
```

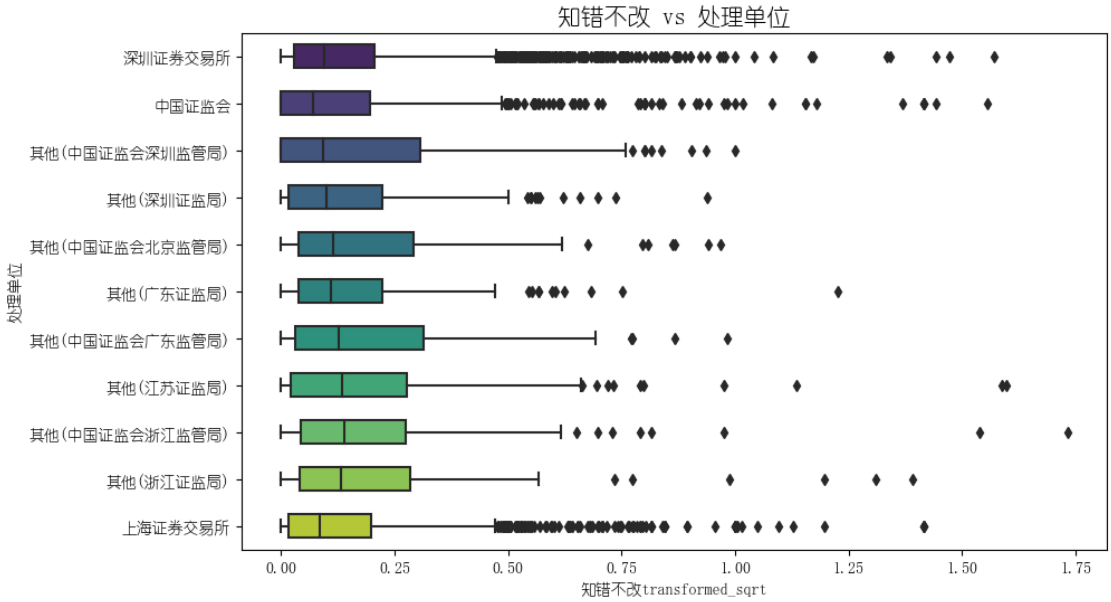


In [9]:
```
#plot “处罚单位”
file_name_column = '处理单位'
text_values = df[file_name_column].astype(str)

word_counts = text_values.value_counts()

plt.figure(figsize=(8, 10))
word_counts.plot(kind='barh', width=0.8, color='#008844')
plt.title('处理单位')
plt.xlabel('出现次数')
plt.ylabel('处理单位名称')
plt.gca().invert_yaxis()
plt.show()
```

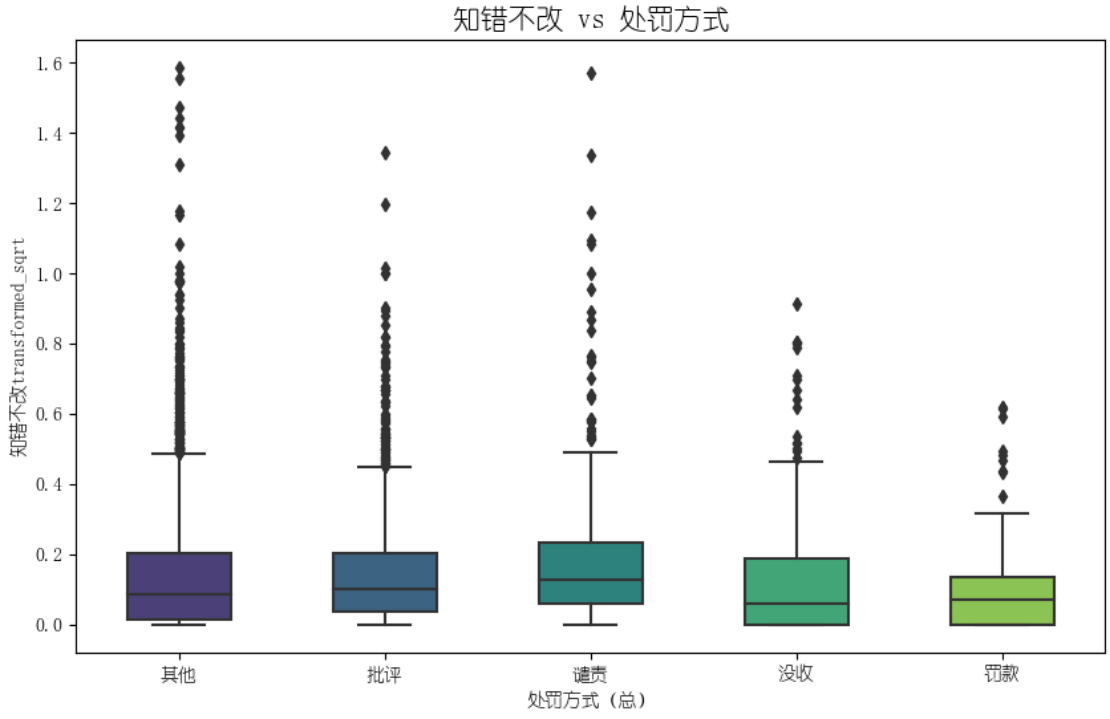


以下开始进行变量之间关系的可视化:

In [10]:
```
#plot 可视化 知错不改与处理单位的关系
plt.figure(figsize=(10, 6))
sns.boxplot(x='知错不改transformed_sqrt', y='处理单位', width=0.5, palette='viridi
plt.title('知错不改 vs 处理单位', size=15)
plt.show()
```

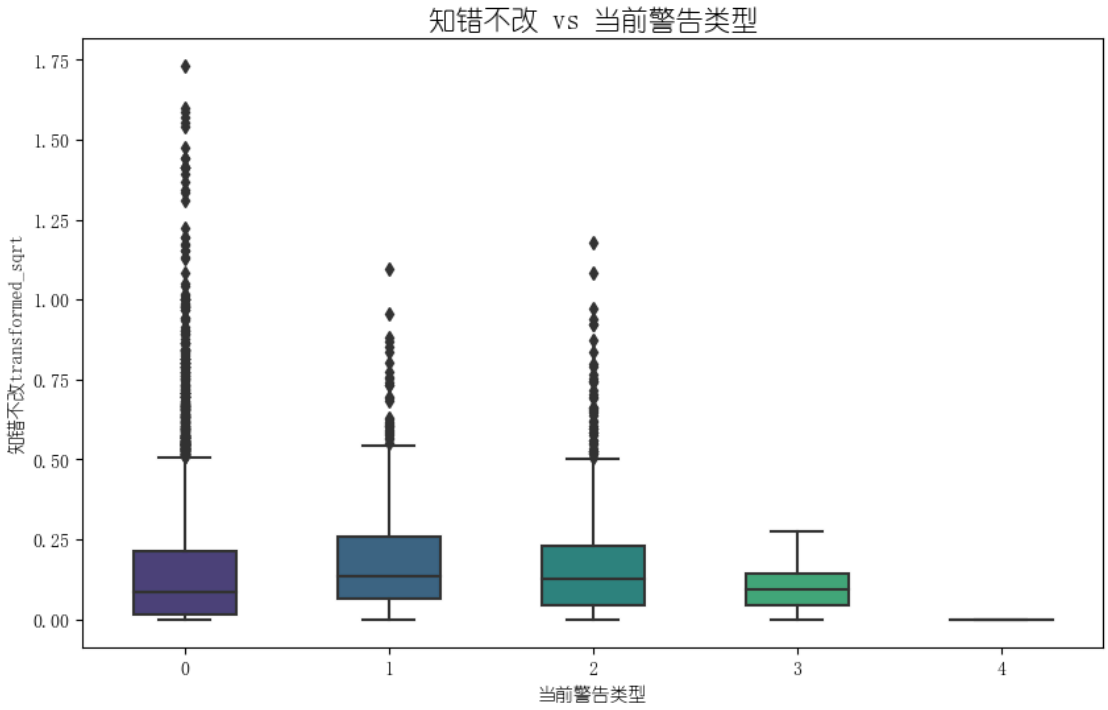


In [11]:
```
#plot 可视化 知错不改与处罚方式的关系
plt.figure(figsize=(10, 6))
sns.boxplot(x='处罚方式（总）', y='知错不改transformed_sqrt', width=0.5, palette='
plt.title('知错不改 vs 处罚方式', size=15)
plt.show()
```



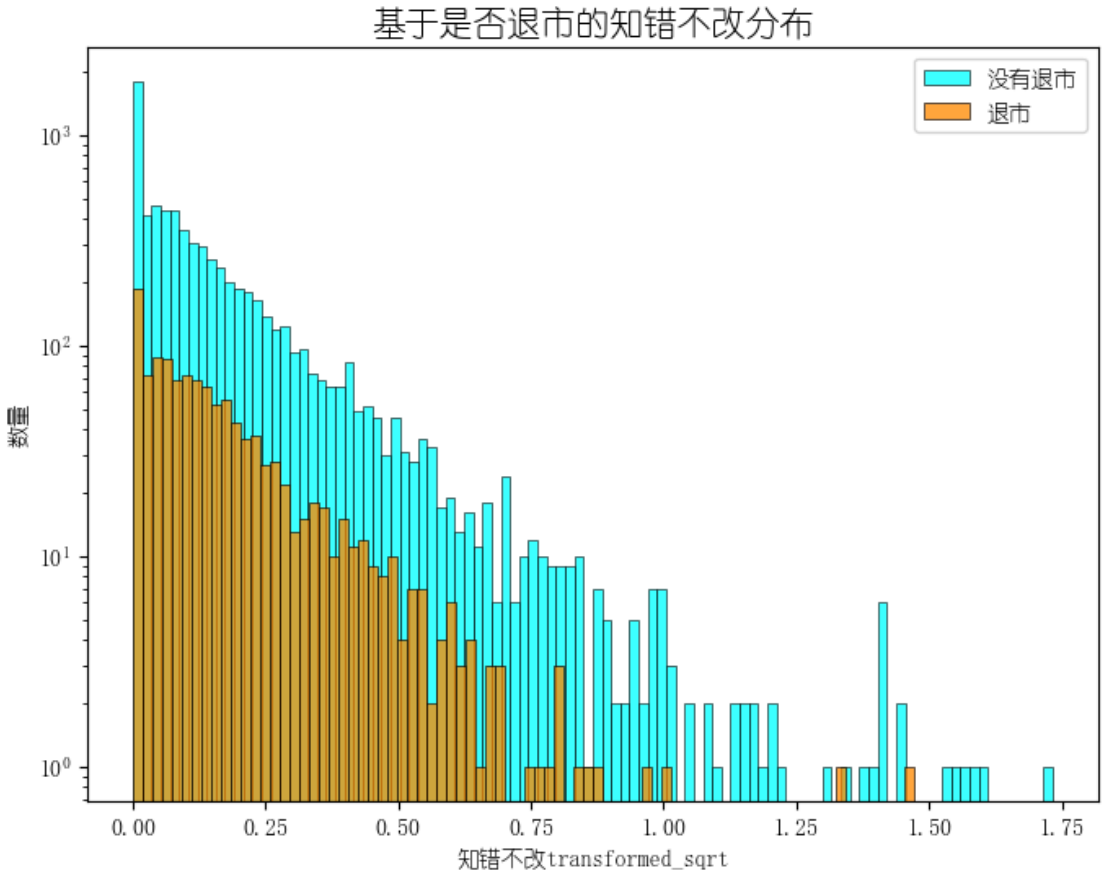
处罚方式越严重，公司改正动机会改善，知错不改降低。符合预期。

In [30]:
```
#plot 可视化 知错不改与当前警告类型的关系
# 无警示状态=0, ST=1,*ST=2,S*ST=3,PT=4
plt.figure(figsize=(10, 6))
sns.boxplot(x='当前警告类型', y='知错不改transformed_sqrt', width=0.5, palette='vi
plt.title('知错不改 vs 当前警告类型', size=15)
plt.show()
```



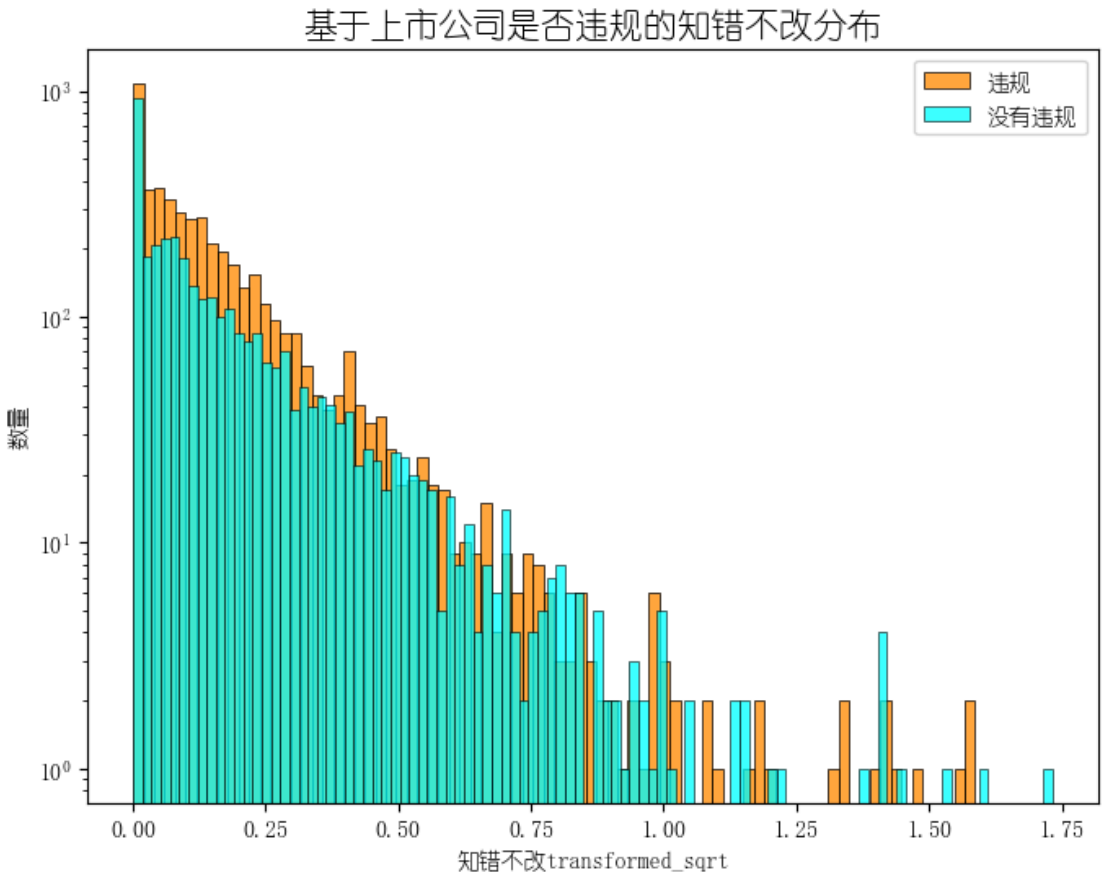
警示状态越严重，即从1-4公司持续犯同样错的动机就会逐渐减少。从侧面可以反应警示对于企业规范的作用。符合预期。

In [18]:
```
#plot 知错不改和是否退市
recommended = df.loc[df['最终是否退市'] == 1, '知错不改transformed_sqrt']
not_recommended = df.loc[df['最终是否退市'] == 0, '知错不改transformed_sqrt']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', label='没有退市',bins=100)
sns.histplot(x=recommended, color='#FF8800', label='退市',bins=80)
plt.title('基于是否退市的知错不改分布', size=15)
plt.ylabel('数量')
plt.yscale('log')
plt.legend()
plt.show()
```



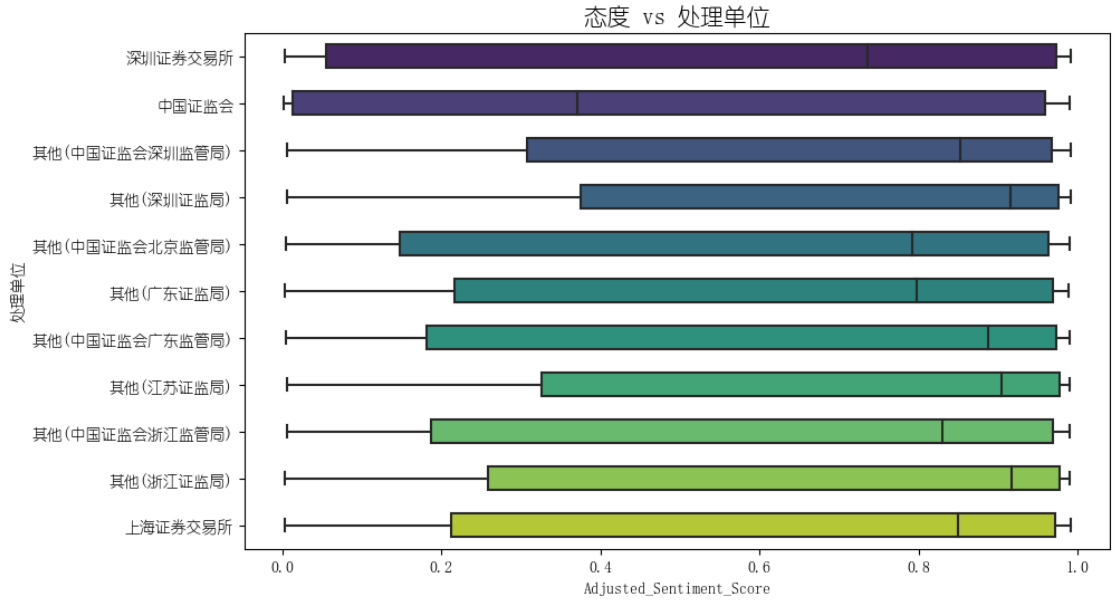
以退市为条件分类，并没有改变知错不改分布形状；下面的违规亦然。

In [19]:
```
#plot 知错不改和上市公司是否违规
recommended = df.loc[df['上市公司是否违规'] == 1, '知错不改transformed_sqrt']
not_recommended = df.loc[df['上市公司是否违规'] == 0, '知错不改transformed_sqrt']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#FF8800', label='违规',bins=80)
sns.histplot(x=not_recommended, color='#00FFFF', label='没有违规',bins=100)
plt.title('基于上市公司是否违规的知错不改分布', size=15)
plt.ylabel('数量')
plt.yscale('log')
plt.legend()
plt.show()
```



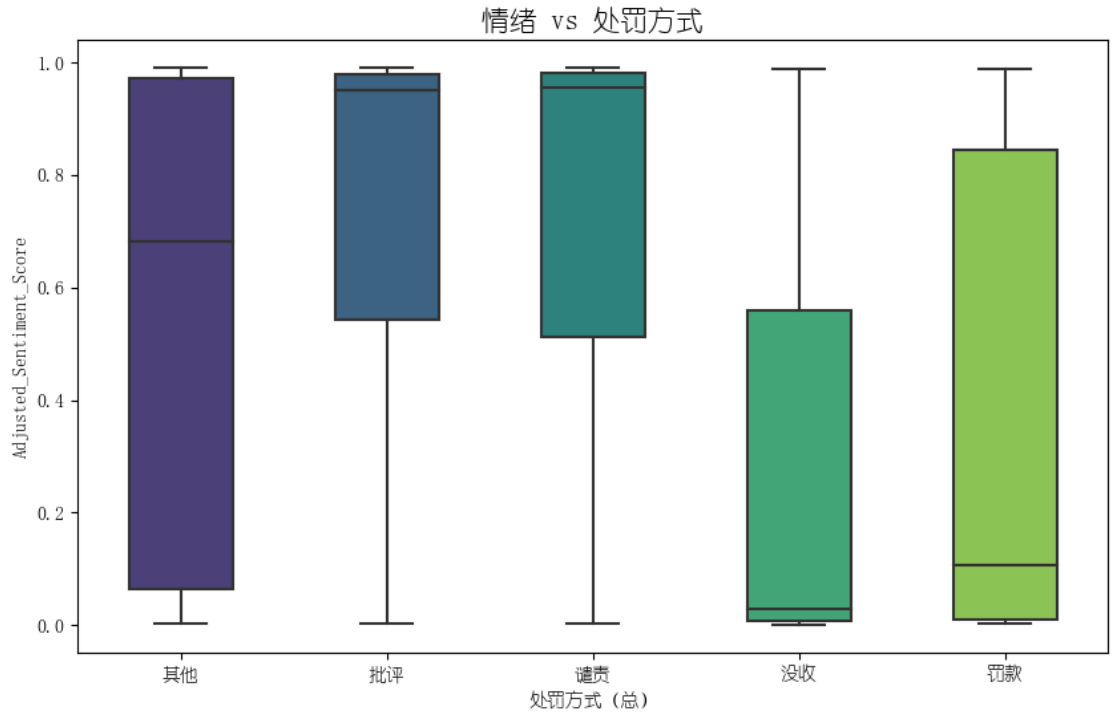
从趋势来看，违规与否不影响知错不改的向下趋势

In [22]:
```
#plot 可视化sentiment（情绪）与处理单位的关系
plt.figure(figsize=(10, 6))
sns.boxplot(x='Adjusted_Sentiment_Score', y='处理单位', width=0.5, palette='viridi
plt.title('态度 vs 处理单位', size=15)
plt.show()
```



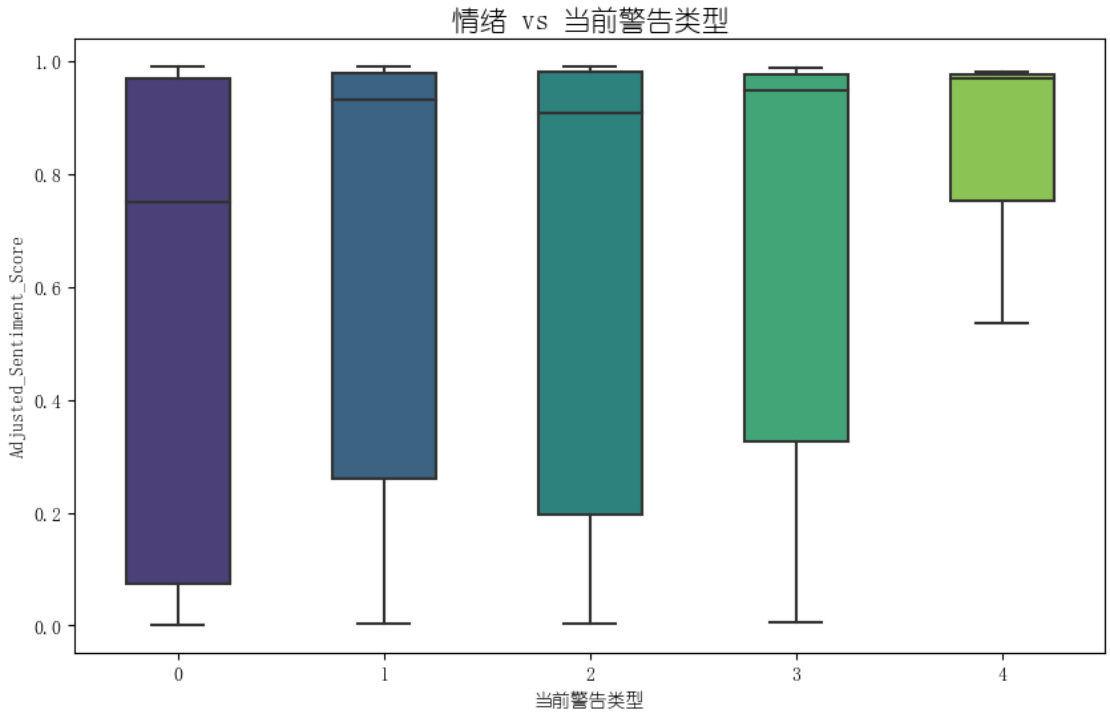
在所有机构中，中国证监会的情绪平均值较小，表达更加偏向于中性

In [28]:
```
#plot 可视化sentiment与处罚方式的关系
plt.figure(figsize=(10, 6))
sns.boxplot(x='处罚方式（总）', y='Adjusted_Sentiment_Score', width=0.5, palette='
plt.title('情绪 vs 处罚方式', size=15)
plt.show()
```



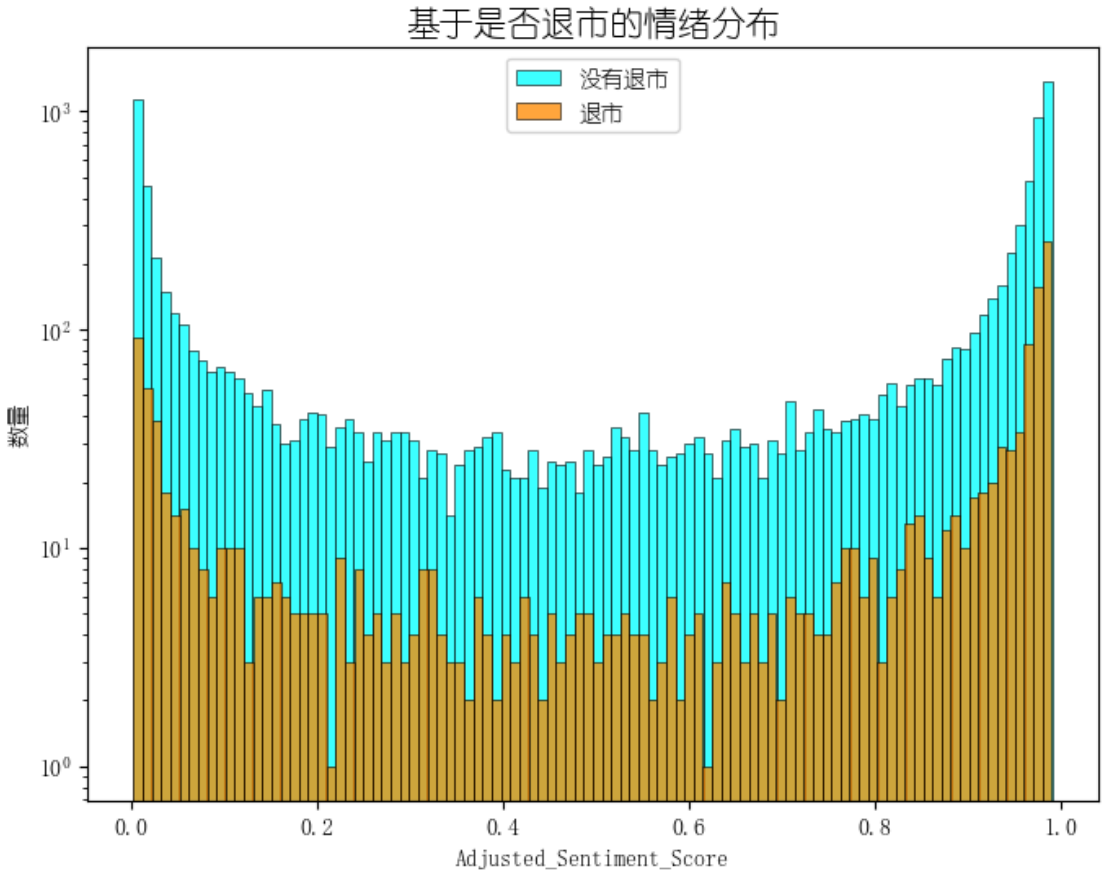
批评和谴责直观以为会比没收与罚款情绪相对温和。结果相反。

In [27]:
```
#plot 可视化sentiment与当前警告类型的关系
# 无警示状态=0, ST=1,*ST=2,S*ST=3,PT=4
plt.figure(figsize=(10, 6))
sns.boxplot(x='当前警告类型', y='Adjusted_Sentiment_Score', width=0.5, palette='vi
plt.title('情绪 vs 当前警告类型', size=15)
plt.show()
```



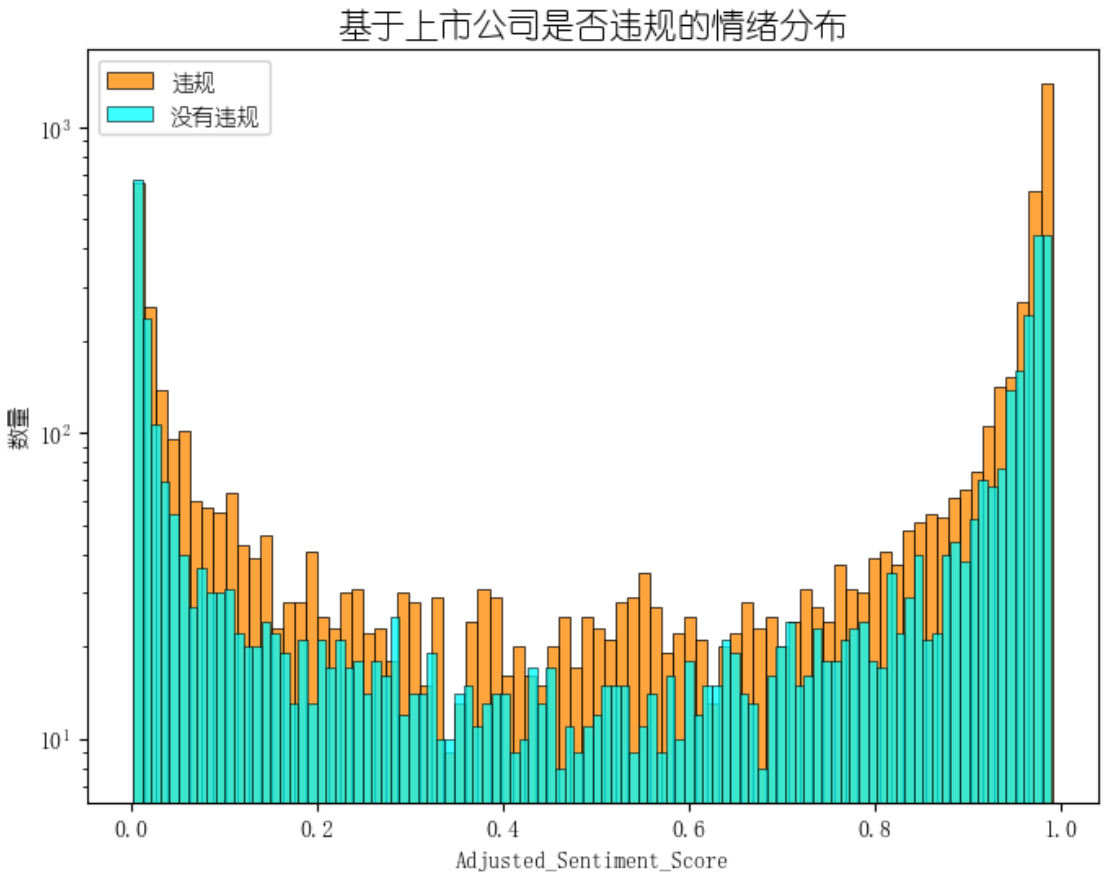
总体而言，警示状态越严重，情绪越负面，符合预期。

In [25]:
```
#plot sentiment和是否退市
recommended = df.loc[df['最终是否退市'] == 1, 'Adjusted_Sentiment_Score']
not_recommended = df.loc[df['最终是否退市'] == 0, 'Adjusted_Sentiment_Score']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', label='没有退市',bins=100)
sns.histplot(x=recommended, color='#FF8800', label='退市',bins=100)
plt.title('基于是否退市的情绪分布', size=15)
plt.ylabel('数量')
plt.yscale('log')
plt.legend()
plt.show()
```



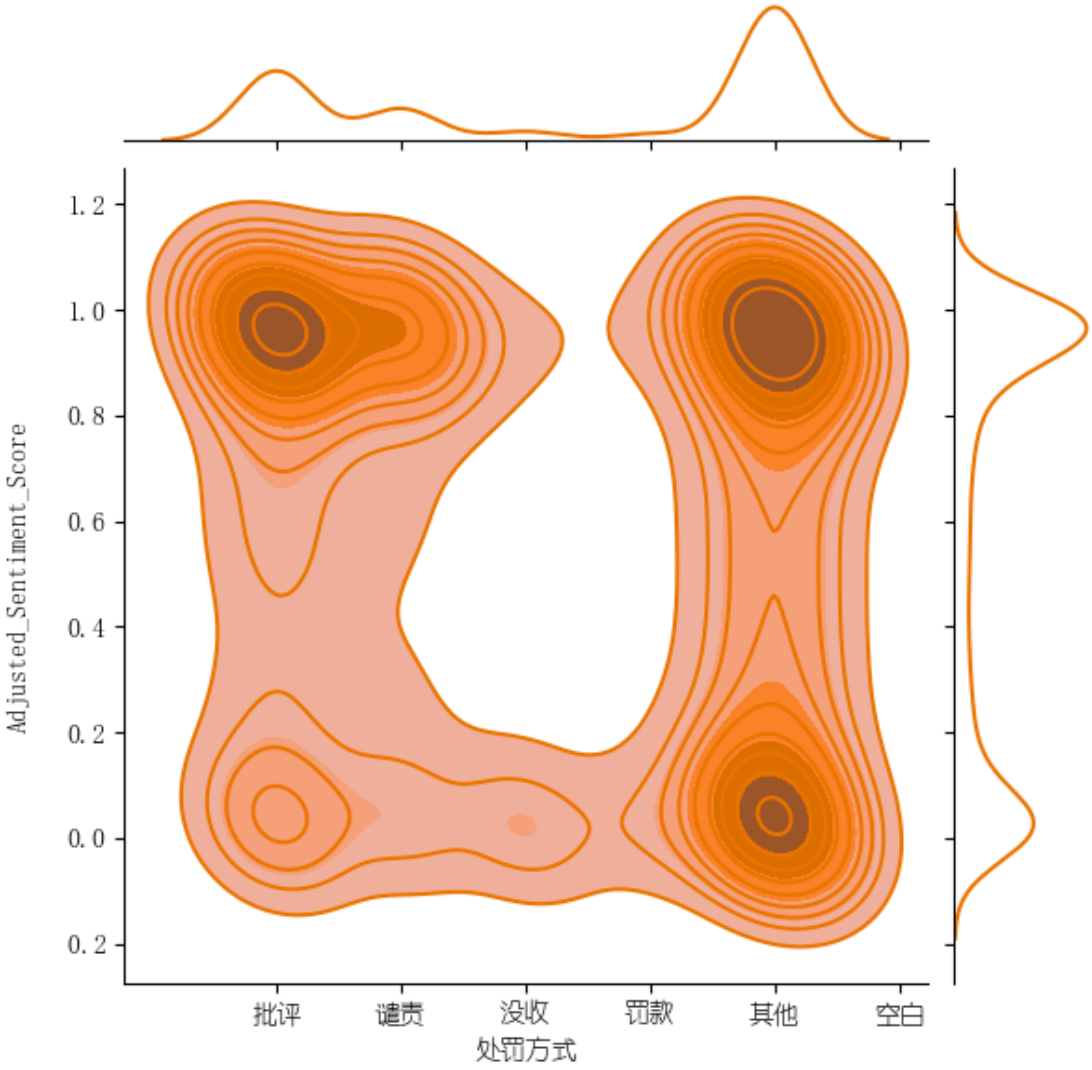
不考虑时间的情况下，退市的公司的公告情绪分布与未退市的没有大的不同，下面的违规亦然：

In [25]:
```
#plot sentiment和上市公司是否违规
# 无警示状态=0, ST=1,*ST=2,S*ST=3,PT=4
recommended = df.loc[df['上市公司是否违规'] == 1, 'Adjusted_Sentiment_Score']
not_recommended = df.loc[df['上市公司是否违规'] == 0, 'Adjusted_Sentiment_Score']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#FF8800', label='违规',bins=80)
sns.histplot(x=not_recommended, color='#00FFFF', label='没有违规',bins=100)
plt.title('基于上市公司是否违规的情绪分布', size=15)
plt.ylabel('数量')
plt.yscale('log')
plt.legend()
plt.show()
```

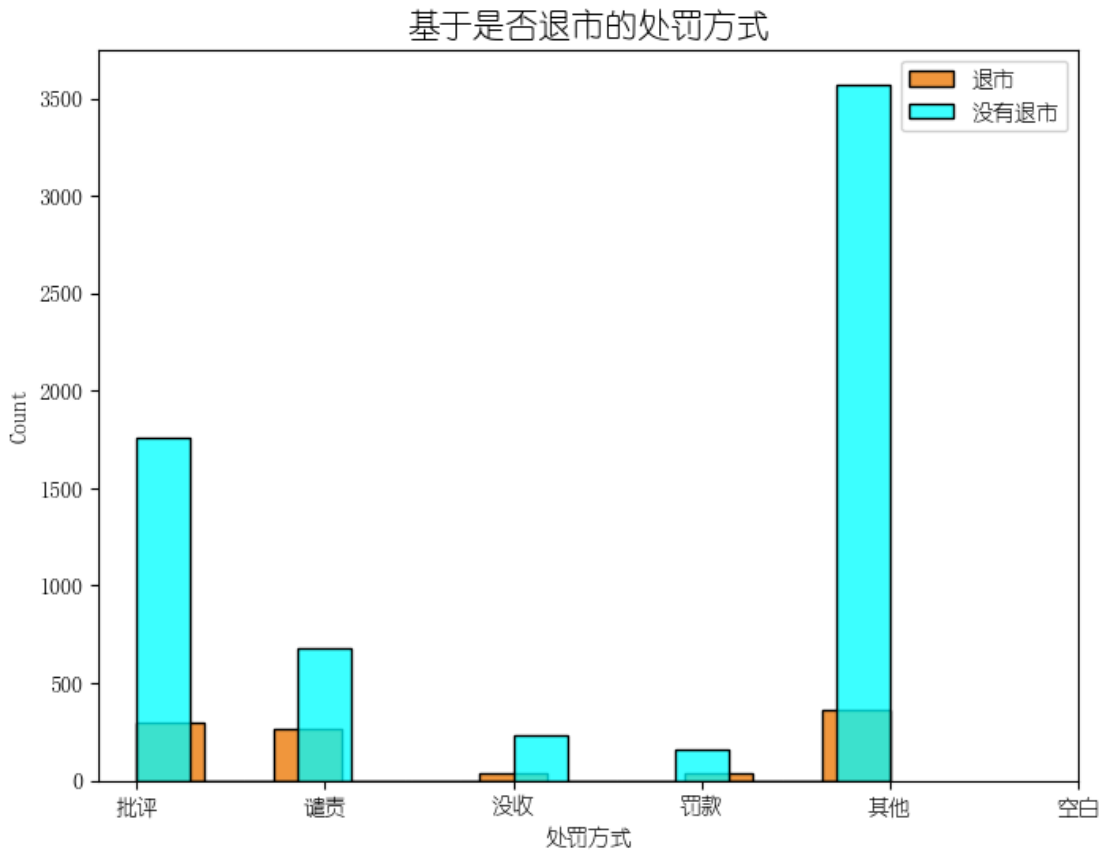


In [19]:
```
#plot sentiment和处罚方式
df['处罚方式'] = df['处罚方式（总）'].replace({'批评': 1.0, '谴责': 2.0, '没收': 3.
plt.figure(figsize=(8, 8))
g = sns.jointplot(x='处罚方式', y='Adjusted_Sentiment_Score', kind='kde', color='#
g.plot_joint(sns.kdeplot, fill=True, color='#EE7700', zorder=0, levels=6)

xtick_labels = ['批评', '谴责', '没收', '罚款', '其他', '空白']
g.ax_joint.set_xticks([1, 2, 3, 4, 5, 6])
g.ax_joint.set_xticklabels(xtick_labels)

plt.show()
```

```
<Figure size 800x800 with 0 Axes>

D:\anaconda\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph
8722 (\N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [21]:
```
#plot 基于是否退市的处罚方式
recommended = df.loc[df['最终是否退市'] == 1, '处罚方式']
not_recommended = df.loc[df['最终是否退市'] == 0, '处罚方式']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#EE7700', label='退市')
sns.histplot(x=not_recommended, color='#00FFFF', label='没有退市')

tick_positions = [1, 2, 3, 4, 5, 6]
tick_labels = ['批评', '谴责', '没收', '罚款', '其他', '空白']

plt.xticks(tick_positions, tick_labels)

plt.title('基于是否退市的处罚方式', size=15)
plt.legend()
plt.show()
```



从总的分布趋势来看，是否退市没有改变分布形态。两边多，中间少。下一个也是同样的情况，分布状态没有改变。

In [30]:
```
#plot 基于是否退市的违规行为字数分布
recommended = df.loc[df['最终是否退市'] == 1, '字数']
not_recommended = df.loc[df['最终是否退市'] == 0, '字数']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', kde=True, label='没有退市', binw
sns.histplot(x=recommended, color='#EE7700', kde=True, label='退市', binwidth=8)
plt.title('基于是否退市的违规行为字数分布', size=15)
plt.legend()
plt.show()
```