Literature review, research framework, innovations and descriptive statistics

I. Comb the relevant literature on Financial distress for reference

Data used to predict financial distress can be broadly categorized into two types: structured data (e.g., market data and financial data) and unstructured data (e.g., text, images, and web).

1. Using structured data to predict financial distress

Structured data usually involves the use of financial indicators as features. In recent years, scholars have also studied other aspects of the contribution of structured data.

Structured data to predict company distress. This can be further categorized into internal company data, external company data and mixed data.

Intra-firm data include factors such as corporate environment, innovation, capital structure and corporate governance.Pham et al. analyzed corporate distress in 41 countries and found a correlation between a better working environment and higher financial stability.Bai and Tian, in a bankruptcy prediction study of U.S. listed firms, found a negative correlation between innovation performance and the probability of bankruptcy.Ji et al. found that the digital financial development level is an effective indicator for predicting the bankruptcy risk of Chinese A-share listed companies. In terms of capital structure, the proportion of female directors, bank control over the firm, and sharp changes in capital structure due to leveraged buyout transactions are effective indicators for predicting corporate bankruptcy.Liang et al. find through stepwise discriminant analysis that the selection of six specific corporate governance indicators improves the predictive performance of financial distress, and that whether or not a firm adheres to the corporate governance code also affects the predictive performance. In addition, environmental, social, and governance ratings points, the initial compensation of new executives, and the extent of corporate tax avoidance all improve the accuracy of predicting corporate financial failure.

External data consist mainly of market and macroeconomic factors. Market-based distance-to-default models are usually good predictors of financial distress in the following year. Countries with efficient governments, high levels of business activity and strong controls on corruption can reduce the probability of corporate bankruptcy. Hybrid data refers to the integration of different aspects of indicators to obtain a higher level of predictive accuracy than would be possible with a single indicator alone. One approach is to combine data external to the firm with internal data, such as mixing market factors or corporate governance indicators with financial indicators and macroeconomic variables.

2. Using unstructured data to predict financial distress

Unstructured data generally refers to textual data, which is favored by most researchers because of the official and authoritative nature of the text of financial reports disclosed by companies.Ashraf et al. utilized the quality of financial reports as a feature.Wang et al. used a new stochastic subspace approach that combines sentiment and textual information for financial distress prediction.Li et al. used a self-constructed lexicon related to the financial domain to statistically extract sentiment and tone from positive and negative words in financial report texts as features for predicting company failures.Mai et al. used Word2Vec to transform Management Discussion and Analysis (MD&A) texts from financial reports into word vector matrices and used average weighting to obtain the document vectors as model inputs.Matin et al. further utilized convolutional neural networks to extract features from word vector matrix to extract features as inputs.Jiang et al. proposed a framework for predicting financial distress of unlisted public companies using current reports.Nguyen and Huynh used a financial sentiment dictionary and rule-based sentiment analysis to improve the classification performance of a corporate bankruptcy prediction model. In addition to official financial report texts, some scholars have used other types of textual data as features.Fedorova et al. measured the frequency of relevant keywords in Twitter posts as an indicator for assessing economic policy uncertainty.Zhao et al. used LM word lists as features for extracting sentiment tone from comments on online stock forums. In addition to textual data, there are also studies using images and networks for prediction.Hosaka extracted financial ratios from financial statements and represented them as grayscale images, and then used a GoogLeNet-based convolutional neural network for bankruptcy prediction of Japanese listed companies.Kou et al. demonstrated the importance of network variables based on transaction data and payments for bankruptcy prediction. Similar studies include the use of patent text information, analyst reports, and question and answer text information.

From the existing studies, it can be found that most of the current studies focus on analyzing the impact of certain text processing methods, such as Word2Vec and BERT, on financial distress prediction. There also exists a comparative analysis of word2vec and BERT, as well as a study on the effectiveness of improved methods based on word2vec and BERT in financial distress prediction. A study proposes four models: the Word2Vec-average model, the Word2Vec-weighted model, the BERT-word model, and the BERT-sentence model to test the effect of financial textual information on financial distress prediction. The study uses data from Chinese listed companies as a sample for model validation, and verifies the predictive performance of vectorized textual information under four different prediction horizons through cross-validation and sliding time window analysis.

II. Research framework, innovations in predicting delisting of listed companies in conjunction with disclosure violations

Previous literature utilizes unstructured data, either by extracting specific variables from text, such as emotions, or by combining raw features extracted from deep learning and merging them with numerical features. The latter studies are fewer and most of them achieve feature fusion by simple feature splicing. The latter model framework is shown in Fig:

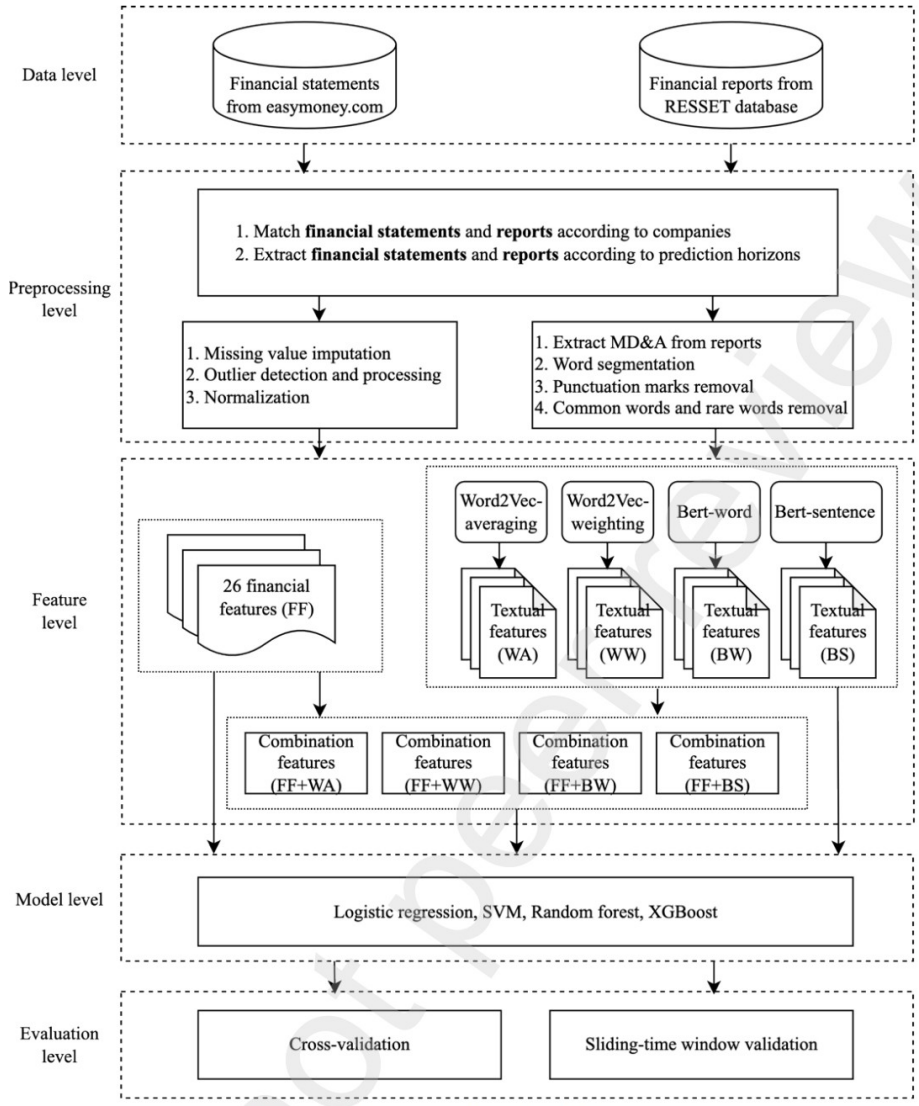
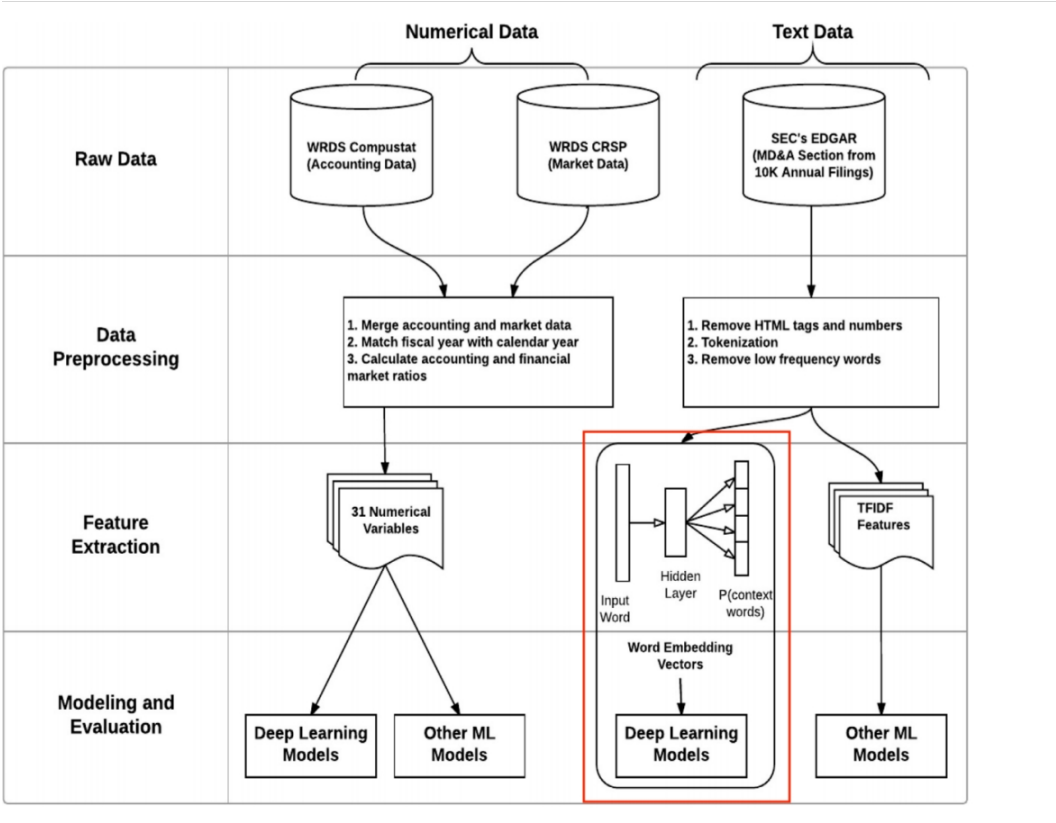


Figure 1 Framework and architecture

A feature of such an idea is that the model that merges different kinds of FEATURES is different from the model that ultimately makes the prediction. For example, in the example, the FEATURES level utilizes the more advanced BERT for preprocessing and feature extraction, while the MODEL level uses machine learning to make predictions from the spliced features. In addition, we observe that the Deep learning models for bankruptcy prediction using textual disclosures article predicts corporate bankruptcy without simple feature splicing, but rather, CNNs dealing with numeric features and CNNs dealing with text are merged in the hidden layer, i.e., model concatenation. Instead, the CNN dealing with numeric feature and the CNN dealing with text are merged in the hidden layer, i.e., the models are merged and directly perform the classification task. As shown in the red part of the icon:



Taking the above two papers as inspiration, we attempt to model merging bert with deep learning models to achieve merging at the internal model level and make predictions directly. Such a modeling paradigm is also currently very popular in the AI field, as such a model allows the model to process multiple types of data, merge at the model level and then classify them, including video, audio, pictures as well as text, etc. i.e., MULTIMODAL LEARNING. the research is mainly focused on emotion recognition as the core of the classification task.

The following is a compendium of literature on multimodal learning: Sentiment Analysis (SA) is highly regarded in the field of Artificial Intelligence (AI) and Natural Language. Natural Language Processing (NLP). There is a growing need to automatically analyze user sentiment about a product or service.

More and more people are sharing opinions online in the form of videos rather than just text. This has made sentiment analysis using multiple modalities, called multimodal sentiment analysis (MSA), an important area of research.MSA exploits recent advances in machine learning and deep learning at different stages, including multimodal feature extraction and fusion, and sentiment polarity detection, with the aim of minimizing the error rate and improving the performance.The recent developments in MSA architectures are classified into ten categories, i.e., Early Fusion, Late fusion, hybrid fusion, model-level fusion, tensor fusion, hierarchical fusion, bimodal fusion, attention-based fusion, quantum-based fusion, and word-level fusion.

BERT is now the main model in many real-world applications, employing large-scale corpora, including unsupervised and supervised corpora. Through unsupervised pre-training, BERT learns the structural and semantic information of the language; through supervised p r e - t r a i n i n g , BERT further learns specific NLP tasks, such as sentiment analysis, text categorization, and so on. This pre-training approach enables BERT to better understand the semantic information, thus improving its performance in various NLP tasks.

In the multimodal learning model, the main focus is on learning from sensory modalities such as auditory, visual, and textual in the multimodal learning
in which there are several branches of research. The MultiComp lab at Carnegie Mellon

University provides excellent taxonomies. Our problem falls into the category of so-called multimodal fusion - connecting information from two or more modalities for prediction. Since textual data is our primary modality, our review focuses on the literature that considers text as the primary modality and presents models that utilize converter architectures.

In terms of image and text transformations, Supervised Multimodal Bitransformers for Classifying Images and Text by Kiela et al. (2019) use pre-trained ResNet and pre-trained BERT features on unimodal images and text, respectively, and feed them into the Bitransformers. The key innovation is the tuning of image features as additional tokens to the transformer model. In addition, there are models - ViLBERT (Lu et al., 2019) and VLBert
(Su et al., 2020) - defines pre-training tasks for both images and text. Both models are pretrained on the Conceptual Captions dataset, which contains approximately 3.3 million image-caption pairs (with alternative text labeling).

(the Web image of the subject). In both cases, for any given image, a pre-trained target detection model (e.g., Faster R-CNN) obtains a vector representation of the image regions, which count as input marker embeddings for the transformer model. There is also LXMERT (Tan and Mohit 2019), which is another pre-trained transformer model implemented as part of the library starting with Transformers version 3.1.0.The inputs to LXMERT are the same as ViLBERT and VLBERT. However, LXMERT is pre-trained on an aggregated dataset, which also includes the visual quiz dataset. In total, LXMERT was pre-trained on 9.18 million image-text pairs.

In audio, video, and text conversion, including MulT, a multimodal converter for unaligned multimodal language sequences
(Tsai et al., 2019) and multimodal adaptive gates that integrate multimodal information into large pre-trained converters
(MAG) (Rahman et al., 2020).Wasifur Rahman et al. (2020) proposed an accessory to BERT and XLNet called Multimodal Adaptive Gate (MAG).MAG allows BERT and XLNet to accept multimodal nonverbal data during fine-tuning. It does this by generating transformations of the internal representations of BERT and XLNet; transformations conditional on visual and auditory modalities. The commonly used CMUMOSI and CMU-MOSEI datasets for multimodal sentiment analysis were investigated. Fine-tuning MAGBERT and magg -XLNet as well as linguistic fine-tuning of BERT and XLNet only significantly improved sentiment analysis performance compared to the previous baseline. On the CMUMOSI dataset, MAG-XLNet achieves the first human-level multimodal sentiment analysis performance in NLP. mulT is similar to ViLBert, where joint attention is used between pairs of modalities. mag injects information about other modalities at certain transformer layers through a gating mechanism.

In terms of text and knowledge graph embeddings, knowledge graph embeddings for document categorization are used to enrich BERT (Ostendorff et al., 2019) with features of author entities from the Wikidata knowledge graph in addition to metadata features for book category categorization. ernie (Zhang et al., 2019) matches tags in the input text with the knowledge graph's entities for matching. They fuse these embeddings to generate entity-aware text embeddings and text-aware entity embeddings with this matching.

*Specifically, for multimodal models more relevant to this thesis: Ken Gu et al. (2021) provide MultimodalToolkit,1 an open-source Python package for merging textual and tabular (categorical and numerical) data with a transformer for downstream applications. Its toolkit integrates well with hug Face's existing APIs, such as tokenization and model hub,2 allowing easy download of different pre-trained models.

In the existing literature we find a variety of existing models that enable feature fusion, where model level merging is as icon red:

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.

As per the results of literature combing, among them Wasifur Rahman Multimodal Adaptive Gate (MAG) and others are the most effective. Therefore we initially determined that the next practical step with the above seven methods is to conduct a comparative test.

In a word, the core innovation of multimodal learning is to merge different types of variables to be processed at the model level, and in the future, not only the features of the three categories of NUMBER CATEGORIC TEXT are merged and integrated, but also the information of audio, video, picture and so on can be added to join into the model, which can provide ideas for solving the problems of missing variables and imbalance, and so on. And, the current selection of

Models such as Wasifur Rahman Multimodal Adaptive Gate (MAG) have that adaptation.

On the practical side, recent research has utilized the above methodology to conduct three classification tasks on sexism that have strong technical relevance to our study (Vallecillo-Rodríguez, del Arco, Ureña-López, Martín-Valdivia, & Montejo-Ráez, 2009). 2023). The same MultimodalToolkit provided by Ken Gu et al. (2021) was used for their application.

1. System Overview

Delisting of listed companies is a categorization task. We propose two different architectures, the Base Architecture (BA) and the Digital Variable Integration. The digital information solution considers the information related to the text with the aim that the system takes into account the semantics of the text as well as the digital information.

1.1 infrastructure

The basic architecture is the Transformers architecture for text categorization. In this architecture, we tokenize the text and pass it to the model. The model preprocesses the input and generates the output. For text categorization, we obtain a categorization token from the last hidden state of the model. This token is intended to encode the entire input sequence. This marker is then passed as input to a feed-forward network, which classifies the text and generates classification results based on the number of classes required for each particular task. The two experiments we present, called "baseline" and "baseline with data augmentation", respectively, use this architecture.

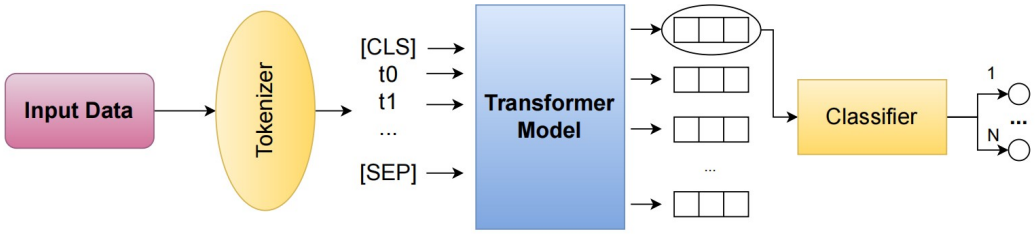


Figure 1: Base architecture proposed for EXIST shared task. N represents the number of output nodes and depends on the task because it corresponds to the number of labels to classify.

1.2 Integration of digital information

This design is a multi-modal architecture, called Transformer With Tabular, which receives a data frame as input. The data frame is preprocessed to extract the text, categorical and numerical information representing the text to be categorized. Columns. The textual features are then tokenized, the categorical features are encoded, and the numerical features are converted into appropriate grid
style. The text features are then passed to the Transformer model.The output of the Transformer model is passed to a combination module that combines the digital and categorical features to generate a unique tensor, which is then passed to the classifier to generate the final prediction. In this case in the figure, there are no numerical features, so this architecture is not shown in Figure 2. named "Proposed architecture to incorporate annotator information for EXIST shared task".

The architecture's combinatorial module represents a strategy for combining different features. In this case, it explores the following approaches:

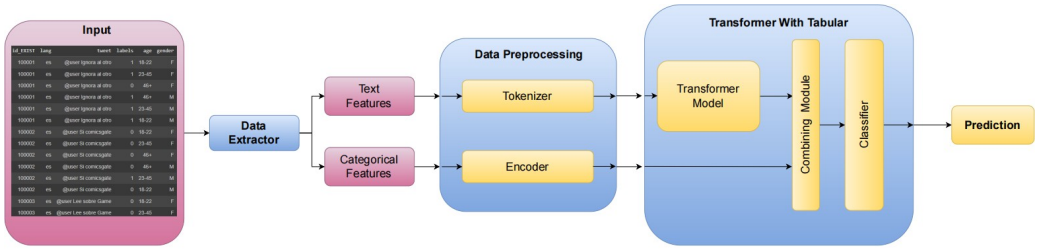


Figure 2: Proposed architecture to incorporate annotator information for EXIST shared task.

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

2. data selection

Combination of digital and textual information.

2.1 Digital information component: financial data, market data

Based on the summary in Mai 2020, we initially use this as our standard:

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 text message

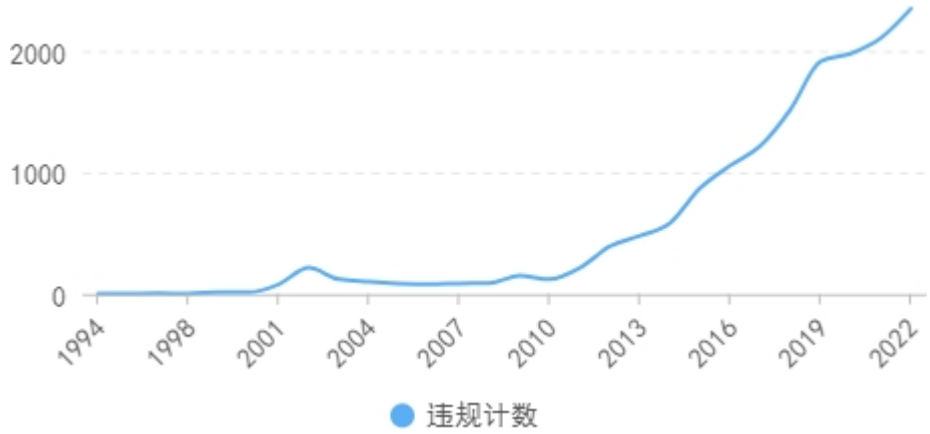
The violation data are the categorical and numeric variables in the existing dataset as well as the text in the violations, and for the text part, since it is different from the usual literature that uses the MD&A text in the annual reports, we do a more comprehensive descriptive s t a t i s t i c s , in which we purposely use the bert model fine-tuned by financial-news-Chinese to extract the text sentiment and add it to the descriptive statistics to improve our understanding of the text content in this dataset. Chinese fine-tuned bert model to extract the text sentiment and add it into the descriptive statistics, so as to improve our understanding of the text content of the dataset.

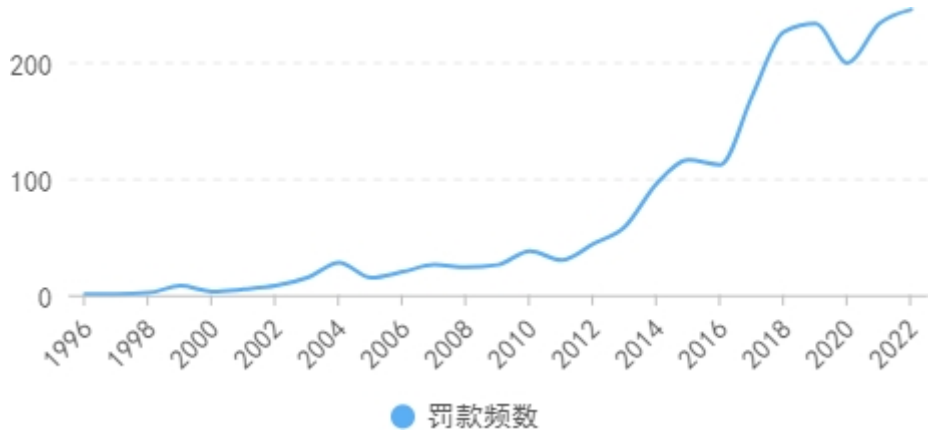
III. Descriptive statistics

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/Liu Chengxin/Desktop/Thesis--Exit Alert/Dataset/Visualization/Visualization20
```

Let's first look at the overall relationship between delistings, violations, and fines: all are found to be trending upward:

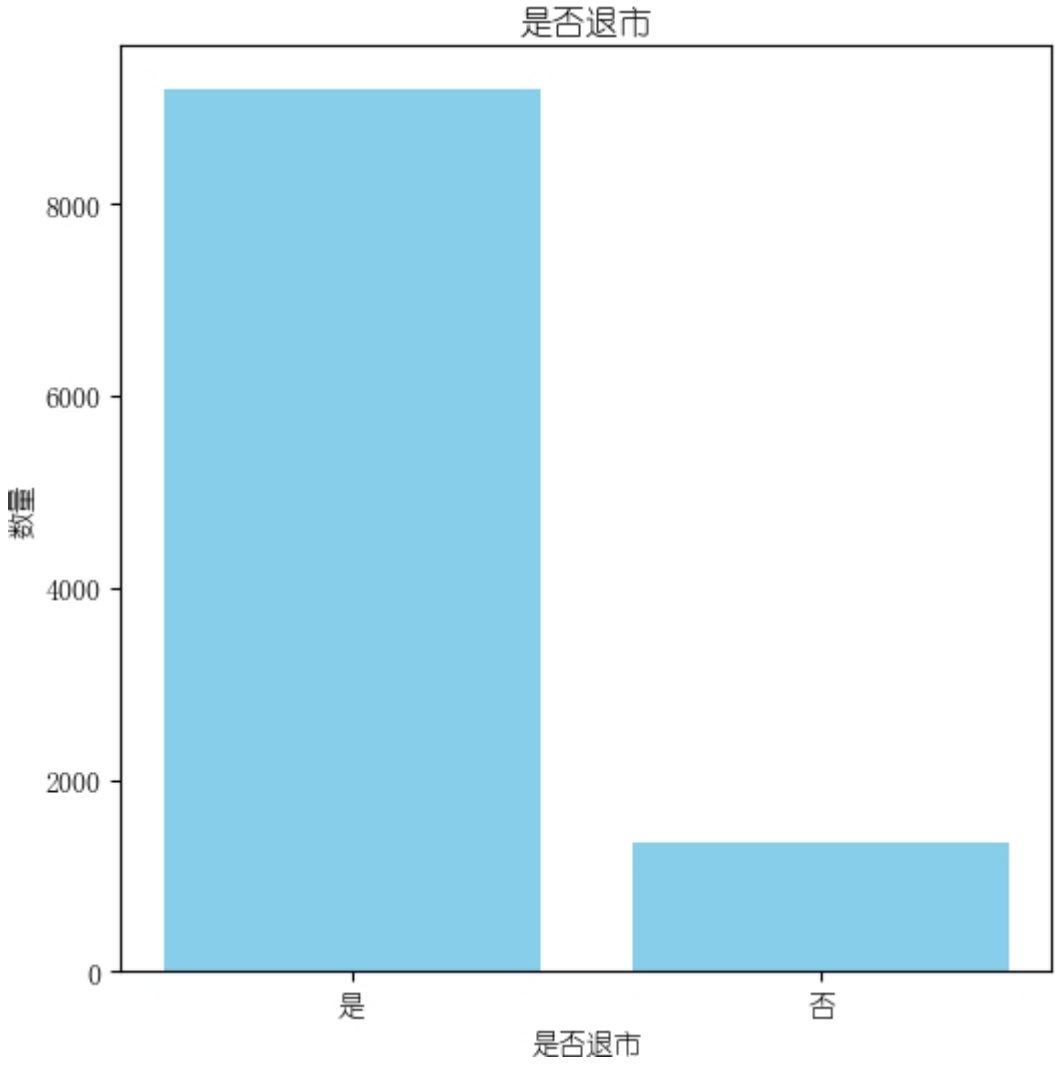




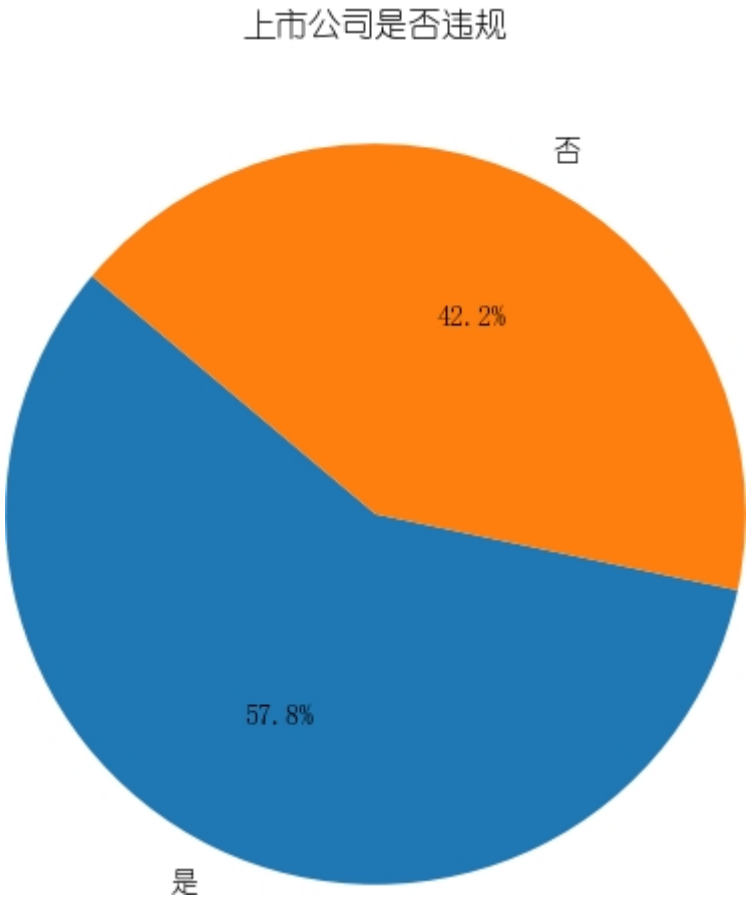
Take a look at the statistics from a total frequency perspective:

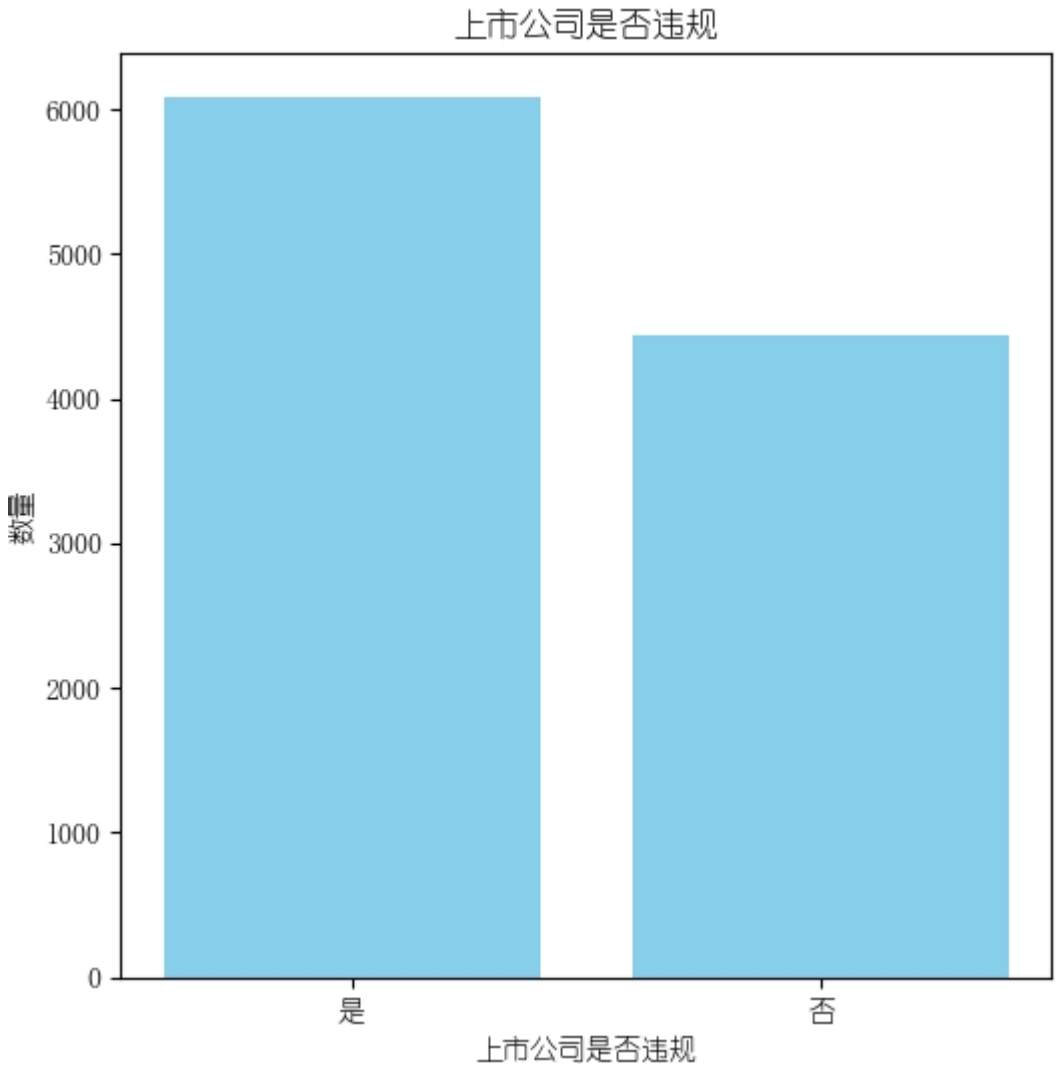
In [3].

```
#plot "whether to delist"
counts = df['Whether or not to eventually delist'].value_counts()

# plt.figure(figsize=(6, 6))
# plt.pie(counts, labels=['yes', 'no'], autopct='%1.1f%%', startangle=140)
# plt.title('Whether to delist')
# plt.show()

plt.figure(figsize=(6, 6))
bars = plt.bar(['yes', 'no'], counts,
color='skyblue') # plt.xticks(rotation=45,
ha='right') plt.title('Whether to delist')
plt.xlabel('Whether
delisted')
plt.ylabel('Number')
plt.show()
```



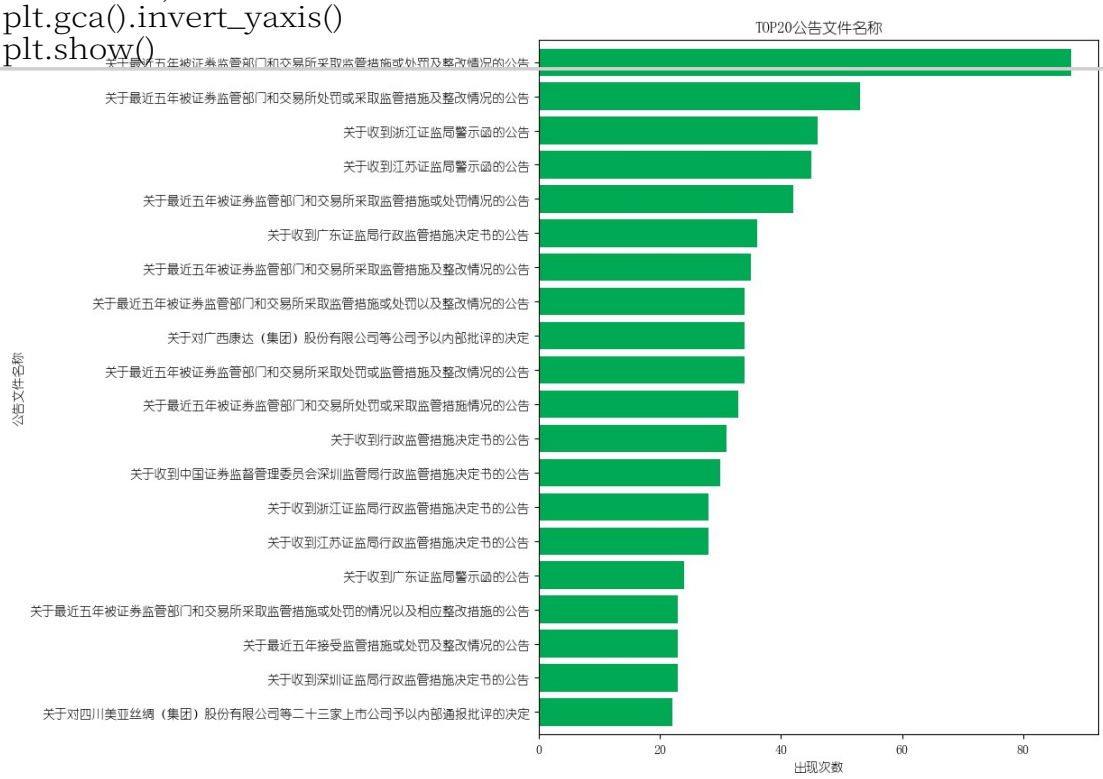
In [4]:
```
#plot "Is it a violation"
counts = df['Is the listed company in violation'].value_counts()

plt.figure(figsize=(6, 6))
plt.pie(counts, labels=['yes', 'no'], autopct='%1.1f%%', startangle=140)
plt.title('Whether the listed company is violating the law')
plt.show()

plt.figure(figsize=(6, 6))
bars = plt.bar(['yes', 'no'], counts, color='skyblue')
# plt.xticks(rotation=45, ha='right')
plt.title('Whether the listed company violated the
law')    plt.xlabel('Whether the listed company
violated the law')
plt.ylabel('Number')
plt.show()
```





The following is a formal correlation visualization around the variables in the violation dataset, with negative sentiment indicators in the text extracted using bert:

In [5]:
```
# plot "Announcement file name" top20
file_name_column = 'Announcement file name'
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
Announcement File Name')
plt.xlabel('Number of
Occurrences')
plt.ylabel('Announcement
File Name')
plt.gca().invert_yaxis()
plt.show()
```



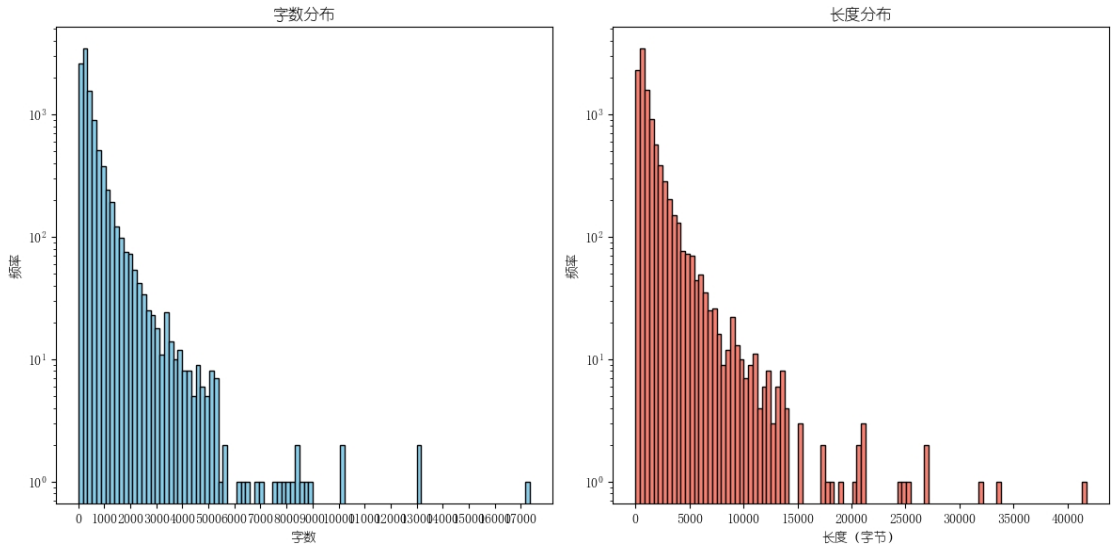
In [6].

```
# plot Violation word count, length, word frequency
text_column = 'Violation'

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 = R e v i e w T e x t .str. replace('(\xa0)', '')
    ReviewText = ReviewText.str.replace('\n', '')
    return ReviewText

df['infraction'] =

preprocess(df['infraction']) texts =

df[text_column].astype(str) df['word

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

max_text_length = df['length'].max()
max_word_count = df['word count'].max()

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(df['word count'], bins=100, color='skyblue', edgecolor='black')
plt.title('Word count distribution')
plt.xlabel('Number of words')
plt.ylabel('Frequency')
plt.xticks(np.arange(0, max_word_count, 1000))
plt.yscale('log')

plt.subplot(1, 2, 2)
plt.hist(df['length'], bins=100, color='salmon', edgecolor='black')
plt.title('Length distribution')
plt.xlabel('Length in bytes')
plt.ylabel('Frequency')
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)

# Delete some unimportant words
texts = texts.str.replace('of', '')
texts = texts.str.replace('year', '')
texts = texts.str.replace('month',
'') texts = texts.str.replace('day',
'') # texts = texts.str.replace('in',
'') # texts = texts.str.replace('not
yet', '')
# texts = texts.str.replace('million
dollars', '') # texts =
texts.str.replace('for', '')
# texts = texts.str.replace('You', '')
texts = texts.str.replace('The', '')
texts = texts.str.replace('The', '')
```

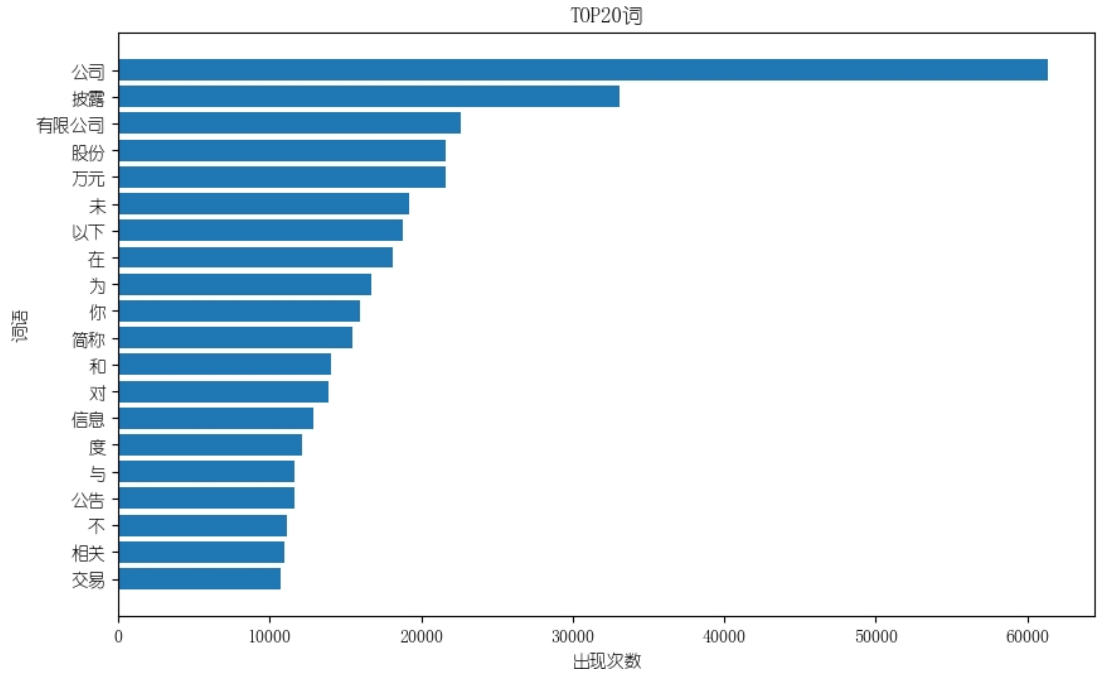
```
texts = texts.str.replace('The', '')
```

```
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))  # Take the top20
plt.figure(figsize=(10, 6))
plt.barh(words, counts)
plt.xlabel('number of
occurrences')
plt.ylabel('words')
plt.title('Top 20 words')
plt.gca().invert_yaxis() plt.
show()
```

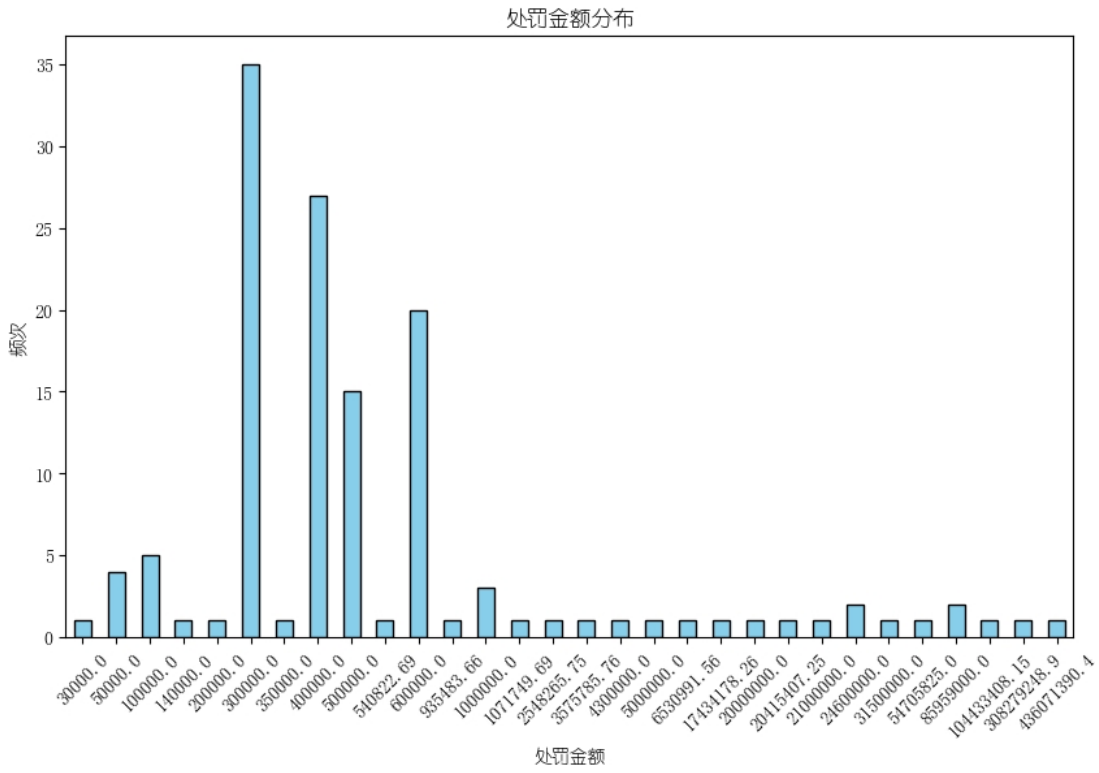
```
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:5:  FutureWarnin
g: The default value of regex will change from True to False in a future versio n.
  ReviewText = ReviewText.str.replace("(<br/>)", "")
C:\Users\17629\AppData\Local\Temp\ipykernel_13852\2349425165.py:6. FutureWarnin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

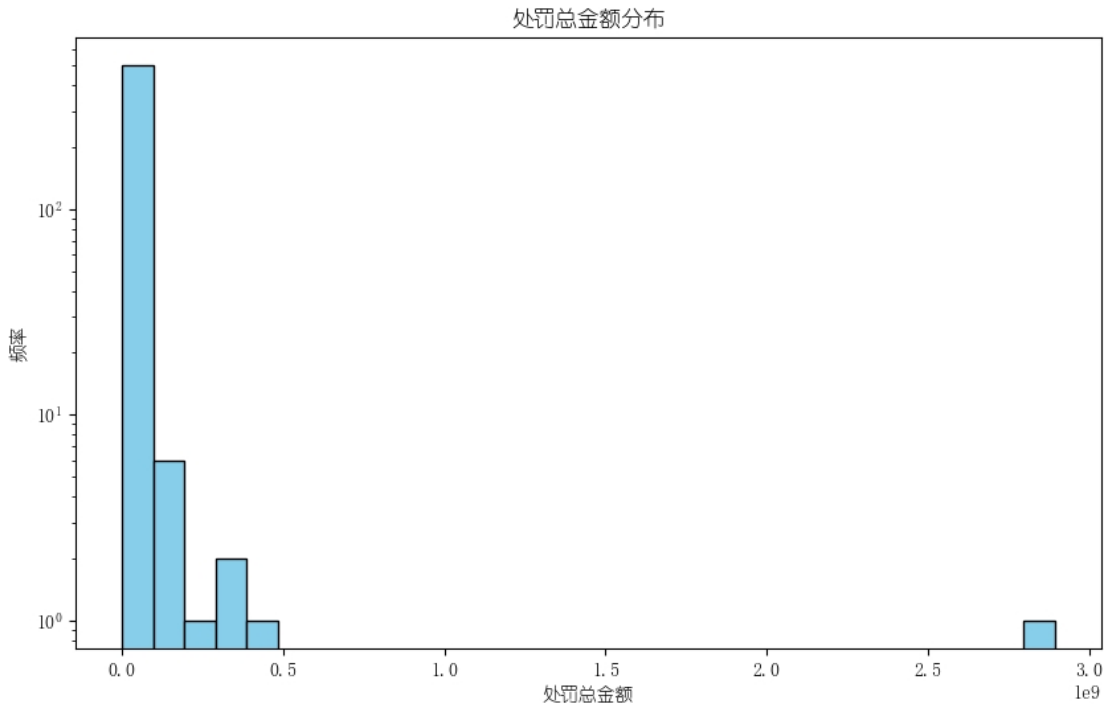


In [7]:
```
# plot "Penalty amount - listed companies" distribution
punishment_amount_column = ' Punishment Amount - Listed Companies'
numeric_values = pd.to_numeric(df[punishment_amount_column], errors='coerce').drop

plt.figure(figsize=(10, 6))
counts =  numeric_values.value_counts().sort_index()
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribution of penalty amounts')
plt.xlabel('penalty amount')
plt.ylabel('frequency')
plt.xticks(rotation=45, ha='center')
plt.show()
```



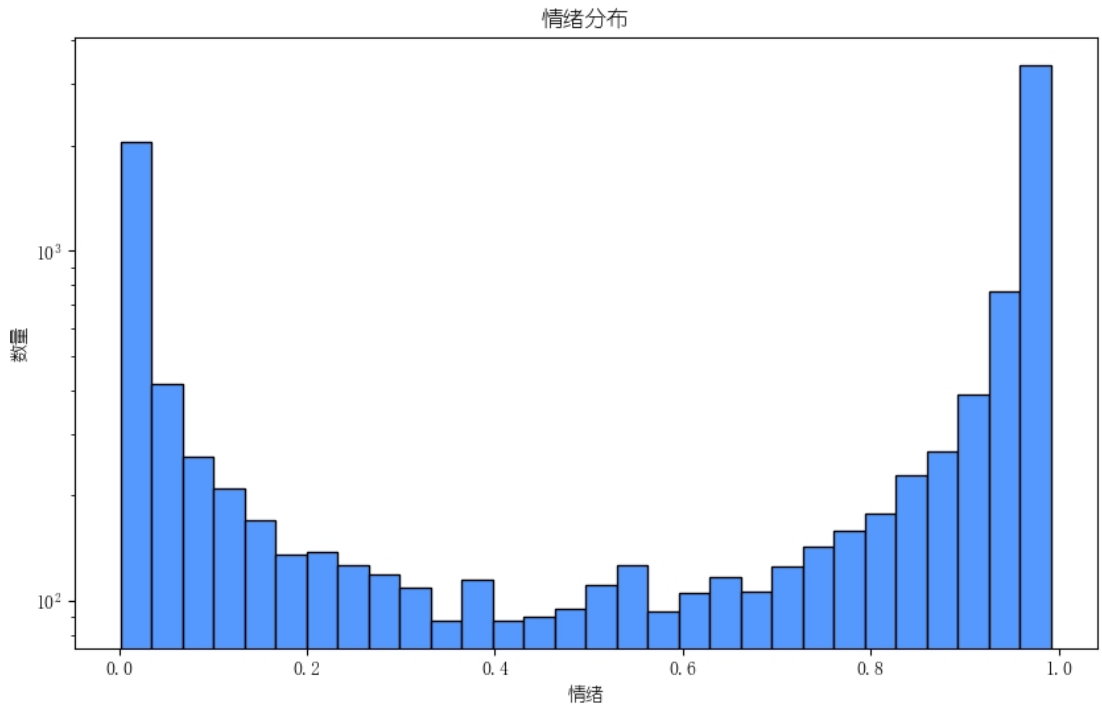
In [8].

```
# plot "total penalties - listed companies" distribution
punishment_column = 'Total amount of punishment'
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('Distribution of penalty totals')
plt.xlabel('total amount
of penalties')
plt.ylabel('frequency')
plt.yscale('log')
plt.show()
```



Sentiment variable is set at (0, 1), default is all NEGATIVE, value = 0, sentiment is neutral

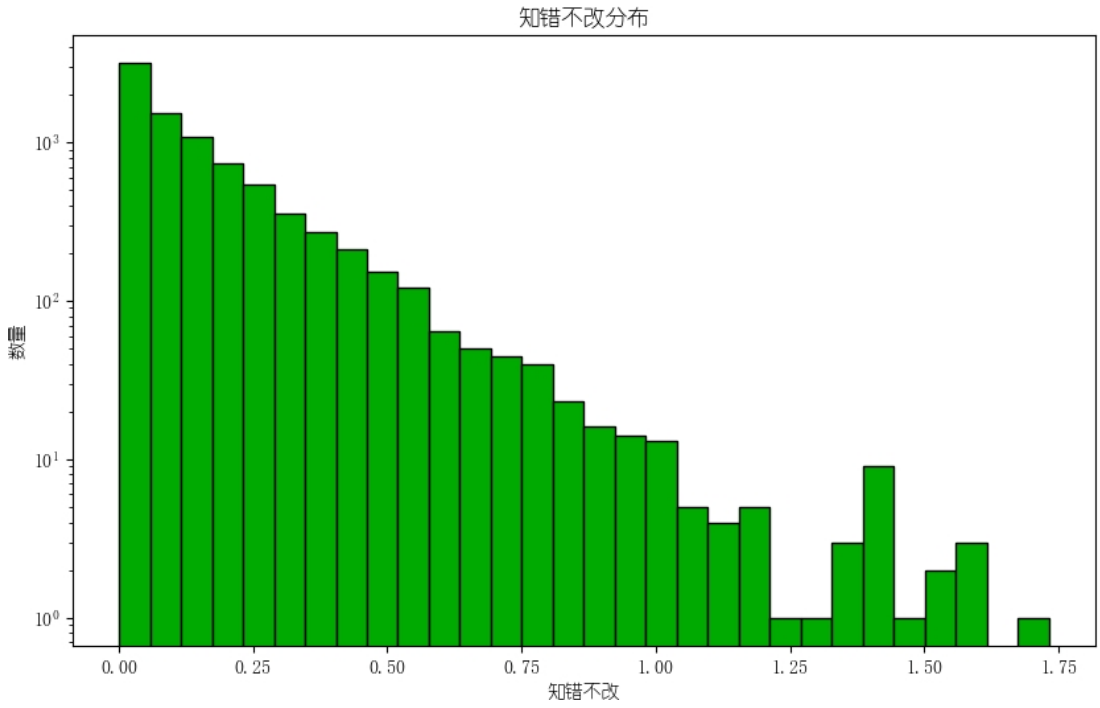
In [9]:
```
#plot sentiment distribution
punishment_column = ' Adjusted_Sentiment_Score'
numeric_values = pd.to_numeric(df[punishment_column], errors='coerce').dropna()

plt.figure(figsize=(10, 6))
plt.hist(numeric_values, bins=30, color='#5599FF', edgecolor='black')
plt.title('Sentiment distribution')
plt.xlabel('Emotions')
plt.ylabel('numb
er')
plt.yscale('log')
plt.show()
```

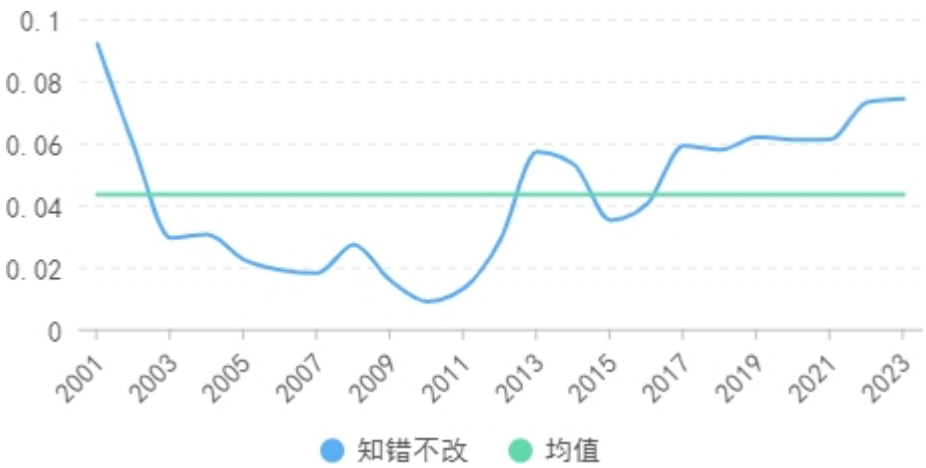


Definition: The greater the cumulative number of times a company repeats a particular type of violation at a given point in time, the greater the ratio of all violation types involved, and the greater this value.

In [10].

```
#plot Distributions that don't change their mistakes
punishment_column = 'know_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('Distribution of known errors')
plt.xlabel('know
wrong')
plt.ylabel('number')
plt.yscale('log')
plt.show()
```

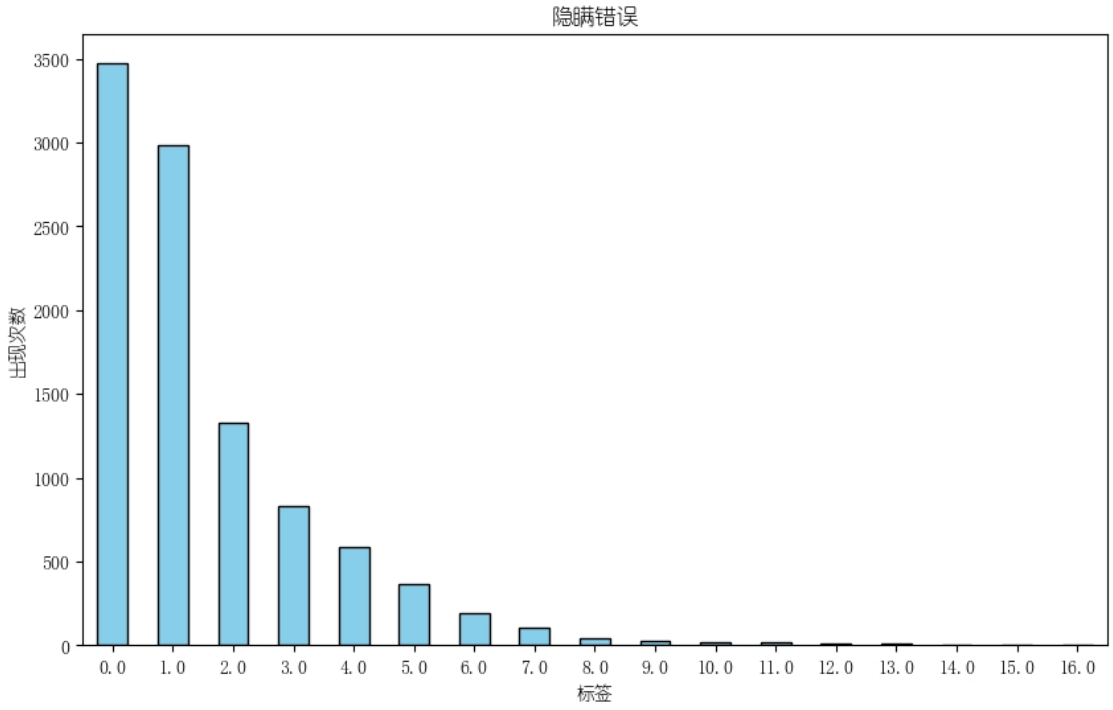


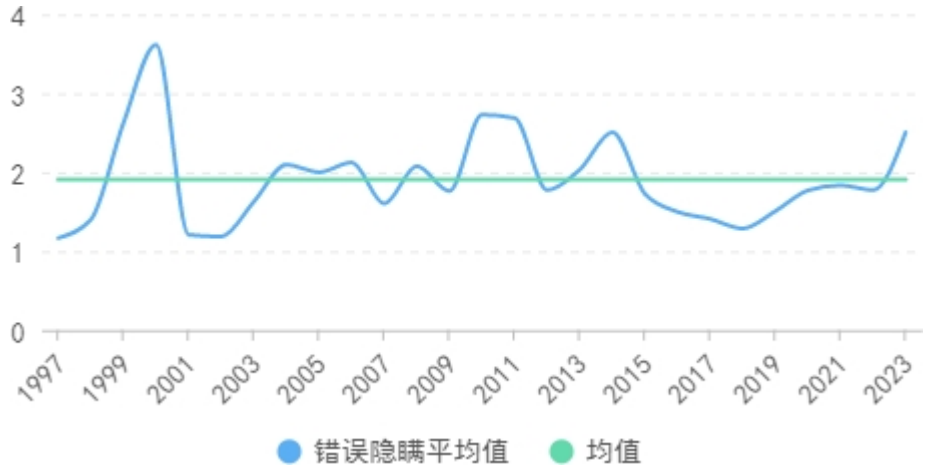


Definition of concealment of error: difference between the actual time the violation was committed and the time it was officially disclosed (in years)

In [11].

```
# plot "Hide the mistake"

label_column =  'Hide errors'
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('Concealmen
t error')
plt.xlabel('Label')
plt.ylabel('Number of
occurrences')
plt.xticks(rotation='horizontal',ha='center')
plt.xticks(integer_counts.index.asype(int)) plt.show()
```

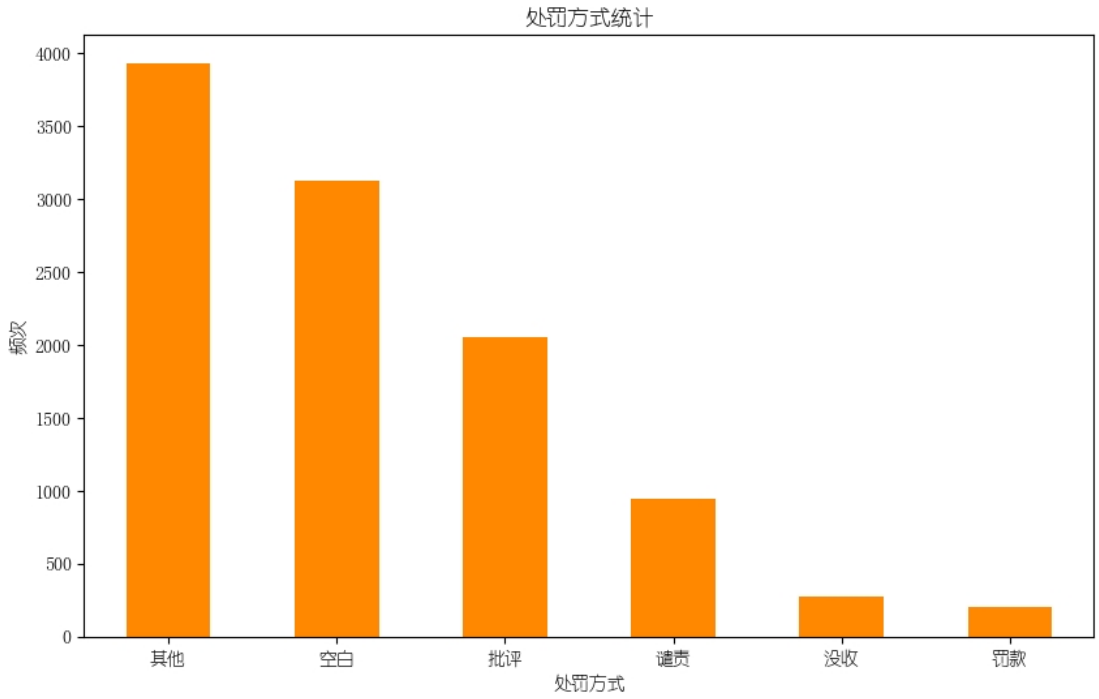


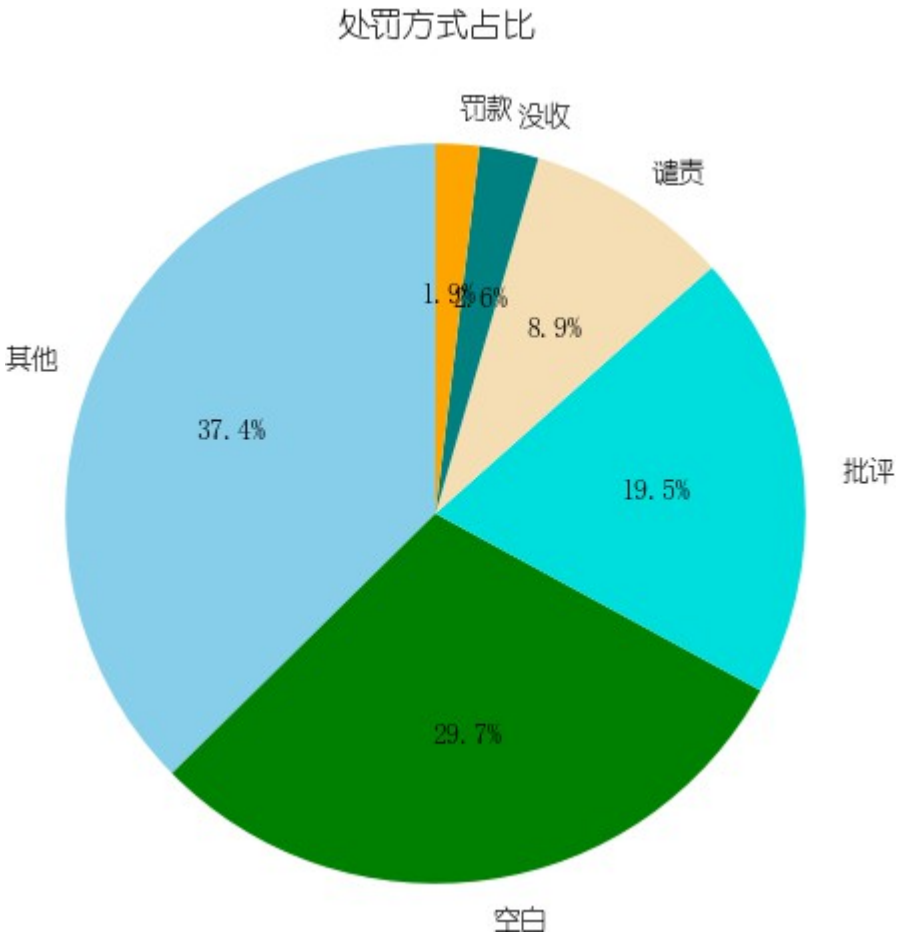


Fluctuates around two years, which means that on average for all firms, the time lag between a historical violation and being dealt with by regulators is constant around two years

In [12].

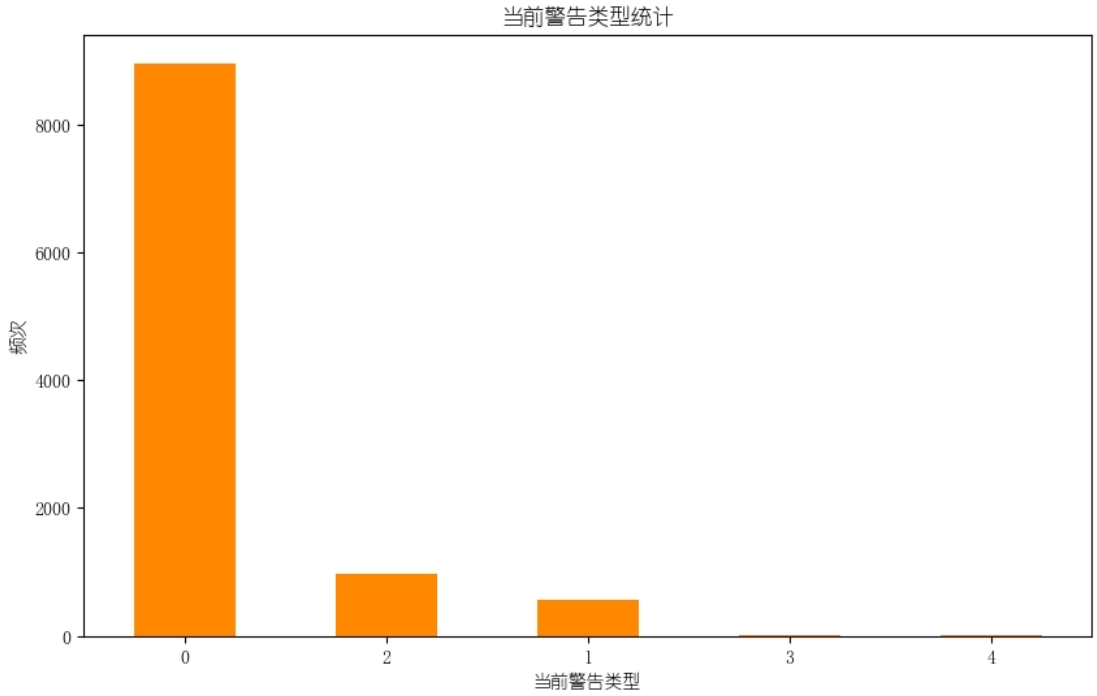
```
# plot "Penalty modalities"
punishment_column = 'Type of punishment (total)'

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

text_counts = df[punishment_column].value_counts()

plt.figure(figuresize=(10, 6))
text_counts.plot(kind='bar', color='#FF8800')
plt.title('Penalty Mode Statistics')
plt.xlabel('Penalty Mode')
plt.ylabel('frequency')
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('Percentage of Penalty Methods')
plt.ylabel('')
plt.show()
```



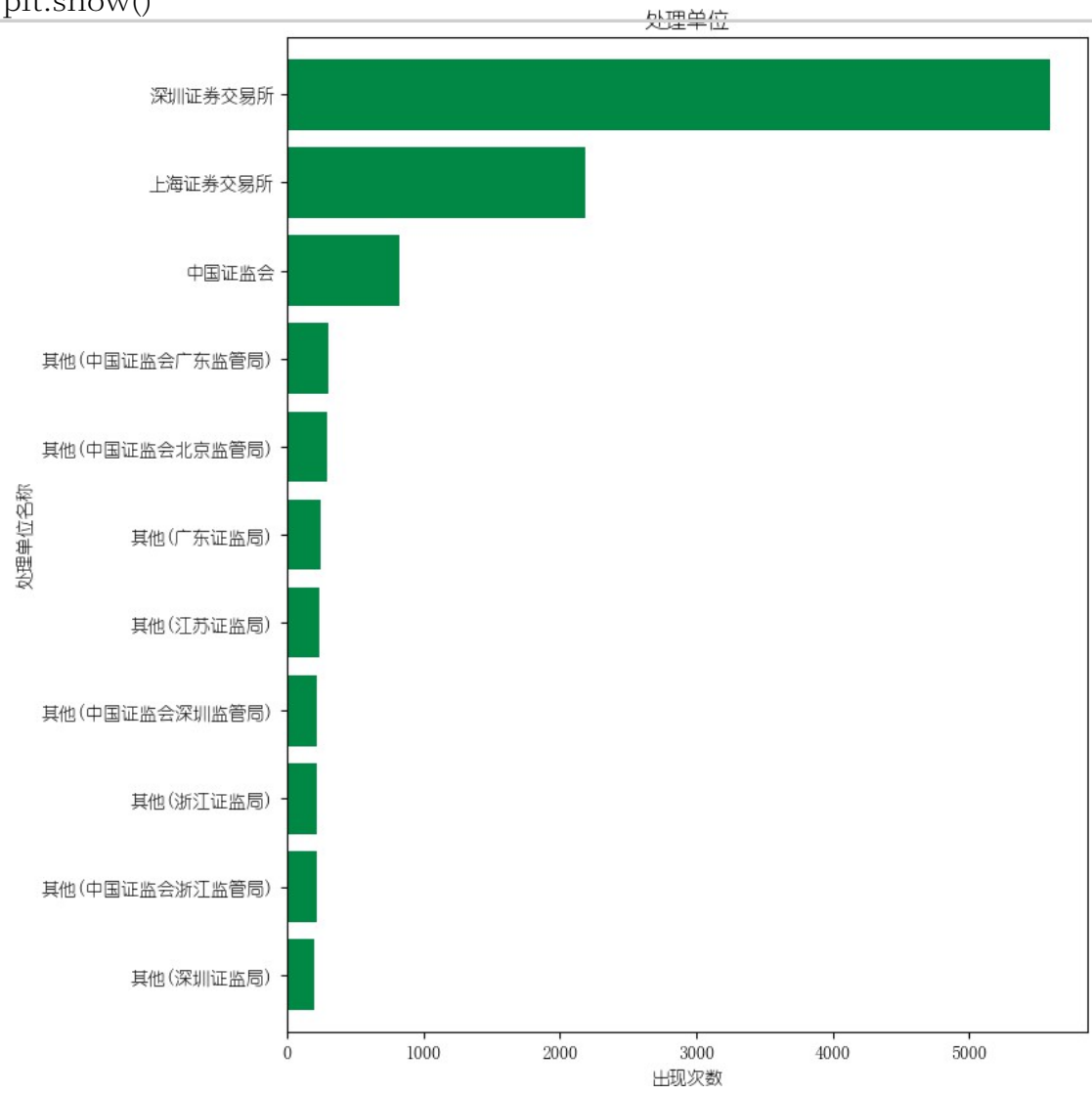

处罚方式占比
罚款
没收
谴责
批评
空白
其他
1.9%
2.6%
8.9%
19.5%
29.7%
37.4%

In [31].

```
# plot "Current warning type"
# No warning status = 0, ST=1,*ST=2,S*ST=3,PT=4
punishment_column = 'Current warning type'

df[punishment_column].fillna(1 , inplace=True) # treat empty as 0

text_counts =  df[punishment_column].value_counts()

plt.figure(figsize=(10,  6 ))
text_counts.plot(kind='bar', color='#FF8800')
plt.title('Current warning type
statistics') plt.xlabel('Current
warning type')
plt.ylabel('Frequency')
plt.xticks(rotation=0, ha='center')
plt.show()
```



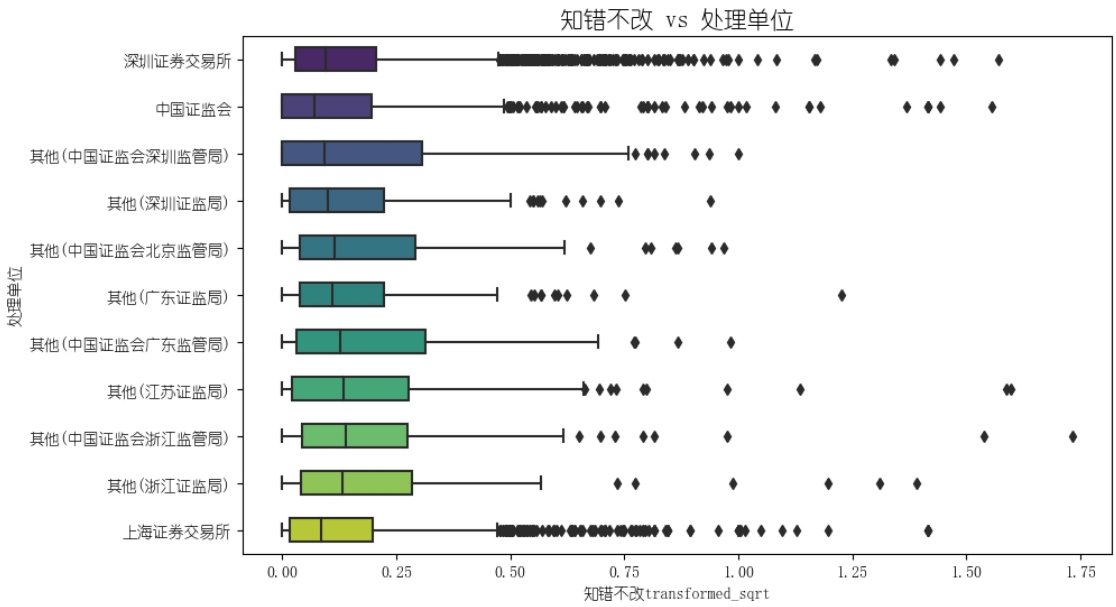
In [9]:
```
#plot "penalty units"
file_name_column = 'Processing unit'
text_values = df[file_name_column].astype(str)

word_counts = text_values.value_counts()

plt.figure(figsize=(8, 10))
word_counts.plot(kind='barh', width=0.8, color='#008844')
plt.title('Processing unit')
plt.xlabel('Number of
occurrences')
plt.ylabel('Processing unit
name')
plt.gca().invert_yaxis()
plt.show()
```



The visualization of the relationship between the variables begins below:

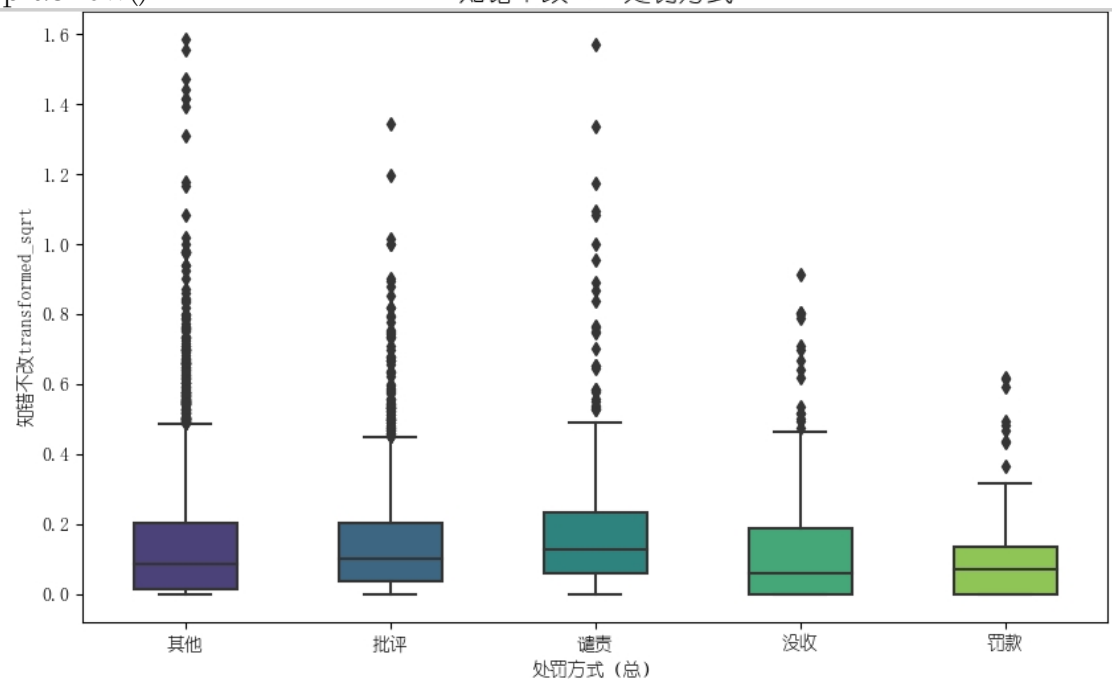
In [10].

```
#plot visualization Knowing your mistakes and dealing with units
plt.figure(figsize=(10, 6))
sns.boxplot(x='know_error_transformed_sqrt', y='processing_units', width=0.5,
palette='viridi plt.title('know_error_transformed vs processing_units', size=15))
plt.show()
```



In [11].

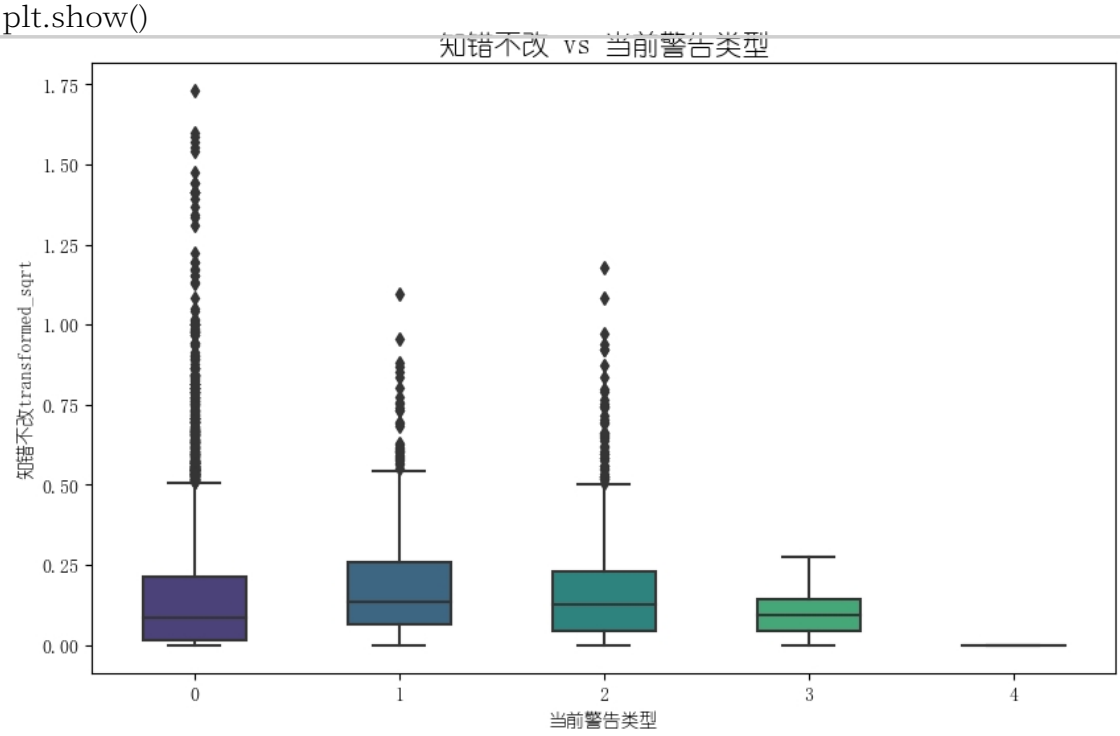
```
#plot Visualization The relationship between knowing you're wrong and the way
you're punished
plt.figure(figsize=(10, 6))
sns.boxplot(x='Punishment Methods (Total)', y='Knowing the wrong
transformed_sqrt', width=0.5, palette=' plt.title('Knowing the wrong vs
Punishment Methods', size=15)
plt.show()
```



The more severe the form of punishment, the better the company's motivation to correct itself and the lower the knowledge of wrongdoing. Consistent with expectations.

In [30].

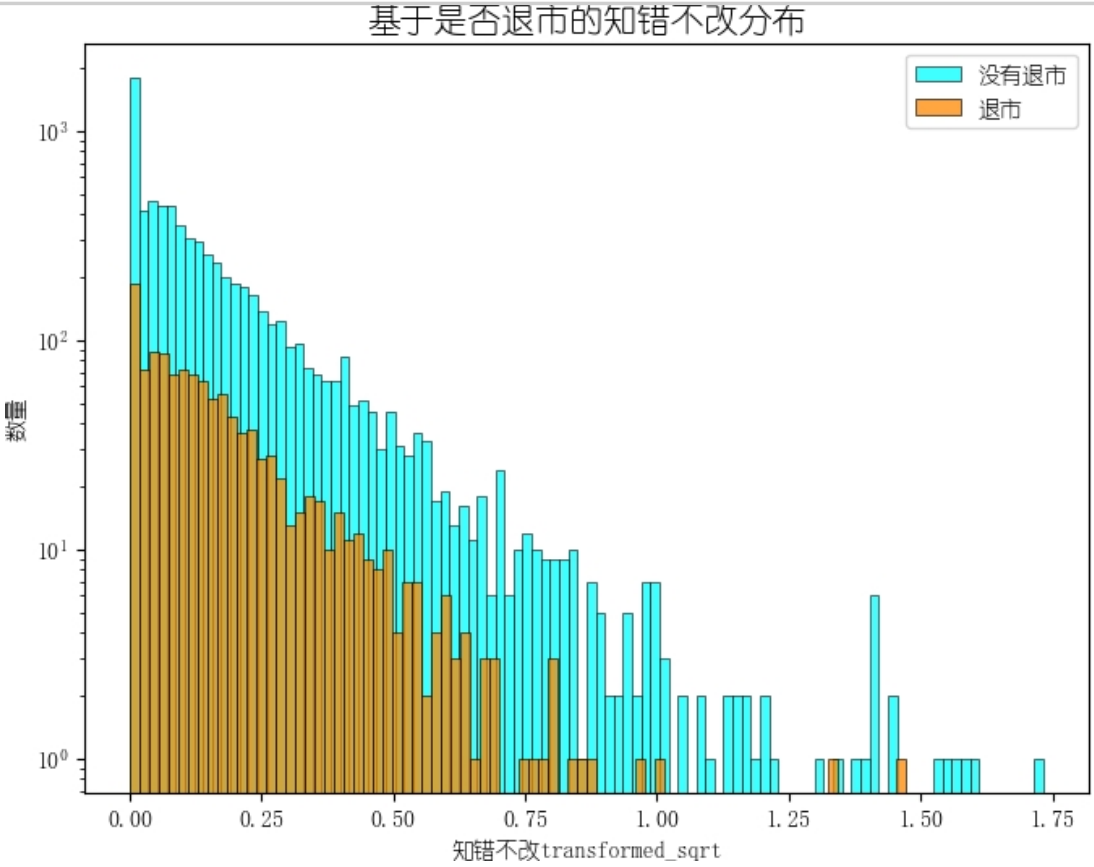
```
#plot Visualization Relationship between
known errors and current warning type #
No warning
status=0,ST=1,*ST=2,S*ST=3,PT=4
plt.figure(figsize=(10, 6))
sns.boxplot(x='current warning type', y='know_error_transformed_sqrt', width=0.5,
palette='vi plt.title('know_error_transformed vs current warning type', size=15)
plt.show()
```



The more severe the warning state is, i.e., the incentive for firms from 1-4 to consistently make the same mistakes diminishes. The side effect of warnings on corporate norms can be reflected. Consistent with expectations.

In [18].

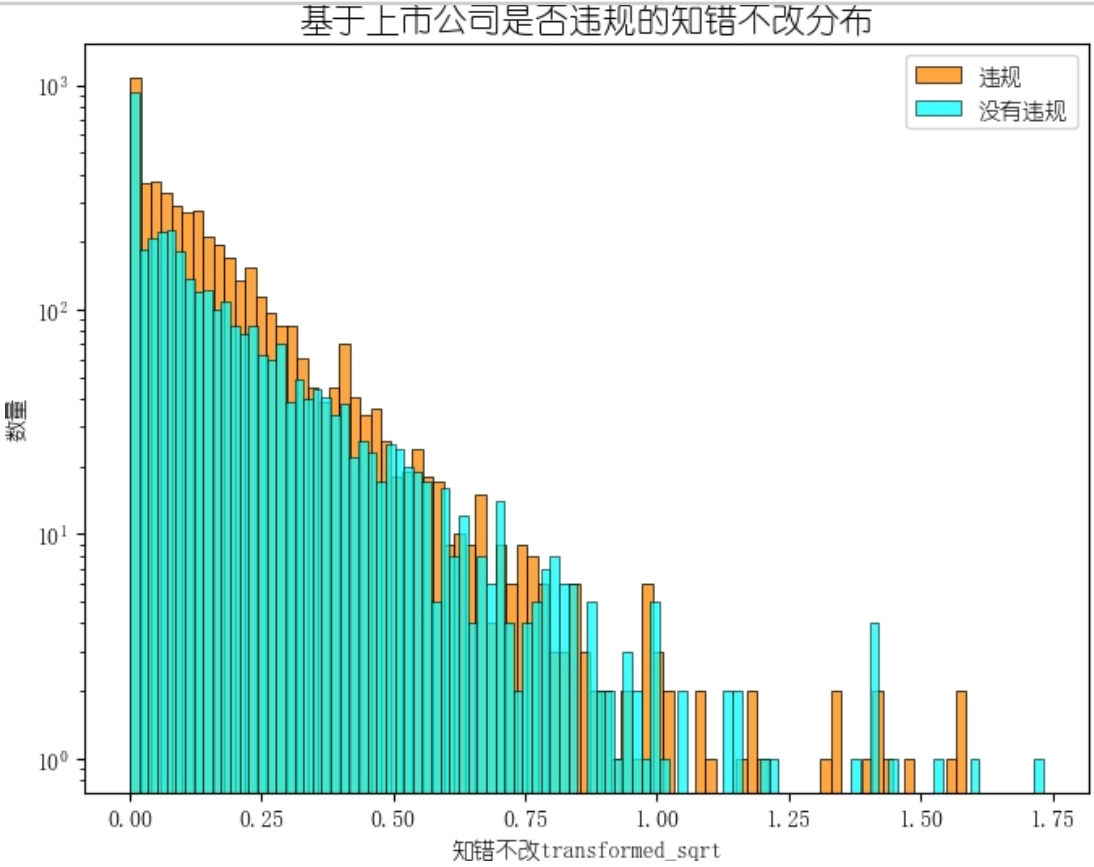
```
#plot Knowing what's wrong and whether to delist or not
recommended = df.loc[df['ultimately delisted or not'] == 1, 'aware_transformed_sqrt']
not_recommended = df.loc[df['ultimately delisted or not'] == 0,
'aware_transformed_sqrt']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', label='not
delisted',bins=100) sns.histplot(x=recommended, color='#FF8800',
label='delisted',bins=80 ) plt.title('Distribution of known errors based on
whether or not delisted', size=15)
plt.ylabel('number')
plt.yscale('log')
plt.legend()
plt.show()
```



Conditional categorization by delisting does not change the shape of the distribution of knowledge of wrongdoing; the same is true for violations below.

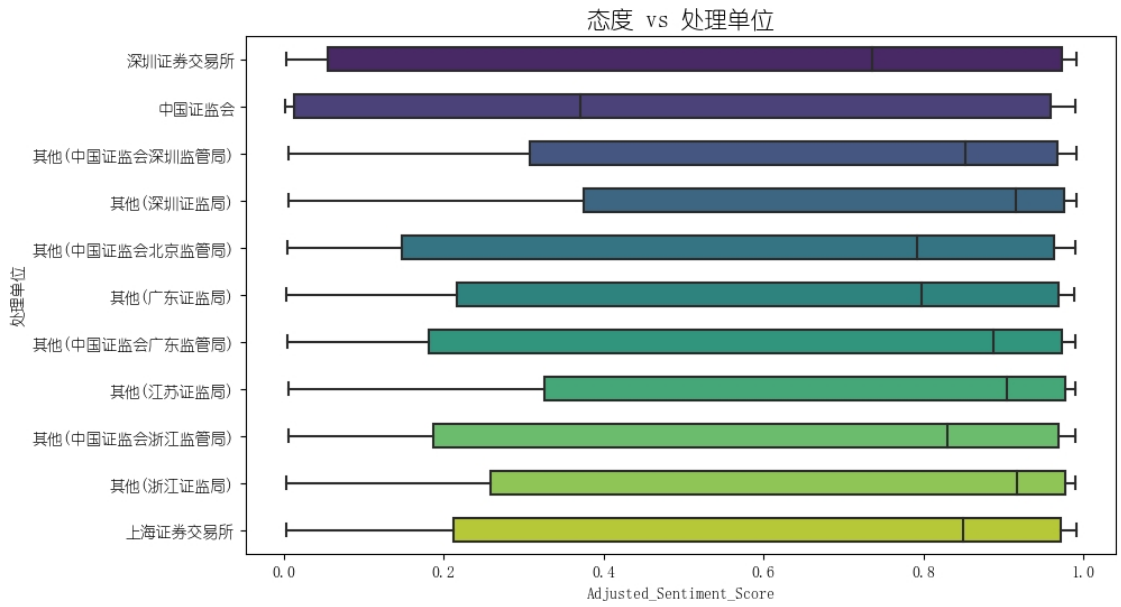
In [19].

```
#plot Knowing no wrong and whether or not public companies are in violation
recommended = df.loc[df['Is the listed company in violation'] == 1, 'Knowingly incorrect transformed_sqrt'] not_recommended = df.loc[df['Is the listed company in violation'] == 0, 'Knowingly incorrect transformed_sqrt']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#FF8800', label='violation',bins=80)
sns.histplot(x=not_recommended, color='#00FFFF', label='no violation',bins=100 )
plt.title('Distribution of known errors based on whether a listed company is in violation', size=15)
plt.ylabel('number')
plt.yscale('log')
plt.legend()
plt.show()
```



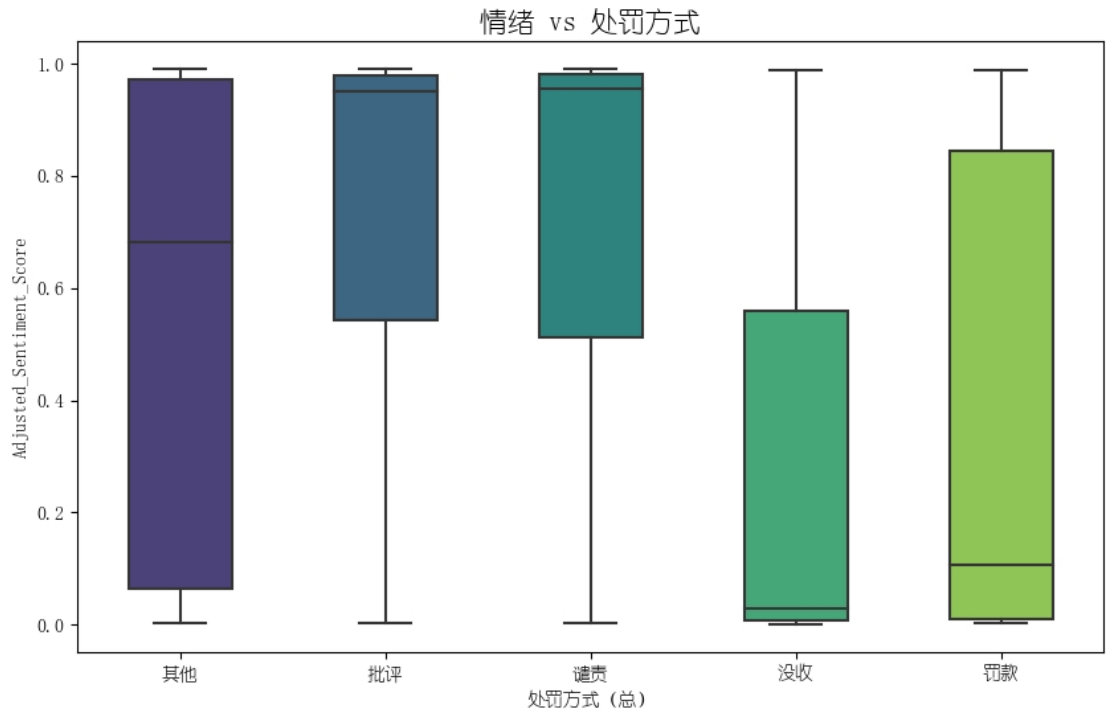
In terms of trends, violations or lack thereof do not affect the downward trend of knowledge of wrongdoing

In [22].
```
#plot Visualization of Sentiment vs. Processing Units
plt.figure(figsize=(10, 6))
sns.boxplot(x='Adjusted_Sentiment_Score', y='Treatment Units', width=0.5,
palette='viridi plt.title('Attitude vs Treatment Units', size=15)
plt.show()
```



Among all institutions, the CSRC has a smaller sentiment average and expresses a more neutral bias

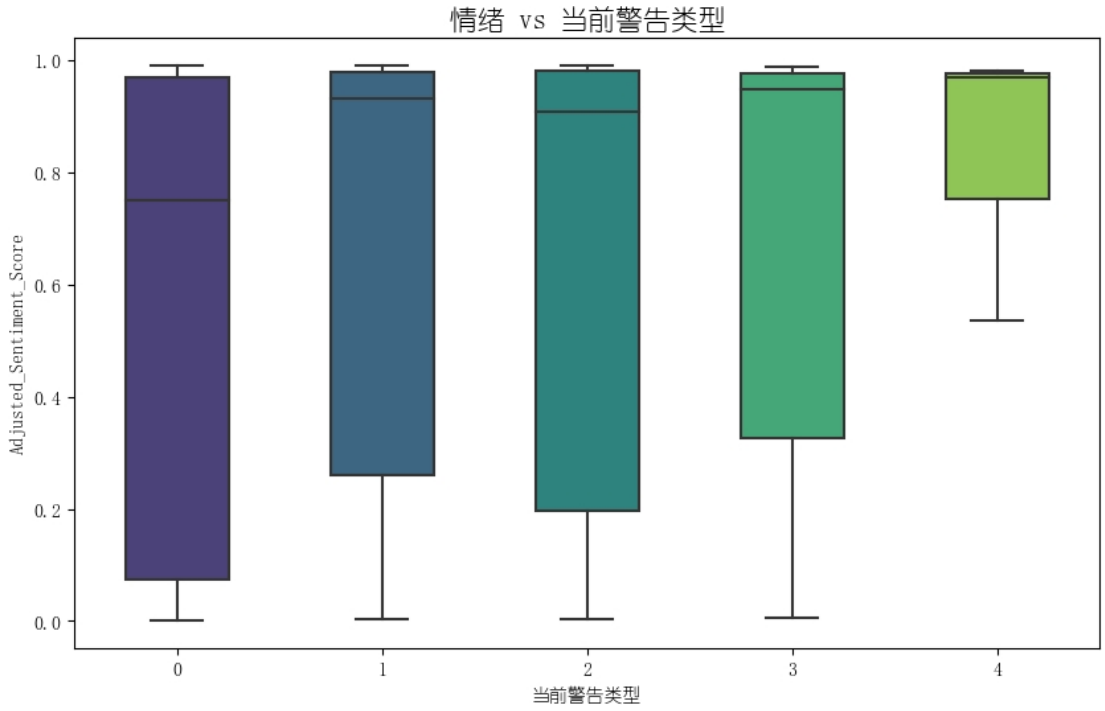
In [28].
```
#plot Visualize Sentiment vs. Penalty Approach
plt.figure(figsize=(10, 6))
sns.boxplot(x='Mode of Punishment (Total)', y='Adjusted_Sentiment_Score',
width=0.5, palette=' plt.title('Sentiment vs Mode of Punishment', size=15)
plt.show()
```



Criticism and condemnation were intuitively thought to be relatively milder than confiscation and fines. The opposite turned out to be true.

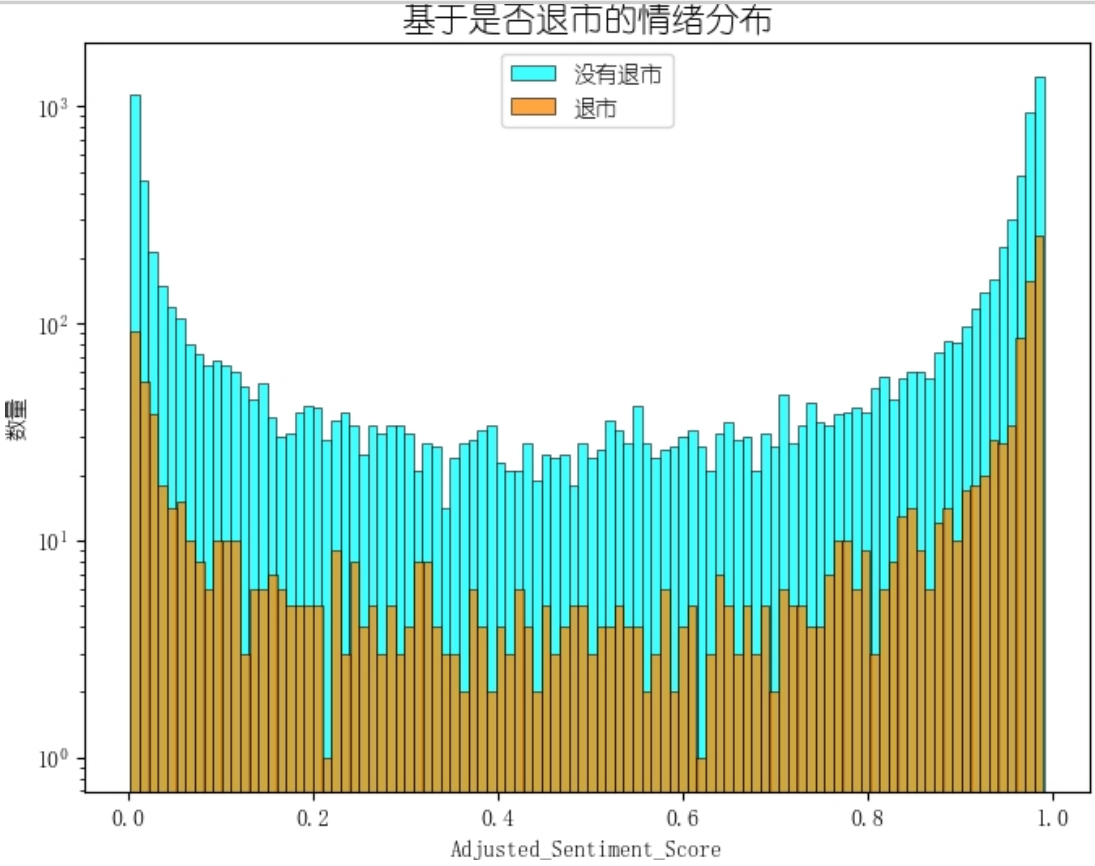
In [27].

```
#plot Visualize sentiment in relation to
current warning type # No warning state =
0, ST=1,*ST=2,S*ST=3,PT=4
plt.figure(figsize=(10, 6))
sns.boxplot(x='Current Warning Type', y='Adjusted_Sentiment_Score', width=0.5,
palette='vi plt.title('Sentiment vs Current Warning Type', size=15)
plt.show()
```



Overall, the more severe the warning status, the more negative the sentiment, as expected.

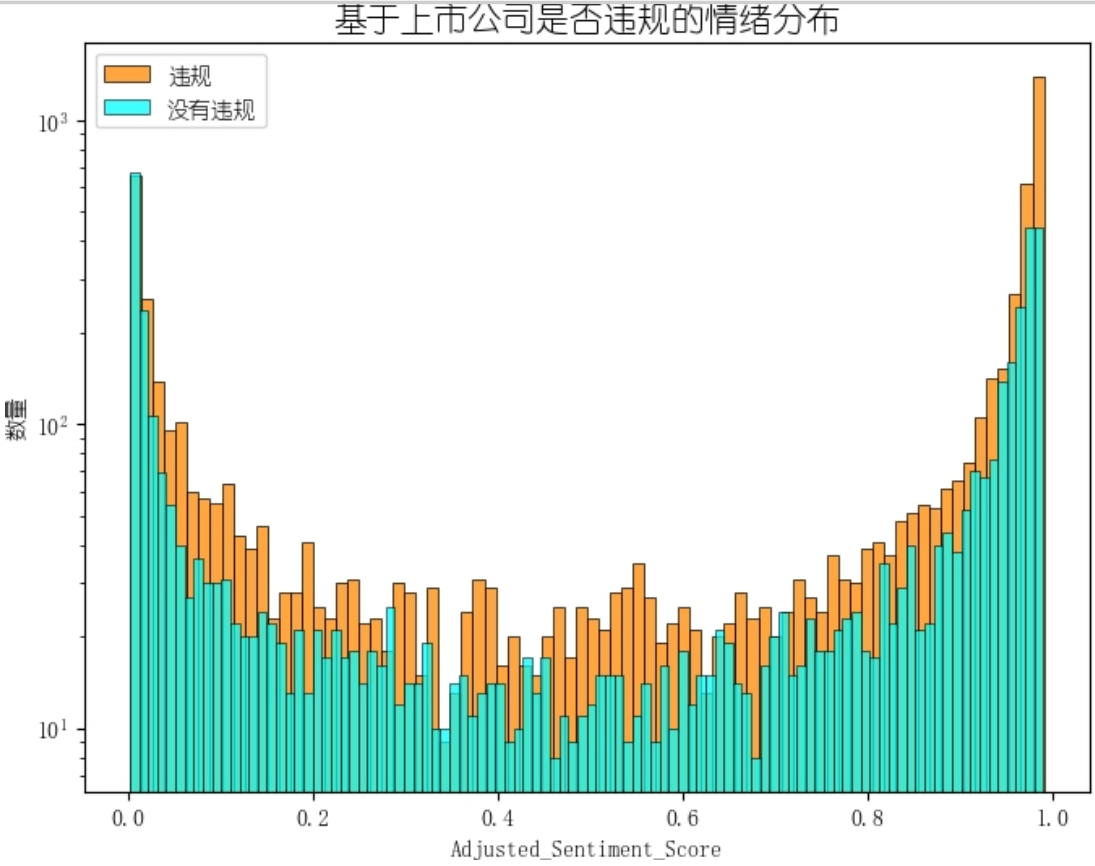
In [25]:
```
#plot sentiment and whether to delist
recommended = df.loc[df['Eventually delisted or not'] == 1,
'Adjusted_Sentiment_Score'] not_recommended = df.loc[df['Eventually delisted or
not'] == 0, 'Adjusted_Sentiment_Score']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', label='not
delisted',bins=100) sns.histplot(x=recommended, color='#FF8800',
label='delisted',bins= 100) plt.title('Sentiment distribution based on whether
to delist', size=15)
plt.ylabel('number')
plt.yscale('log')
plt.legend()
plt.show()
```



The distribution of announcement sentiment for delisted firms is not significantly different from that of non-delisted ones, regardless of time, and the same is true for the violations below:

In [25].

```
#plot sentiment and whether the public company is in violation
# No warning status = 0, ST=1,*ST=2,S*ST=3,PT=4
recommended = df.loc[df['Is the listed company in violation'] == 1,
'Adjusted_Sentiment_Score'] not_recommended = df.loc[df['Is the listed company in
violation'] == 0, 'Adjusted_Sentiment_Score ']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#FF8800', label='violation',bins=80)
sns.histplot(x=not_recommended, color='#00FFFF', label='no violation',bins=100 )
plt.title('Sentiment distribution based on whether listed companies are in
violation', size=15)
plt.ylabel('number')
plt.yscale('log')
plt.legend()
plt.show()
```

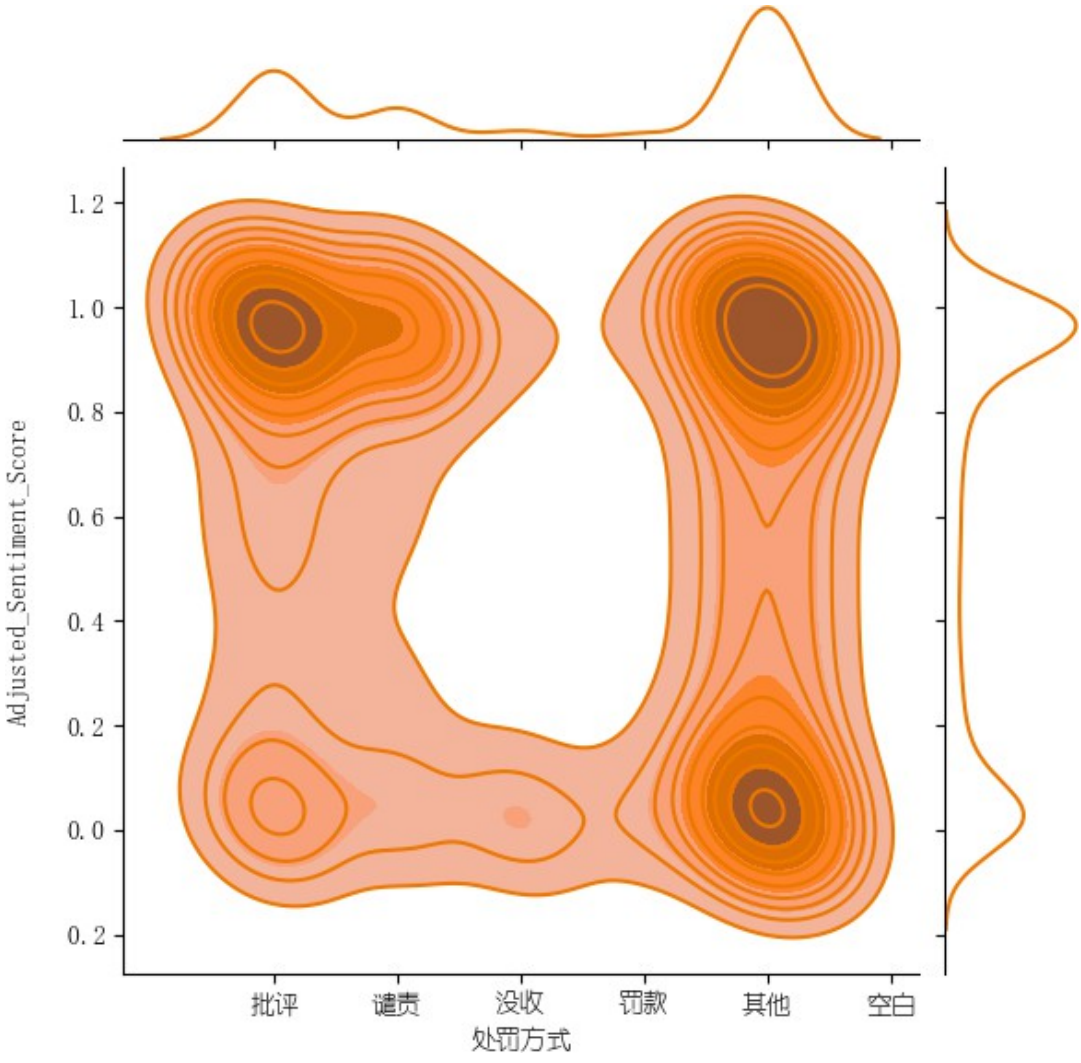


In [19].

```
#plot sentiment and the way it's penalized
df['Penalty Method'] = df['Penalty Method (Total)'].replace({'Criticize': 1.0,
'Reprimand': 2.0, 'Forfeit': 3
plt.figure(figsize=(8, 8))
g =  sns.jointplot(x='Penalty way',  y='Adjusted_Sentiment_Score',  kind='kde',
color='# g.plot_joint(sns.kdeplot, fill=True, color='#EE7700', zorder =0,
levels=6)

xtick_labels = ['criticized', 'condemned', 'confiscated', 'fine', 'other',
'blank'] g.ax_joint.set_xticks([1, 2, 3, 4, 5, 6])
g.ax_joint.set_xticklabels(xtick_labels)

plt.show()
```

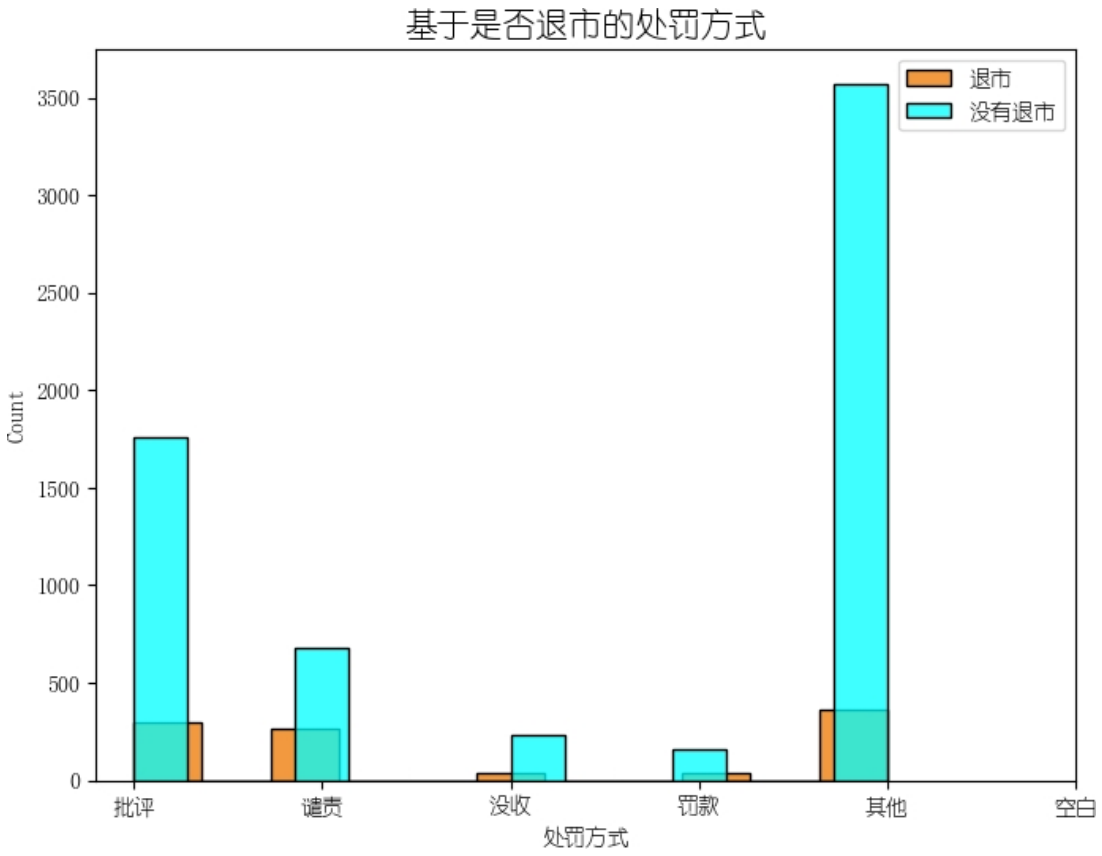
<Figure size 800x800 with 0 Axes>

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 Penalties based on whether to delist or not
recommended = df.loc[df['ultimately delisted or not'] == 1, 'penalty
method'] not_recommended =  df.loc[df['ultimately delisted or not'] == 0,
'penalty method']

plt.figure(figsize=(8, 6))
sns.histplot(x=recommended, color='#EE7700', label='delisted')
sns.histplot(x=not_recommended, color='#00FFFF', label='not delisted')

tick_positions = [1, 2, 3, 4, 5, 6]
tick_labels = ['Criticize', 'Reprimand', 'Confiscate', 'Fine', 'Other', 'Blank']

plt.xticks(tick_positions, tick_labels)

plt.title('Penalty approach based on whether to
delist', size=15) plt.legend()
plt.show()
```



Looking at the overall distributional trend, whether or not to delist did not change the distributional pattern. There are more on both sides and less in the center. The same is true for the next one, the distribution pattern has not changed.

In [30].

```
#plot Word count distribution of violations based on whether or not they were delisted
recommended = df.loc[df['ultimately delisted or not'] ==
1, 'number of words'] not_recommended =
df.loc[df['ultimately delisted or not'] == 0, 'number of
words']

plt.figure(figsize=(8, 6))
sns.histplot(x=not_recommended, color='#00FFFF', kde=True, label='not delisted',
binw sns.histplot(x=recommended, color='#EE7700', kde=True, label = 'delisted',
binwidth=8) plt.title('Distribution of violation words based on whether or not
delisted', size=15)
plt.legend()
plt.show()
```