说说整块分析内容，爬虫数据那块如果要文档说明，需要找另一个小伙伴

整个分析步骤分以下几步：

1. 数据预处理
2. 数据分析包含数据可视化，指标定义，数据对比
3. 建模，包含，数据预测，股市与博文相关性比较

先来说说数据预处理模块

数据预处理，主要是：分词，数据清洗，停用词，情感分析，中文判断

首先拿到数据，会先去掉标点符号，以及机械压缩，和去掉表情包，这样做的目前是减少噪音，确保情感数值的准确率

```
#去掉标点符号，以及机械压缩
def preprocess_word(word):
    word1 = str(word)
    word1 = re.sub(r'#\w+#', '', word1)
    word1 = re.sub(r'【.*?】', '', word1)
    word1 = re.sub(r'@[\w]+', '', word1)
    word1 = re.sub(r'[a-zA-Z]', '', word1)
    word1 = re.sub(r'\.\d+', '', word1)
    return word1
```

```
def emjio_tihuan(x):
    x1 = str(x)
    x2 = re.sub('(\[.*?\])', "", x1)
    x3 = re.sub(r'@[\w\u2E80-\u9FFF]+:?|\[\w+\]', '', x2)
    x4 = re.sub(r'\n', '', x3)
    return x4
```

接着会进行分词处理，在分词的过程中，只保留名词 形容词 动词等，并且判断该分词是否为中文，是否是两个词以上，是否在该词不在停用词库里面，减少噪音对情感数值的判断

```python
def get_cut_words(content_series):
    try:
        # 对文本进行分词和词性标注
        words = pseg.cut(content_series)
        # 保存名词和形容词的列表
        nouns_and_adjs = []
        # 逐一检查每个词语的词性，并将名词和形容词保存到列表中
        for word, flag in words:
            #判断是否为名词或者形容词或者动词
            if flag in ['Ag','a','ad','an','Ng','n','v']:
                if word not in stop_words and len(word) >= 2 and is_all_chinese(word) == True:
                    # 如果是名词或形容词，就将其保存到列表中
                    nouns_and_adjs.append(word)
        if len(nouns_and_adjs) != 0:
            return ' '.join(nouns_and_adjs)
        else:
            return np.NAN
    except:
        return np.NAN
```

在上面都处理好之后，我们接下来会进行情感判断，采用snownlp 情感库，进行情感数值和情感判断，情感分值小于等于0.4的，划分为负面，否则划分为正面

```python
def sentiment1(x):
    text = str(x)
    s = SnowNLP(text)
    sentiment = s.sentiments
    if sentiment <= 0.4:
        return "负面"
    else:
        return "正面"

def sentiment2(x):
    text = str(x)
    s = SnowNLP(text)
    sentiment = s.sentiments
    return sentiment
```

最后保留新_博文表，这个是处理好的文件

原始数据为：184704

清洗过后的数据为：180639

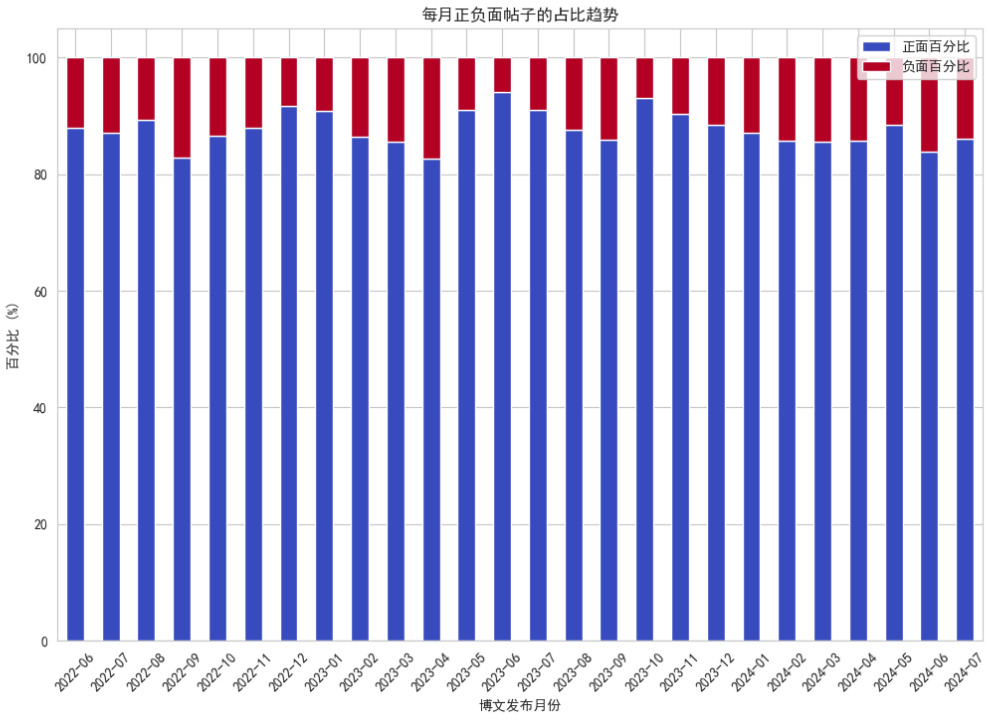
接下来，根据处理好的数据，我们去做数据展示

先做时间处理，把时间处理成通用时间，这样方便调用时间，然后把时间按照月维度进行排序，接着对时间进行聚合来查看每个月，正负面占比情况

```
def time_process1(x):
    try:
        x1 = str(x).split(" ")
        x1 = x1[0]
        if '2023年' not in x1 and '2022年' not in x1:
            x1 = '2024年' + x1
        else:
            x1 = x1
        return x1
    except:
        return np.NAN


df['博文发布时间'] = df['博文发布时间'].apply(time_process1)
df['博文发布时间'] = pd.to_datetime(df['博文发布时间'],format='%Y年%m月%d日')
df = df.sort_values(by=['博文发布时间'],ascending=True)
# 提取月份并创建一个新的列来存储月份信息
df['博文发布月份'] = df['博文发布时间'].dt.to_period('M')
```

做法如下：

```
emotion_df = df.groupby('博文发布月份').apply(emotion_type)
# 提取正面和负面数据
emotion_df = emotion_df.apply(pd.Series)  # 将字典拆分为多列
emotion_df['总数'] = emotion_df['正面'] + emotion_df['负面']  # 计算每月总数
emotion_df['正面百分比'] = (emotion_df['正面'] / emotion_df['总数']) * 100  # 计算正面百分比
emotion_df['负面百分比'] = (emotion_df['负面'] / emotion_df['总数']) * 100  # 计算负面百分比

# 绘制百分比趋势柱状图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.figure(figsize=(10, 6))
emotion_df[['正面百分比', '负面百分比']].plot(kind='bar', stacked=True, colormap='coolwarm', figsize=(12, 8))
plt.title('每月正负面帖子的占比趋势')
plt.xlabel('博文发布月份')
plt.ylabel('百分比 (%)')
plt.xticks(rotation=45)
plt.legend(loc='upper right')
plt.savefig('每月正负面帖子的占比趋势.png')
plt.show()
```



从这里可以看出，大部分情绪都是偏正向的

接着我们去查看，帖子的互动量以及情感热度的相关内容

帖子的互动量主要是根据归一化进行数据处理

```
df['博文转发数'] = df['博文转发数'].fillna(0)
df['博文评论数'] = df['博文评论数'].fillna(0)
df['博文点赞数'] = df['博文点赞数'].fillna(0)

# 对每一列进行Min-Max归一化处理
df['转发数_归一化'] = (df['博文转发数'] - df['博文转发数'].min()) / (df['博文转发数'].max() - df['博文转发数'].min())
df['评论数_归一化'] = (df['博文评论数'] - df['博文评论数'].min()) / (df['博文评论数'].max() - df['博文评论数'].min())
df['点赞数_归一化'] = (df['博文点赞数'] - df['博文点赞数'].min()) / (df['博文点赞数'].max() - df['博文点赞数'].min())

# 计算互动量
df['互动量'] = df['转发数_归一化'] + df['评论数_归一化'] + df['点赞数_归一化']
```

然后根据处理好的数值，把转发 评论 点赞这三个指标的数值相加 最后得出互动量

情感热度，就是上面采用snownlp来获取的情感数值

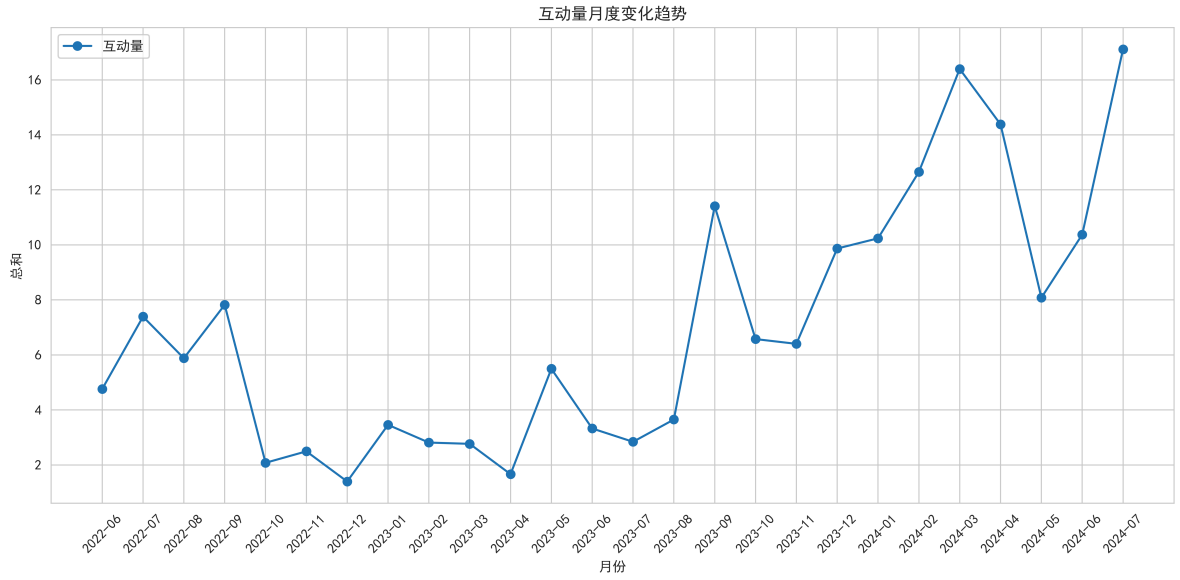
根据这上面的两个指标，我们去查看这两个指标的描述性统计，count是总计，mean是平均值，std是标准差，min-max指的是最小 最大，25% 50% 75%指的是阶段

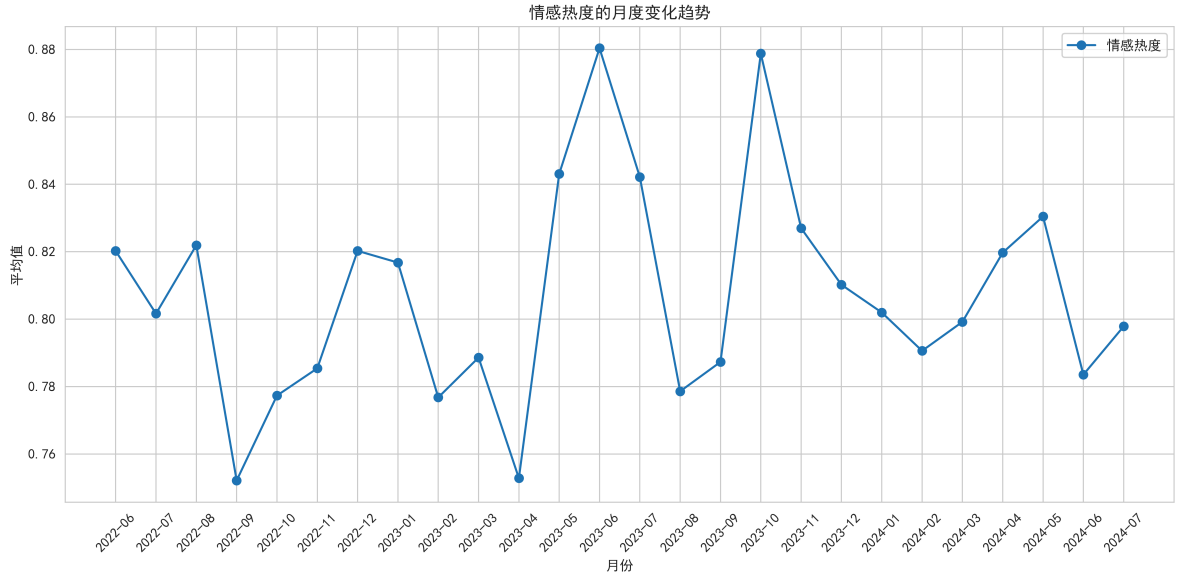
互动量统计信息:
count　180638.000000
mean　　0.001004
std　　0.014810
min　　0.000000
25%　　0.000000
50%　　0.000007
75%　　0.000145
max　　2.277020

情感热度统计信息:

count	180638.000000
mean	0.804950
std	0.291583
min	0.000000
25%	0.684409
50%	0.976153
75%	0.999927
max	1.000000

接着我们根据这两个指标，去查看他们每个月的趋势走向，互动量是统计每个月的总和，情感热度是统计每个月的均值情况，这里是需要注意的



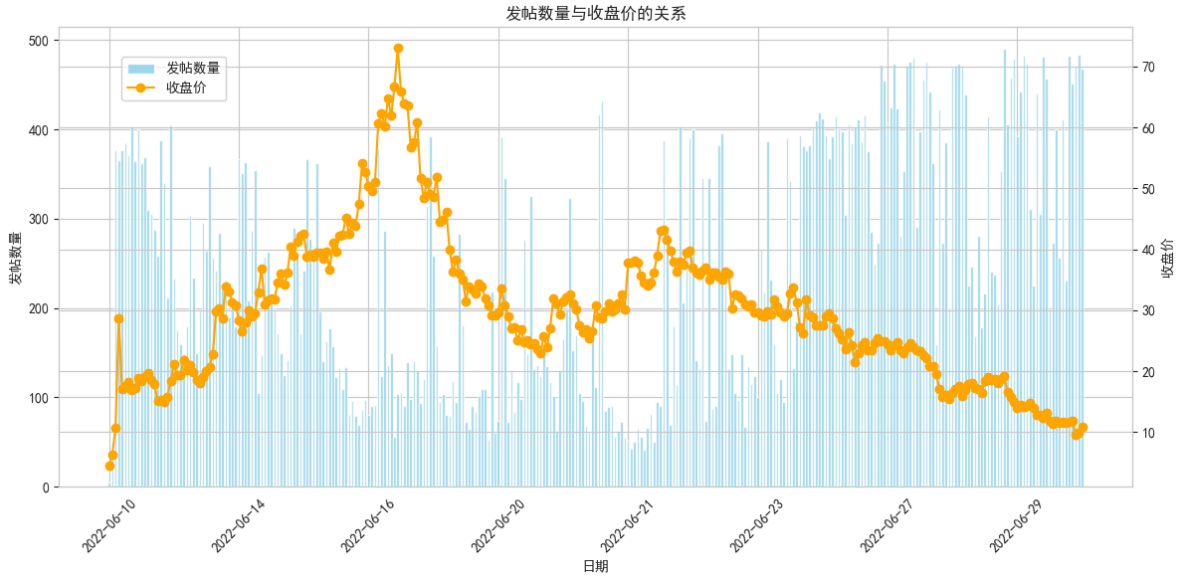


当可视化这边处理好之后，我们来查看舆情数据和股市他们之间的影响

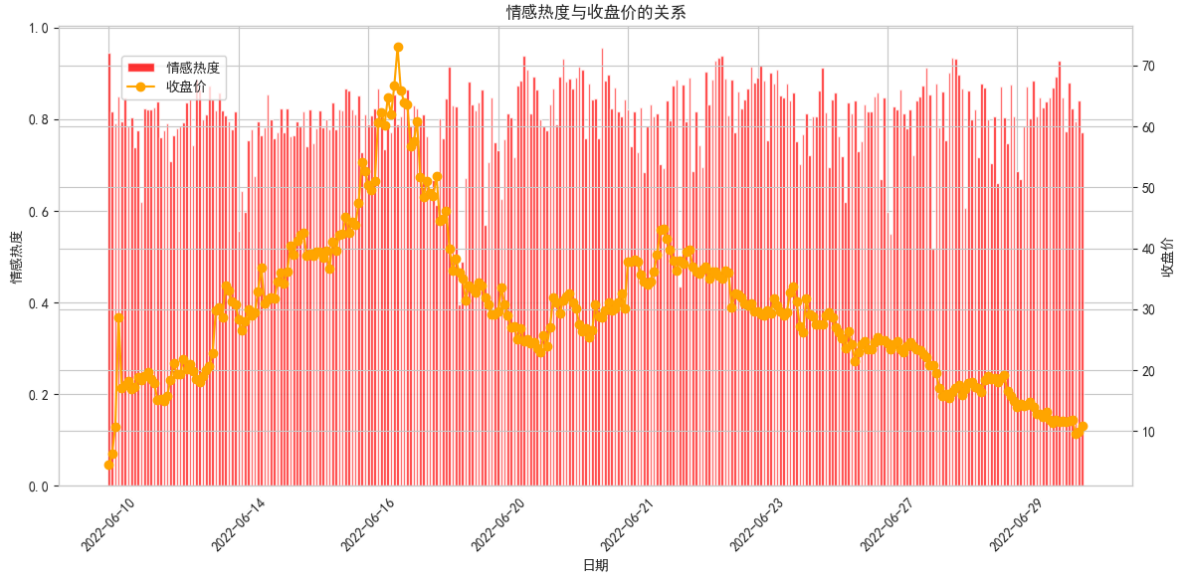
我们先查看，发帖数据和收盘价的关系，这个是实现代码

```python
def picture1():
    # 创建一个图形和轴对象
    fig, ax1 = plt.subplots(figsize=(12, 6))

    # 绘制发帖数量的柱状图
    ax1.bar(data3['日期'], data3['发帖数量'], color='skyblue', label='发帖数量', alpha=0.8)
    ax1.set_xlabel('日期')
    ax1.set_ylabel('发帖数量')
    ax1.tick_params(axis='y')
    # 设置横坐标标签的显示间隔（例如每隔7个日期显示一个标签）
    ax1.xaxis.set_major_locator(ticker.MaxNLocator(nbins=10))  # 你可以调整nbins的值来控制显示的标签数量
    ax1.set_xticklabels(data3['日期'], rotation=45)

    # 创建第二个 y 轴，共享 x 轴
    ax2 = ax1.twinx()

    # 绘制收盘价的折线图
    ax2.plot(data3['日期'], data3['收盘价'], color='orange', marker='o', label='收盘价')
    ax2.set_ylabel('收盘价')
    ax2.tick_params(axis='y')

    # 添加图例
    fig.legend(loc='upper left', bbox_to_anchor=(0.1,0.9))
    plt.rcParams['font.sans-serif'] = ['SimHei']
    # 设置图表标题
```

这个是可视化结果，从图片来看，发帖的数据和收盘价的影响不是很大



接着我们再来查看，情感数值和收盘价的相关，从这里可以看出，影响也不是很大



基于上面的两个图，我们去查看他们的相关性指标

发帖数量和收盘价之间的相关系数: -0.46915987129994313
情感得分和收盘价之间的相关系数: -0.07756562083182118

- **发帖数量和收盘价之间的相关系数：-0.469**：这个值表明发帖数量和收盘价之间存在中等程度的负相关关系。也就是说，当发帖数量增加时，收盘价倾向于下降。虽然不是非常强烈的相关性，但足够显著。
- **情感得分和收盘价之间的相关系数：-0.078**：这个值表示情感得分和收盘价之间几乎没有相关性，情感得分的变化对收盘价没有明显的直接影响。

接着我们根据发帖的数据和情感得分，和收盘价做一个**线性回归模型**评估

这里是建模的过程：

```
# 准备数据
X = data3[['发帖数量', '情感得分']]
y = data3['收盘价']

# 训练线性回归模型
model = LinearRegression()
model.fit(X, y)

# 输出回归系数
print('回归系数:', model.coef_)
print('截距:', model.intercept_)

# 计算模型评分
score = model.score(X, y)
print('模型评分:', score)

# 查看每一项对收盘价的影响
coefficients = pd.DataFrame({
    '特征': ['发帖数量', '情感得分'],
    '回归系数': model.coef_
})

print(coefficients)
```

这些是建模得出来的指标：

回归系数: [-0.04332699 -22.37674945]

截距: 58.29154626514482

模型评分: 0.24350834490398376

	特征	回归系数
0	发帖数量	-0.043327
1	情感得分	-22.376749

从回归系数来看，

- **发帖数量的回归系数：-0.043**：这意味着在保持其他变量不变的情况下，发帖数量每增加一个单位，收盘价会下降约0.043元。虽然影响的幅度较小，但方向是负的，符合相关性分析的结果。
- **情感得分的回归系数：-22.377**：情感得分对收盘价的影响显著，每增加一个单位的情感得分，收盘价会下降约22.38元。这表明情感得分对股价有较大的影响，不过由于相关系数较低，这个结果可能受到其他因素的干扰，或者样本中的某些异常值影响了回归结果。

从截距来看，

- **截距：58.292**：当发帖数量和情感得分都为零时，模型预测的收盘价为58.292元。这个截距的实际意义不大，更多是用于模型的数学计算。

从模型评分来看，
模型评分：0.244：R^2值为0.244，表明模型只能解释约24.4%的收盘价变化。这说明发帖数量和情感得分对收盘价的解释能力有限，还有大量其他因素影响着股价的变化。R^2值较低，说明模型的拟合效果不理想。

因为线性回归模型效果不是很好，这时候，我们就要考虑非线性模型，看看能否得出更好的结论和效果

我们这边采用了**随机森林的模型**

```python
# 分割数据为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 初始化随机森林回归模型
model = RandomForestRegressor(n_estimators=100, random_state=42)

# 训练模型
model.fit(X_train, y_train)

# 预测
y_pred = model.predict(X_test)

# 计算模型评分和均方误差
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f'R²: {r2}')
print(f'MSE: {mse}')

# 特征重要性
importance = model.feature_importances_
features = pd.DataFrame({
    '特征': ['发帖数量', '情感得分'],
    '重要性': importance
}).sort_values(by='重要性', ascending=False)
```

随机森林的指标如下：

R²: 0.3323761400494224
MSE: 104.33052703786866
　　特征　　重要性
0　发帖数量　0.590127
1　情感得分　0.409873

从R^2值来看，R^2值为0.332，这意味着模型能够解释大约33.2%的收盘价波动。相比之前的线性回归模型（R^2值为0.244），这个非线性模型在捕捉数据中的关系上有所改进。这表明发帖数量和情感得分对收盘价的影响确实存在一定的非线性关系。虽然R^2值有所提高，但33.2%的解释力依然较低，说明还有66.8%的收盘价波动是由其他未考虑的因素所引起的

从均方误差来看，

均方误差表示预测值与实际值之间的平均偏差。MSE值为104.33，说明在模型预测中，每个预测值与实际收盘价之间的平均误差平方值为104.33。MSE的数值越小，模型的预测精度越高，总的来说，MSE值相对较大，说明模型在某些数据点上的预测误差仍然较大

从重要性来看，

- **发帖数量的重要性（0.590）**：发帖数量对模型的贡献最大，占比约59%。这意味着在随机森林模型中，发帖数量对收盘价的预测有较大的影响力。
- **情感得分的重要性（0.410）**：情感得分的重要性为41%，表明它对收盘价的影响力也相对较大，但略低于发帖数量。

上面说完了舆情数据和股市的关系后，接下来我们来做预测

预测这边采用的是LSTM模型，

模型背景：

在预测股市数据（如股价走势）时，长短期记忆网络（LSTM）是一个非常流行的选择，特别是当处理时间序列数据时。LSTM 模型可以有效捕捉数据中的长期依赖关系和模式，因此在股市预测中表现较好

建模代码：8.股市数据-时间预测.py

相关指标：
R^2: 0.9569304765649425
MSE: 7.254488642034399

模型的表现效果不错，

`R² = 0.957`，这意味着模型能够解释约95.7%的数据变化，模型拟合度非常高，预测效果较好

`MSE = 7.25`，意味着预测值与实际值之间的差异平方和的平均值是7.25

接着我们拿实际值和预测值做一个可视化对比，这样更加直观的感受他们的差异，这里是实现代码

```python
# 结果展示
results = pd.DataFrame({'实际值': y_test, '预测值': y_pred})
results.to_excel('预测相关数据.xlsx')
# 绘制百分比趋势柱状图
plt.rcParams['font.sans-serif'] = ['SimHei']
# 绘制折线图
plt.figure(figsize=(14, 7))
plt.plot(y_test, color='blue', label='实际值')
plt.plot(y_pred, color='red', label='预测值')
plt.title('实际值 vs 预测值')
plt.xlabel('时间点')
plt.ylabel('收盘价')
plt.legend()
plt.savefig('实际值 vs 预测值.png')
plt.show()
```

这个是实际效果图，从这个效果图，我们不难看出，预测的效果还是很准的，比较接近实际数据