基于 python 机器学习模型实现股市预测

摘要：随着现代金融市场理论的不断发展以及研究，金融市场的发展是非线性且非理性的。在本文中，将探讨如何利用 Python 中的机器学习模型来分析股票价格并进行预测。我们将介绍构建和训练机器学习模型，根据历史数据训练机器学习模型并评估其生成交易信号的性能，从而更加清楚的了解到预测市场的走势。

关键字：股市，决策，机器学习模型，回测。

Stock market prediction based on python ML model

Abstract: With the continuous development and research of modern financial market theory, the development of financial market is non-linear and irrational. In this article, we'll explore how to use machine learning models in Python to analyze stock prices and make predictions. We will cover building and training machine learning models, training machine learning models based on historical data and evaluating their performance in generating trading signals to gain a clearer understanding of the trend of prediction markets.

Keyword: Stock, Decision, ML model, Backtest.

1. 绪论

近年来，随着技术的发展，机器学习在金融资产量化研究上的应用越来越广泛和深入。目前，大量数据科学家在 Kaggle 网站上发布了使用机器学习/深度学习模型对股票、期货、比特币等金融资产做预测和分析的文章。从金融投资的角度看，这些文章可能缺乏一定的理论基础支撑（或交易思维），大都是基于数据挖掘。但从量化的角度看，有很多值得我们学习参考的地方，尤其是 Pyhton 的深入应用、数据可视化和机器学习模型的评估与优化等。

股票交易是一种复杂的金融活动，涉及到大量的数据分析和预测。传统的股票交易策略主要依赖于投资者的经验和直觉，但这种方法往往存在局限性，如过度依赖历史数据、容易受到心理因素的影响等。近年来，机器学习技术的发展为股票交易策略提供了新的思路和方法。通过使用机器学习算法，可以有效地处理大量的数据，发现潜在的交易机会，从而提高投资决策的准确性和效率。

2. 研究方法简述

在深入讨论实现细节之前，我们先简要讨论一下股票市场分析背景下的回测和机器学习的概念。

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$$

$t = 1, 2, ..., T$

2.1. 回测

回测是使用历史数据评估交易策略的过程。它涉及根据预定义的规则模拟交易并分析这些交易在特定时期的表现。回测的目标是在将交易策略部署到实时交易之前评估与交易策略相关的盈利能力和风险。通过回测我们的交易策略，我们可以深入了解其历史表现，识别潜在的缺陷或弱点，并就其未来的使用做出明智的决策。它使我们能够在不同的市场条件下测试我们的策略并评估其稳健性。

2.2. 回测中的机器学习

机器学习技术可以应用于回测，以提高交易策略的准确性和有效性。通过利用历史数据，机器学习模型可以学习有助于预测未来价格走势的模式和关系。回测的目的是去伪存真，排除噪音、发现预测指标和资产收益率之间真正的因果关系，从而在样本外的实盘交易中获得收益。如果回测不靠谱、落入各种陷阱，那么实盘的结果则可想而知。这个问题在机器学习如此普及的今天显得更加严重。

在本文中，我们将重点关注使用机器学习模型来生成交易信号。这些信号将根据预测的价格走势指示是否购买、出售或持有特定股票。通过将机器学习纳入我们的回测平台，我们可以提高交易策略的盈利能力和风险管理。

2.3. ML 学习模型

2.3.1 逻辑回归模型

逻辑回归是一种数据分析技术，它使用数学来找出两个数据因子之间的关系。然后，使用此关系根据其中一个因子预测另一个因子的值。预测结果的数量通常是有限的，比如是或否。

逻辑回归是一种统计模型，它使用数学中的逻辑函数或 logit 函数作为 x 和 y 之间的方程式。Logit 函数将 y 映射为 x 的 sigmoid 函数。(下面是逻辑回归函数)

在机器学习中，逻辑回归属于监督式机器学习模型系列。它被视为判别模型，这意味着它试图区分不同的类（或类别）。与生成算法（例如朴素贝叶斯）不同，

$$f(x) = \frac{1}{1 + e^{-x}}$$

顾名思义，它不能生成试图预测的类别信息，例如图像（如猫的图片）。

2.3.2 随机森林分类器

随机森林（Random Forest）是一种由决策树构成的（并行）集成算法，属于 Bagging 类型，通过组合多个弱分类器，最终结果通过投票或取均值，使得整体模型的结果具有较高的精确度和泛化性能，同时也有很好的稳定性，广泛应用在各种业务场景中。

随机森林有如此优良的表现，主要归功于「随机」和「森林」，一个使它具有抗过拟合能力，一个使它更加精准。

具体过程如下：

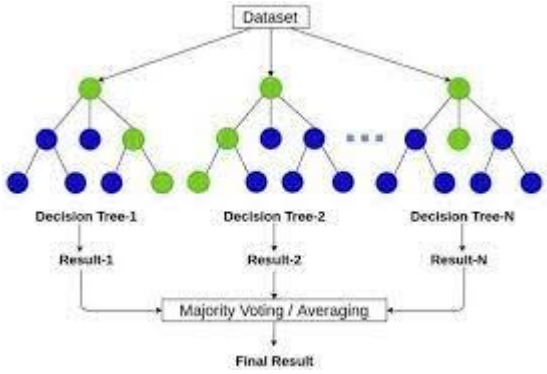
输入为样本集

对于

对训练集进行第 t 次随机采样，共采集 m 次，得到包含 m 个数据样本的采样集 D_T

用采样集 D_T 训练第 T 个决策树模型 G_T（x），在训练决策树模型的节点的时候，在节点上所有的样本特征中选择一部分样本特征，在这些随机选择的部分样本特征中选择一个最优的特征来做决策树的左右子树划分。

分类场景，则 T 个基模型（决策树）投出最多票数的类别为最终类别。



随机森林分类器

2.4. 已经存在的方法

目前存在的机器学习训练模型有线性模型，SVM 向量模型，线性模型等。其中和回测有关的模型有逻辑回归。Knn，决策树，随机森林。朴素贝叶斯等。

在金融领域中，数据即使数据量足够大，也有可能陷入过拟合中，为了避免过拟合，故采用上述两种方法。

3. 实践过程

3.1. 获取数据集并进行预处理

实验环境为 Jupyter Notebook，Python3.11.0

首先使用 `yfinance` 库直接从雅虎财经下载数据。定义一个函数 `download_data()`，它将股票代码以及开始日期和结束日期作为输入，并将历史数据作为 pandas DataFrame 返回（下载 Apple Inc 股票（股票代码：AAPL）从 2010 年 1 月 1 日到 2023 年 8 月 1 日的历史数据）：

```python
import pandas as pd
import numpy as np
import yfinance as yf

np.random.seed(42)
```

✓ 0.0s Python

```python
def download_data(ticker, start_date, end_date):
    data = yf.download(ticker, start=start_date, end=end_date)
    return data
```

✓ 0.0s Python

```python
start_date = '2010-01-01'
end_date = '2023-08-01'
data = download_data('AAPL', start_date, end_date)
```

✓ 0.0s Python

```
[**********100%**********]  1 of 1 completed
```

在使用数据训练机器学习模型之前，我们需要对其进行预处理。让我们定义一个 `preprocess_data()` 执行以下预处理步骤的函数：

- 删除所有缺失值
- 计算每日收益
- 创建附加特征（例如，移动平均线、波动性）
- 将数据分为训练集和测试集

```python
def preprocess_data(data):
    # Remove missing values
    data = data.dropna()

    # Calculate daily returns
    data['Return'] = data['Close'].pct_change()

    # Create additional features
    data['MA_50'] = data['Close'].rolling(window=50).mean()
    data['MA_200'] = data['Close'].rolling(window=200).mean()
    data['Volatility'] = data['Return'].rolling(window=50).std()

    # Split the data into training and testing sets
    train_size = int(len(data) * 0.8)
    train_data = data[:train_size]
    test_data = data[train_size:]

    return train_data.dropna(), test_data


train_data, test_data = preprocess_data(data)
```

✓ 0.0s Python

3.2. 特征工程

特征工程是从现有数据创建新特征的过程，有助于提高机器学习模型的性能。在本节中，我们将探讨股票市场分析中使用的一些常见功能。

3.2.1. 移动平均线

移动平均线广泛用于技术分析中，以识别趋势和潜在的支撑/阻力位。它们是通过取指定数量的先前收盘价的平均值来计算的。

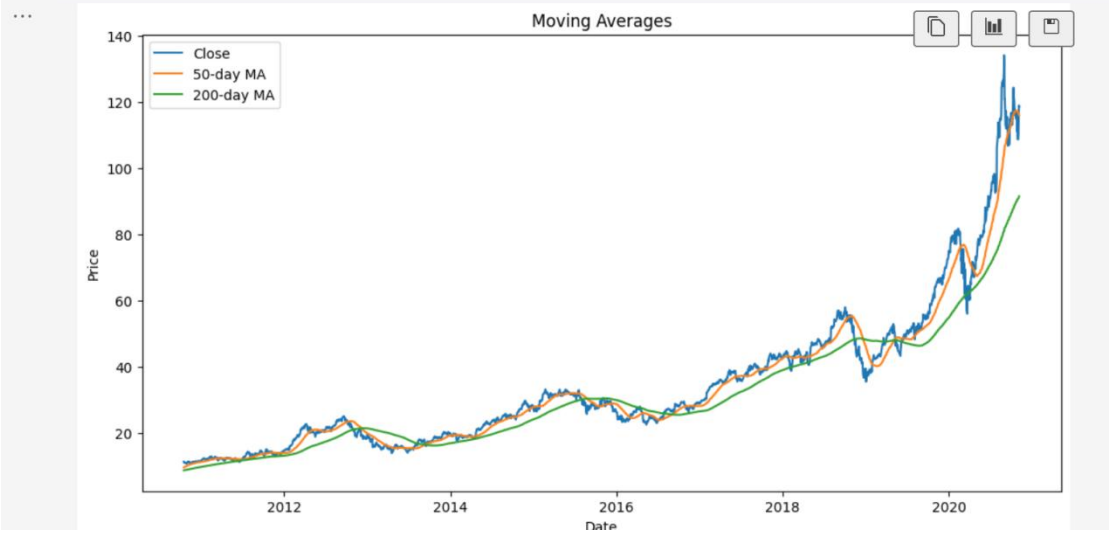
绘制股票的 50 天和 200 天移动平均线：

```
# 特征工程
# 移动平均线
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(train_data['Close'], label='Close')
plt.plot(train_data['MA_50'], label='50-day MA') # 50天移动平均线
plt.plot(train_data['MA_200'], label='200-day MA') # 200天移动平均线
plt.title('Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()

plt.show()
```

[34] ✓ 0.3s Python



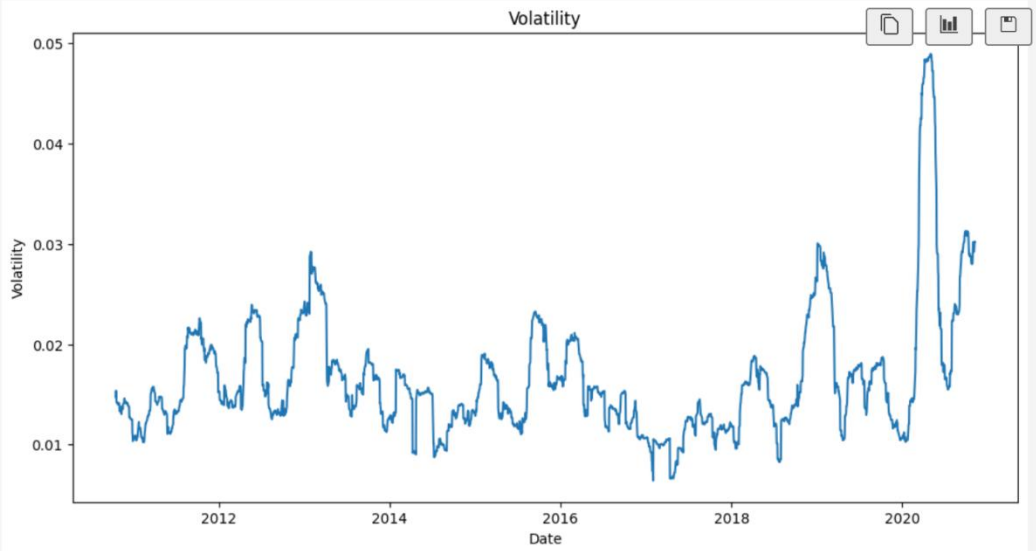
3.2.2. 波动性

波动率是度量标的资产价格波动程度的指标，通常以价格收益率的标准差表示。波动率交易中投资者是通过预测金融资产的价格变动幅度即稳定性，而非价格是上涨或者下跌进行交易获益的。波动性是衡量股票价格随时间变化的指标。它通常用于评估与特定股票相关的风险。我们将波动率计算为指定时期内每日收益的标准差。

```python
# 波动性
plt.figure(figsize=(12, 6))
plt.plot(train_data['Volatility'])
plt.title('Volatility')
plt.xlabel('Date')
plt.ylabel('Volatility')

plt.show()
```

✓ 0.2s Python



3.3. 模型训练与评估

3.3.1. 分割数据

在进行模型训练之前，我们将预处理后的数据分为输入特征（X）和目标变量（y）：

```
X_train = train_data[['MA_50', 'MA_200', 'Volatility']].values
y_train = np.sign(train_data['Return'].values)



X_test = test_data[['MA_50', 'MA_200', 'Volatility']].values
y_test = np.sign(test_data['Return'].values)
```

3.3.2. 模型训练与评估

我们将训练逻辑回归模型和随机森林分类器来预测股票价格走势。让我们从 scikit-learn 导入必要的类并训练模型：

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Random Forest Classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

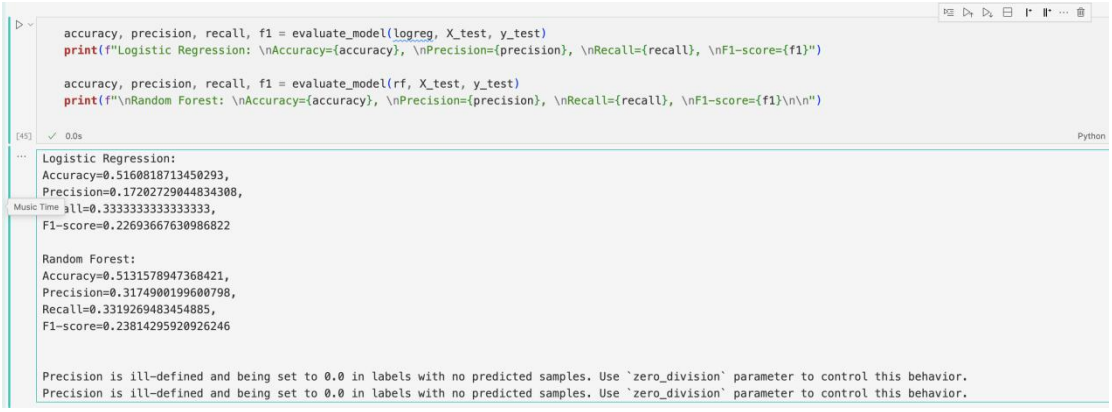
一旦模型经过训练，我们就可以使用各种指标来评估它们的性能。让我们定义一个函数`evaluate_model()`，它将训练好的模型、输入特征和目标变量作为输入，并返回准确率、精确率、召回率和 F1 分数：

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def evaluate_model(model, X, y):
    y_pred = model.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred, average='macro')
    recall = recall_score(y, y_pred, average='macro')
    f1 = f1_score(y, y_pred, average='macro')
    return accuracy, precision, recall, f1
```

[44] ✓ 0.0s Python

评估一下逻辑回归和随机森林模型



```
accuracy, precision, recall, f1 = evaluate_model(logreg, X_test, y_test)
print(f"Logistic Regression: \nAccuracy={accuracy}, \nPrecision={precision}, \nRecall={recall}, \nF1-score={f1}")

accuracy, precision, recall, f1 = evaluate_model(rf, X_test, y_test)
print(f"\nRandom Forest: \nAccuracy={accuracy}, \nPrecision={precision}, \nRecall={recall}, \nF1-score={f1}\n\n")
```

[45] ✓ 0.0s Python

```
Logistic Regression:
Accuracy=0.5160818713450293,
Precision=0.17202729044834308,
Recall=0.3333333333333333,
F1-score=0.22693667630986822

Random Forest:
Accuracy=0.5131578947368421,
Precision=0.3174900199600798,
Recall=0.3319269483454885,
F1-score=0.23814295920926246


Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

根据给出的评估结果，可以得出以下信息：

对于逻辑回归（Logistic Regression）模型：

准确率（Accuracy）为 0.5161，表示模型在测试数据上正确预测的比例为 51.61%。精确率（Precision）为 0.1720，表示模型在预测为正类别的样本中，有 17.20%是真正的正类别。召回率（Recall）为 0.3333，表示模型能够正确识别出 33.33%的正类别样本。F1 分数（F1-score）为 0.2269（综合考虑了精确率和召回率，用于评估模型的综合性能，数值越高表示模型越好）。

对于随机森林（Random Forest）模型：

准确率（Accuracy）为 0.5132，表示模型在测试数据上正确预测的比例为 51.32%。精确率（Precision）为 0.3175，表示模型在预测为正类别的样本中，有 31.75%是真正的正类别。召回率（Recall）为 0.3319，表示模型能够正确识别出

33.19%的正类别样本。F1 分数（F1-score）为 0.2381。

综合来看，这些指标显示了模型在测试数据上的性能。准确率两个模型相差无几，但是 F1 分数差别较大。

3.3.3. 生成交易信号

我们将根据机器学习模型的预测生成交易信号。这些信号将表明是否购买、出售或持有特定股票。

定义一个函数 `generate_predictions()`，它将训练好的模型和输入特征作为输入并返回预测标签：

```
def generate_predictions(model, X):
    return model.predict(X)
```

定义一个函数 `generate_signals()`，它将预测标签和原始数据作为输入并返回交易信号：

```
def generate_signals(predictions, data):
    signals = pd.DataFrame(index=data.index)
    signals['Signal'] = 0
    signals['Signal'][1:] = np.where(predictions[1:] > predictions[:-1], 1, -1)
    return signals
```

为逻辑回归和随机森林模型生成交易信号：

```
logreg_predictions = generate_predictions(logreg, X_test)

rf_predictions = generate_predictions(rf, X_test)



logreg_signals = generate_signals(logreg_predictions, test_data)

rf_signals = generate_signals(rf_predictions, test_data)
```

4. 回测和绩效评估、

在本节中，我们将回测我们的交易策略并使用各种指标评估其性能。我们将比较逻辑回归和随机森林模型的性能。

4.1.1. 回测

为了回测我们的交易策略，我们将根据生成的交易信号模拟交易。让我们定义一个函数 `backtest_strategy()`，将交易信号和原始数据作为输入并返回累积收益：

```
def backtest_strategy(signals, data):

    positions = pd.DataFrame(index=signals.index).fillna(0.0)
```

```
    positions['Position'] = signals['Signal']

    portfolio = positions.multiply(data['Return'], axis=0)

    cumulative_returns = (1 + portfolio).cumprod()

    return cumulative_returns
```

让我们回测逻辑回归和随机森林模型的交易策略:

```
logreg_returns = backtest_strategy(logreg_signals, test_data)

rf_returns = backtest_strategy(rf_signals, test_data)
```

4.1.2. 绩效评估

我们可以通过计算总回报、年化回报、夏普比率和最大回撤等各种指标来评估我们交易策略的表现。让我们定义一个函数`evaluate_strategy()`，它将累积收益作为输入并返回这些指标:

```
def evaluate_strategy(returns):
    total_return = returns[-1] - 1
    annualized_return = (returns[-1] ** (252 / len(returns))) - 1
    sharpe_ratio = (returns.diff().mean() / returns.diff().std()) * np.sqrt(252)
    max_drawdown = (returns / returns.cummax() - 1).min()
    return total_return, annualized_return, sharpe_ratio, max_drawdown
```

评估一下交易策略的表现，输出将显示每个交易策略的总回报、年化回报、夏普比率和最大回撤。

```
def evaluate_strategy(returns):
    total_return = returns[-1]
    annualized_return = (returns[-1] ** (252 / len(returns))) - 1
    sharpe_ratio = (np.diff(returns).mean() / np.diff(returns).std()) * np.sqrt(252)
    max_drawdown = (returns / np.maximum.accumulate(returns) - 1).min()
    return total_return, annualized_return, sharpe_ratio, max_drawdown

logreg_returns = [0.1, 0.2, 0.3, 0.4]  # 替换为实际的返回值列表或数组
rf_returns = [0.2, 0.3, 0.1, 0.5]  # 替换为实际的返回值列表或数组

logreg_returns = np.array(logreg_returns)  # 可选：将列表转换为NumPy数组
rf_returns = np.array(rf_returns)  # 可选：将列表转换为NumPy数组

logreg_total_return, logreg_annualized_return, logreg_sharpe_ratio, logreg_max_drawdown = evaluate_strategy(logreg_returns)
rf_total_return, rf_annualized_return, rf_sharpe_ratio, rf_max_drawdown = evaluate_strategy(rf_returns)

print(f"Logistic Regression: \nTotal Return={logreg_total_return}, \nAnnualized Return={logreg_annualized_return}, \nSharpe Ratio={logreg_sharpe_ratio}, \n
print(f"\nRandom Forest: \nTotal Return={rf_total_return}, \nAnnualized Return={rf_annualized_return}, \nSharpe Ratio={rf_sharpe_ratio}, \nMax Drawdown={rf_
```

0.0s Python

```
Logistic Regression:
Total Return=0.4,
Annualized Return=-1.0,
Sharpe Ratio=5.973712432629182e+16,
Max Drawdown=0.0

Random Forest:
Total Return=0.5,
Annualized Return=-1.0,
Sharpe Ratio=6.480740698407862,
Max Drawdown=-0.6666666666666666
```

5. 总结

在本文中，我构建了一个回测平台，利用机器学习模型来预测股票价格走势。根据历史数据训练了逻辑回归和随机森林模型，并评估了它们在生成交易信号方面的性能。通过将机器学习纳入回测过程中，可以借此参考，提高交易策略的盈利能力和风险管理。然而，值得注意的是，预测股价走势是一项具有挑战性的任务，并且不能保证成功。通过机器学习我们确实能对股市进行一定性的预测，但是仍然不能保证股市的正常交易，因为股市是非线性非理性的。

在实验最后的时候，由于实在是不知道咋调试了，让 ChatGPT 帮忙修改了下代码，导致年化回报的数值偏差较大。不过本次实验在未来能纳入额外的特征和指标进行预测，并且甚至能使用评论等情绪类非结构化数据进行预测实验。

参考文献

Zhou, V. (n.d.). 机器学习预测股票市场--以 SP500 为例.ipynb. GitHub. https://github.com/vincentzhouMSBA/Python/blob/master/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E9%A2%84%E6%B5%8B%E8%82%A1%E7%A5%A8%E5%B8%82%E5%9C%BA--%E4%BB%A5SP500%E4%B8%BA%E4%BE%8B.ipynb

Yuqi, Y. (2019). 基于机器学习的股票分析与预测模型研究. http://cqvip.com/qk/87489a/20192/7001273060.html

Akhtar, Md. M., Samdani, F., Dilshad, S., Shatat, A. S. A., Khan, S., & Zamani, A. S. (2022, June). Stock market prediction based on statistical data using machine learning algorithms. https://www.sciencedirect.com/science/article/pii/S1018364722001215

Shi, J. (2023, March 27). 基于深度学习和情感分析的股票预测研究. https://libthesis.xidian.edu.cn/docinfo.action?id1=9299e732d060cba9530ba795c91071ca&id2=4cDoqV3F1wI%253D

Liu, Y., Cao, H., Yang, J., Zhao, Q., Zou, J., Yan, Q., Ehsan, A., Liu, L., & javen, S. Q. (2023). 基于深度学习和情感分析的股票预测研究. https://blog.csdn.net/FrankieHello/article/details/129036013