大数据商业分析期末项目

题目：LSTM 模型预测股票涨跌幅

——数据驱动的商业决策分析

组　　长　邓栩瀛 21307035

组　　员　肖瑶 22323103

徐子雅 22306066

2023 年 11 月 13 日

目录

1.商业问题
2.数据准备
    数据来源
    数据介绍
        A股日线行情
        每日指标
    数据质量
    数据统计特征
    数据预处理
    数据可视化
        股票收盘价价格走势
        股票成交量变化图
        股票涨跌幅日变化
        K线图绘制
    数据分析报告
        涨跌幅pct_chg
3.建模
    建模算法
        LSTM算法原理
        为什么选择使用LSTM模型
    建模步骤与过程
        导入相关依赖库
        数据集的准备及划分
        构建LSTM模型
            激活函数的选择
            损失函数的选择
            模型构建
            优化器的选择
            模型的编译
            模型训练
        模型验证
    模型评估
4.知识发现与商业决策分析
    股票市场分析
    投资决策
        趋势交易策略
        振荡交易策略
        时间序列异常检测
        股票预测复杂性
        有效市场假说
        复杂性因素
5.感想
    案例分析感想
        数据分析的效率
        预测能力
        投资组合优化
        交易策略
        技术限制
        人类专家的角色
    课程感想
        理论与实践结合
        数据分析的实用性
小组分工

1.商业问题

背景：在股票市场中，准确预测股票的涨跌幅对投资者和机构具有重要意义，这对他们的投资决策和资金管理至关重要，但是股票市场的波动性和不确定性使得准确预测股票价格变得十分困难，因此，涨跌幅成为一种更加可靠和有意义的指标。

目标：利用LSTM模型分析股票的涨跌幅，为投资者提供决策依据。通过对历史股票数据的学习和分析，LSTM模型可以捕捉到股票价格的趋势和模式，并作出相对准确的涨跌幅预测。通过这样的预测信息可以帮助投资者制定更明智的投资策略，降低风险并提高回报。

涨跌幅作为一种相对变动的度量，更加直观地反映了股票价格的波动情况。相比之下，股价本身可能受到多种因素的影响，如市场整体情况、公司业绩、行业竞争等等。因此，仅依靠股价进行分析和预测往往不够全面和准确。与股价相比，涨跌幅更能提供有用的信息，例如，涨跌幅可以告诉投资者一个股票价格相对于前一天或前一段时间的变化情况。通过分析和预测涨跌幅，投资者可以更好地理解股票的价格趋势和波动性，并根据这些信息做出相应的投资决策。此外，涨跌幅还可以用于计算和比较不同股票之间的相对表现，帮助投资者评估投资组合的风险和回报。

2.数据准备

数据来源

https://tushare.pro

Tushare是一个免费的、开源的Python财经数据接口包，主要用于提供股票及金融市场相关的数据，是国内常用的金融数据分析工具之一。Tushare对于投资研究、教学、项目背景调研等多种需求提供了极大的方便。

Tushare的主要特点：

- 数据丰富：Tushare提供了包括实时行情数据、历史行情数据、基本面数据、宏观经济数据、公司基本信息、大盘指数数据、行业数据、新闻和公告等多种数据。
- 接口简单：使用Python调用Tushare接口相当简单。即便是初学者，只需数行代码，就能获取所需的金融数据。
- 社区活跃：由于Tushare的免费和开源特性，其社区相当活跃，有大量的开发者和爱好者进行维护和更新。
- 扩展性强：除了官方提供的数据接口，Tushare也支持自定义数据扩展，可以通过插件方式实现数据的快速扩展。

如需在python中使用Tushare，执行

```
pip install tushare
import tushare as ts
pro = ts.pro_api('your-key')
```

获取your-key

注册网站：https://tushare.pro/register

注册后进入个人主页，找到接口TOKEN，复制后替代上述命令中的`your-key`

数据介绍

本实验数据选取时间范围为2000-04-04~2023-10-31，股票代码为000001.SZ

A股日线行情

名称	类型	描述
ts_code	str	股票代码
trade_date	str	交易日期
open	float	开盘价
high	float	最高价
low	float	最低价
close	float	收盘价
pre_close	float	昨收价(前复权)
change	float	涨跌额
pct_chg	float	涨跌幅 （未复权）
vol	float	成交量 （手）
amount	float	成交额 （千元）

数据调用：

```
df = pro.daily(ts_code='000001.SZ', start_date='20000404', end_date='20231031')
```

trade_date	ts_code	trade_date	open	high	low	close	pre_close	change	pct_chg	vol	amount
2000-04-04	000001.SZ	20000404	18.28	18.34	18.05	18.11	18.28	-0.17	-0.9300	63563.00	1.154267e+05
2000-04-05	000001.SZ	20000405	18.10	18.67	18.05	18.35	18.11	0.24	1.3300	58979.00	1.078698e+05
2000-04-06	000001.SZ	20000406	18.40	18.50	18.25	18.48	18.35	0.13	0.7100	52445.00	9.625639e+04
2000-04-07	000001.SZ	20000407	18.58	19.20	18.41	18.91	18.48	0.43	2.3300	221067.00	4.182184e+05
2000-04-10	000001.SZ	20000410	19.01	19.30	18.70	18.82	18.91	-0.09	-0.4800	105059.00	1.995966e+05
...	...	...	...	...	...	...	...	...	...	...	...
2023-10-25	000001.SZ	20231025	10.51	10.54	10.36	10.38	10.55	-0.17	-1.6114	1411449.68	1.470972e+06
2023-10-26	000001.SZ	20231026	10.31	10.42	10.30	10.41	10.38	0.03	0.2890	599991.47	6.219153e+05
2023-10-27	000001.SZ	20231027	10.38	10.48	10.33	10.45	10.41	0.04	0.3842	919771.36	9.575875e+05
2023-10-30	000001.SZ	20231030	10.40	10.47	10.35	10.45	10.45	0.00	0.0000	805689.55	8.376460e+05
2023-10-31	000001.SZ	20231031	10.43	10.49	10.41	10.46	10.45	0.01	0.0957	652855.23	6.822121e+05

每日指标

名称	类型	描述
ts_code	str	TS股票代码
trade_date	str	交易日期
close	float	当日收盘价
turnover_rate	float	换手率（%）
turnover_rate_f	float	换手率（自由流通股）
volume_ratio	float	量比
pe	float	市盈率（总市值/净利润， 亏损的PE为空）
pe_ttm	float	市盈率（TTM，亏损的PE为空）
pb	float	市净率（总市值/净资产）
ps	float	市销率
ps_ttm	float	市销率（TTM）
dv_ratio	float	股息率 （%）
dv_ttm	float	股息率（TTM）（%）
total_share	float	总股本 （万股）
float_share	float	流通股本 （万股）
free_share	float	自由流通股本 （万）
total_mv	float	总市值 （万元）
circ_mv	float	流通市值（万元）

数据调用：

```
df2=pro.daily_basic(ts_code='000001.SZ',start_date='20000101',end_date='20231031')
```

trade_date	ts_code	trade_date	close	turnover_rate	turnover_rate_f	volume_ratio	pe	pe_ttm	pb	ps	ps_ttm	dv_ratio	dv_ttm	total_share	float_share	free_share	t
2000-04-04	000001.SZ	20000404	18.11	0.5931	0.7918	0.85	36.7690	44.8054	7.1739	14.3115	15.5785	4.7977	4.7977	1.551847e+05	1.071634e+05	80277.1362	2.810
2000-04-05	000001.SZ	20000405	18.35	0.5504	0.7347	0.85	37.2562	45.3992	7.2689	14.5011	15.7850	4.7350	4.7350	1.551847e+05	1.071634e+05	80277.1362	2.847
2000-04-06	000001.SZ	20000406	18.48	0.4894	0.6533	0.80	37.5202	45.7208	7.3204	14.6038	15.8968	4.7017	4.7017	1.551847e+05	1.071634e+05	80277.1362	2.867
2000-04-07	000001.SZ	20000407	18.91	2.0629	2.7538	3.59	38.3932	46.7847	7.4908	14.9437	16.2667	4.5972	4.5972	1.551847e+05	1.071634e+05	80277.1362	2.934
2000-04-10	000001.SZ	20000410	18.82	0.9804	1.3087	1.15	38.2105	46.5620	7.4551	14.8725	16.1893	4.6167	4.6167	1.551847e+05	1.071634e+05	80277.1362	2.920
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2023-10-25	000001.SZ	20231025	10.38	0.7273	1.6410	1.93	4.4256	4.1540	0.5088	1.1197	1.1901	2.1965	2.7457	1.940592e+06	1.940555e+06	860090.6226	2.014
2023-10-26	000001.SZ	20231026	10.41	0.3092	0.7352	0.68	4.4383	4.1660	0.5102	1.1230	1.1935	2.1902	2.7378	1.940592e+06	1.940555e+06	816042.7512	2.020
2023-10-27	000001.SZ	20231027	10.45	0.4740	1.1271	1.20	4.4554	4.1820	0.5122	1.1273	1.1981	2.1818	2.7273	1.940592e+06	1.940555e+06	816042.7512	2.027
2023-10-30	000001.SZ	20231030	10.45	0.4152	0.9873	0.95	4.4554	4.1820	0.5122	1.1273	1.1981	2.1818	2.7273	1.940592e+06	1.940555e+06	816042.7512	2.027
2023-10-31	000001.SZ	20231031	10.46	0.3364	0.8000	0.73	4.4597	4.1860	0.5127	1.1284	1.1992	2.1797	2.7247	1.940592e+06	1.940555e+06	816042.7512	2.029

5557 rows × 18 columns

数据质量

索引的设置以及时间格式的转换

```
df.index = pd.to_datetime(df.trade_date,format='%Y%m%d')  # 索引转为日期
df = df.iloc[::-1]  # 由于获取的数据是倒序的，需要将其调整为正序
df2.index=pd.to_datetime(df2.trade_date,format='%Y%m%d')
df2 = df2.iloc[::-1]
```

将上述获得的A股日线行情和每日指标的数据按时间合并，重复出现的列只保留一次

```
df = df2.combine_first(df)
```

trade_date	amount	change	circ_mv	close	dv_ratio	dv_ttm	float_share	free_share	high	low	...	pre_close	ps	ps_ttm	total_mv	total_share	trade_date
2000-04-04	1.154267e+05	-0.17	1.940730e+06	18.11	4.7977	4.7977	1.071634e+05	80277.1362	18.34	18.05	...	18.28	14.3115	15.5785	2.810395e+06	1.551847e+05	20000404
2000-04-05	1.078698e+05	0.24	1.966449e+06	18.35	4.7350	4.7350	1.071634e+05	80277.1362	18.67	18.05	...	18.11	14.5011	15.7850	2.847639e+06	1.551847e+05	20000405
2000-04-06	9.625639e+04	0.13	1.980380e+06	18.48	4.7017	4.7017	1.071634e+05	80277.1362	18.50	18.25	...	18.35	14.6038	15.8968	2.867813e+06	1.551847e+05	20000406
2000-04-07	4.182184e+05	0.43	2.026461e+06	18.91	4.5972	4.5972	1.071634e+05	80277.1362	19.20	18.41	...	18.48	14.9437	16.2667	2.934543e+06	1.551847e+05	20000407
2000-04-10	1.995966e+05	-0.09	2.016816e+06	18.82	4.6167	4.6167	1.071634e+05	80277.1362	19.30	18.70	...	18.91	14.8725	16.1893	2.920576e+06	1.551847e+05	20000410
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023-10-25	1.470972e+06	-0.17	2.014296e+07	10.38	2.1965	2.7457	1.940555e+06	860090.6226	10.54	10.36	...	10.55	1.1197	1.1901	2.014334e+07	1.940592e+06	20231025
2023-10-26	6.219153e+05	0.03	2.020117e+07	10.41	2.1902	2.7378	1.940555e+06	816042.7512	10.42	10.30	...	10.38	1.1230	1.1935	2.020156e+07	1.940592e+06	20231026
2023-10-27	9.575875e+05	0.04	2.027880e+07	10.45	2.1818	2.7273	1.940555e+06	816042.7512	10.48	10.33	...	10.41	1.1273	1.1981	2.027918e+07	1.940592e+06	20231027
2023-10-30	8.376460e+05	0.00	2.027880e+07	10.45	2.1818	2.7273	1.940555e+06	816042.7512	10.47	10.35	...	10.45	1.1273	1.1981	2.027918e+07	1.940592e+06	20231030
2023-10-31	6.822121e+05	0.01	2.029820e+07	10.46	2.1797	2.7247	1.940555e+06	816042.7512	10.49	10.41	...	10.45	1.1284	1.1992	2.029859e+07	1.940592e+06	20231031

查看数据的完整性

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5557 entries, 2000-04-04 to 2023-10-31
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   amount           5557 non-null   float64
 1   change           5557 non-null   float64
 2   circ_mv          5557 non-null   float64
 3   close            5557 non-null   float64
 4   dv_ratio         5557 non-null   float64
 5   dv_ttm           3645 non-null   float64
 6   float_share      5557 non-null   float64
 7   free_share       5557 non-null   float64
 8   high             5557 non-null   float64
 9   low              5557 non-null   float64
 10  open             5557 non-null   float64
 11  pb               5557 non-null   float64
 12  pct_chg          5557 non-null   float64
 13  pe               5557 non-null   float64
 14  pe_ttm           5557 non-null   float64
 15  pre_close        5557 non-null   float64
 16  ps               5557 non-null   float64
 17  ps_ttm           5557 non-null   float64
 18  total_mv         5557 non-null   float64
 19  total_share      5557 non-null   float64
 20  trade_date       5557 non-null   object 
 21  ts_code          5557 non-null   object 
 22  turnover_rate    5557 non-null   float64
 23  turnover_rate_f  5557 non-null   float64
 24  vol              5557 non-null   float64
 25  volume_ratio     5557 non-null   float64
dtypes: float64(24), object(2)
memory usage: 1.1+ MB
```

```
df.isnull().sum()
```

```
amount              0
change              0
circ_mv             0
close               0
dv_ratio            0
dv_ttm           1912
float_share         0
free_share          0
high                0
low                 0
open                0
pb                  0
pct_chg             0
pe                  0
pe_ttm              0
pre_close           0
ps                  0
ps_ttm              0
total_mv            0
total_share         0
trade_date          0
turnover_rate       0
turnover_rate_f     0
vol                 0
volume_ratio        0
dtype: int64
```

只有股息率（TTM）出现了空缺，但是后续训练不涉及该变量，因此无需进行特殊处理

> 股息率（TTM）是指股票每股分红的金额与股票当前价格之间的比率，表示过去12个月内的股息支付总额与当前股票价格之间的关系，通常以百分比形式表示。

数据统计特征

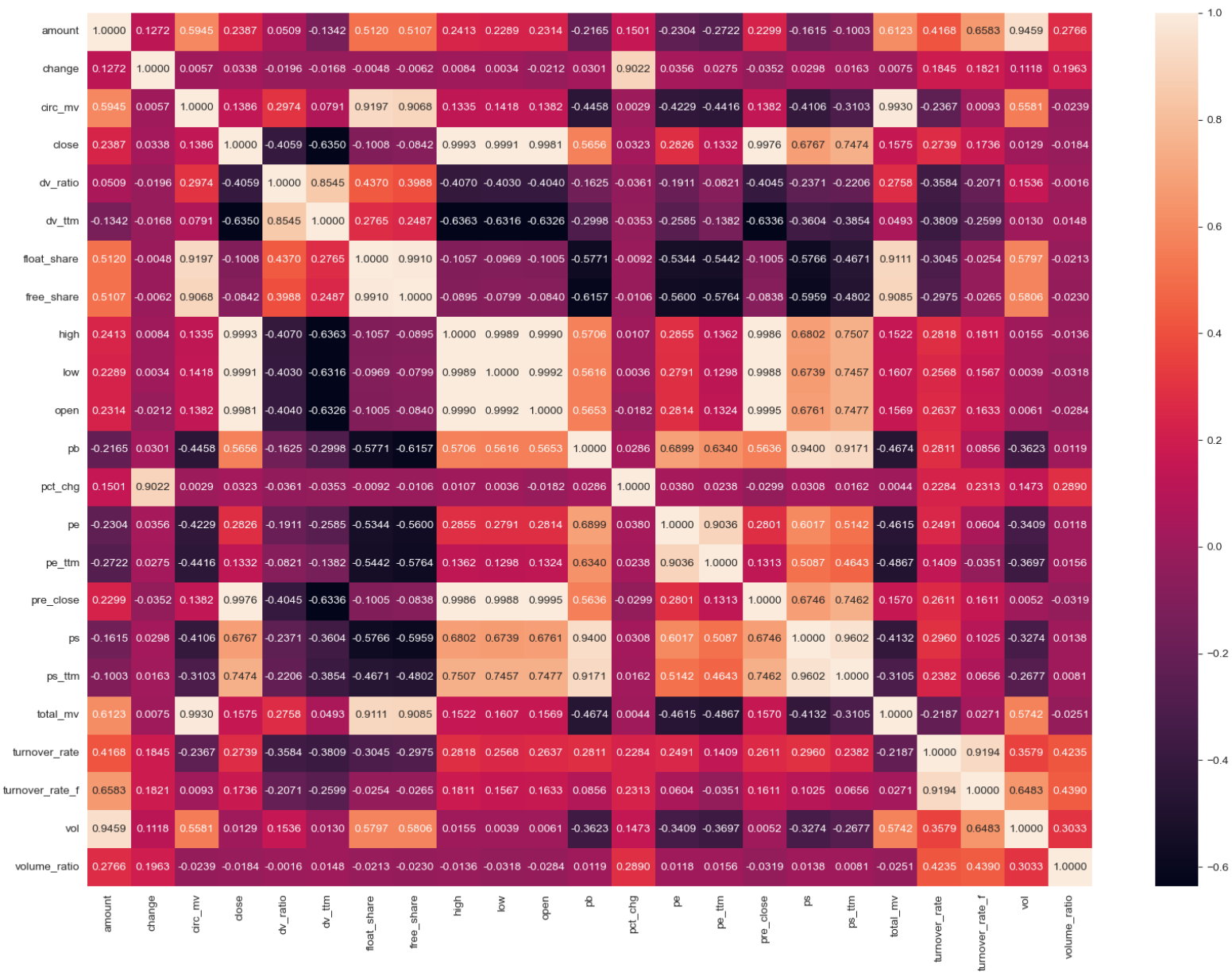
查看涨跌幅pct_chg和收盘价close的均值、最大值、最小值、标准差

```
df.describe().loc[['mean','min','max','std'],['pct_chg', 'close']]
```

	pct_chg	close
mean	0.046328	14.397756
min	-10.020000	5.100000
max	10.070000	48.050000
std	2.364594	6.075764

相关性分析

```
import seaborn as sns
new_df =
df[['amount','change','circ_mv','close','dv_ratio','dv_ttm','float_share','free_share','high','
low','open','pb','pct_chg','pe','pe_ttm','pre_close','ps','ps_ttm','total_mv','turnover_rate','
turnover_rate_f','vol','volume_ratio']].copy()
list_columns = new_df.columns
plt.figure(figsize=(20,14))
sns.heatmap(new_df[list_columns].corr(),annot=True,fmt='.4f',linewidths=0)
plt.show()
```



```
corr_1=new_df.corr()
corr_1['pct_chg'].sort_values(ascending=False)
```

输出pct_chg与其他变量的相关性，降序排列

```
pct_chg           1.000000
change            0.904646
volume_ratio      0.277488
turnover_rate_f   0.238510
turnover_rate     0.237411
amount            0.147310
vol               0.144310
close             0.033179
pe                0.031040
pe_ttm            0.017331
pb                0.016507
high              0.011443
total_mv          0.009019
circ_mv           0.007243
ps                0.004420
low               0.004045
ps_ttm            0.000805
float_share      -0.003621
free_share       -0.004075
open             -0.018042
dv_ratio         -0.026869
pre_close        -0.029851
dv_ttm           -0.034029
Name: pct_chg, dtype: float64
```

数据预处理

根据相关性分析进行数据的选择

```
data = df[['change', 'volume_ratio',
'turnover_rate_f','turnover_rate','amount','vol','pct_chg']].values.reshape(-1, 7)
```

	0	1	2	3	4	5	6
0	-0.17	0.85	0.7918	0.5931	115426.6791	63563.0	-0.93
1	0.24	0.85	0.7347	0.5504	107869.7505	58979.0	1.33
2	0.13	0.80	0.6533	0.4894	96256.3878	52445.0	0.71
3	0.43	3.59	2.7538	2.0629	418218.3845	221067.0	2.33
4	-0.09	1.15	1.3087	0.9804	199596.6004	105059.0	-0.48
5	0.08	1.00	1.2498	0.9363	190863.5385	100333.0	0.43
6	0.22	1.47	1.9638	1.4711	302649.2665	157650.0	1.16
7	0.01	0.75	1.1921	0.8930	182244.0072	95699.0	0.05

归一化处理

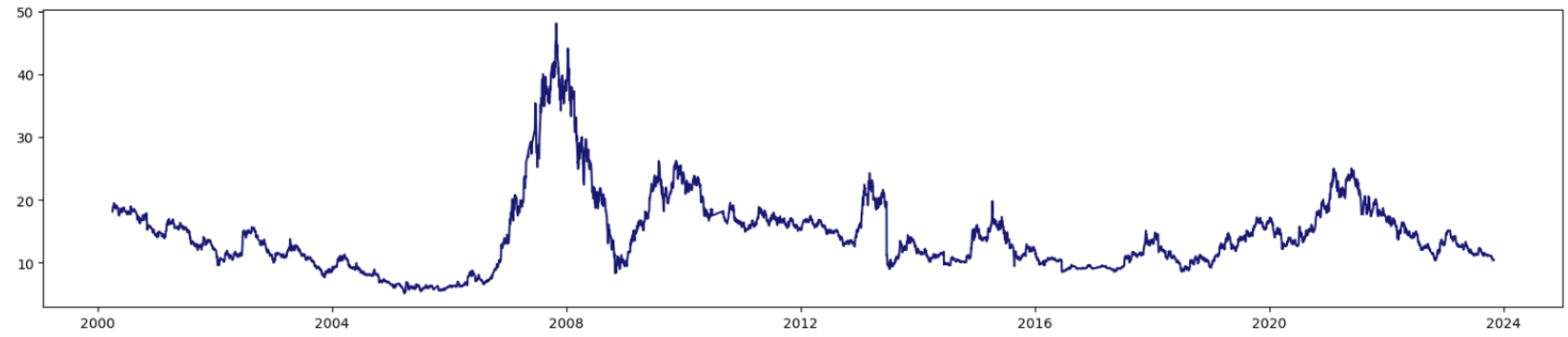
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

```
array([[0.45970149, 0.04122622, 0.06194629, ..., 0.01262686, 0.01079606,
        0.45246391],
       [0.52089552, 0.04122622, 0.05699163, ..., 0.01174713, 0.00989322,
        0.56495769],
       [0.50447761, 0.03858351, 0.04992841, ..., 0.01039516, 0.00860632,
        0.53409657],
       ...,
       [0.49104478, 0.05972516, 0.09104083, ..., 0.11066677, 0.17943057,
        0.51787954],
       [0.48507463, 0.04651163, 0.07891015, ..., 0.09670382, 0.15696159,
        0.4987556 ],
       [0.48656716, 0.03488372, 0.06265782, ..., 0.07860903, 0.12686011,
        0.50351916]])
```

数据可视化

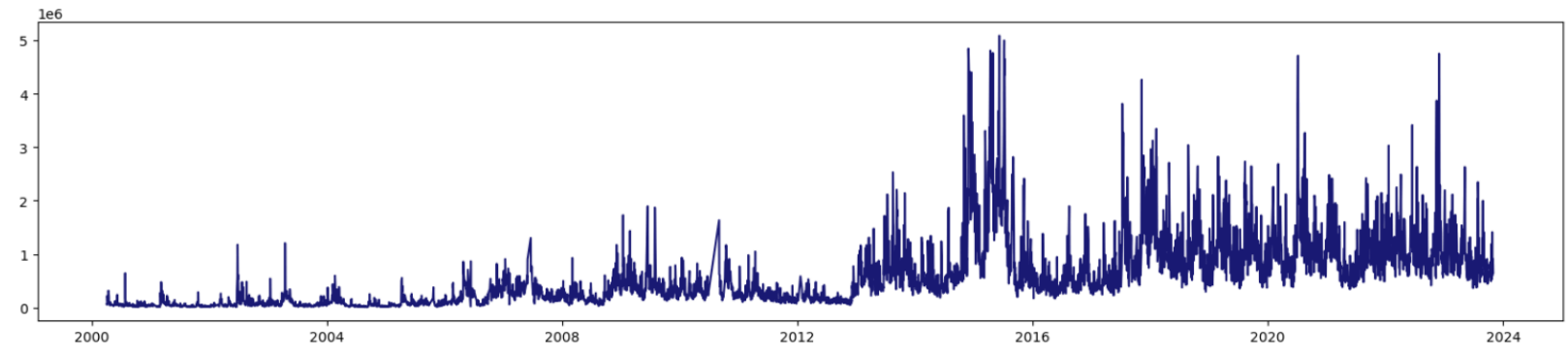
股票收盘价价格走势

```
plt.plot(df.index,df['close'])
```



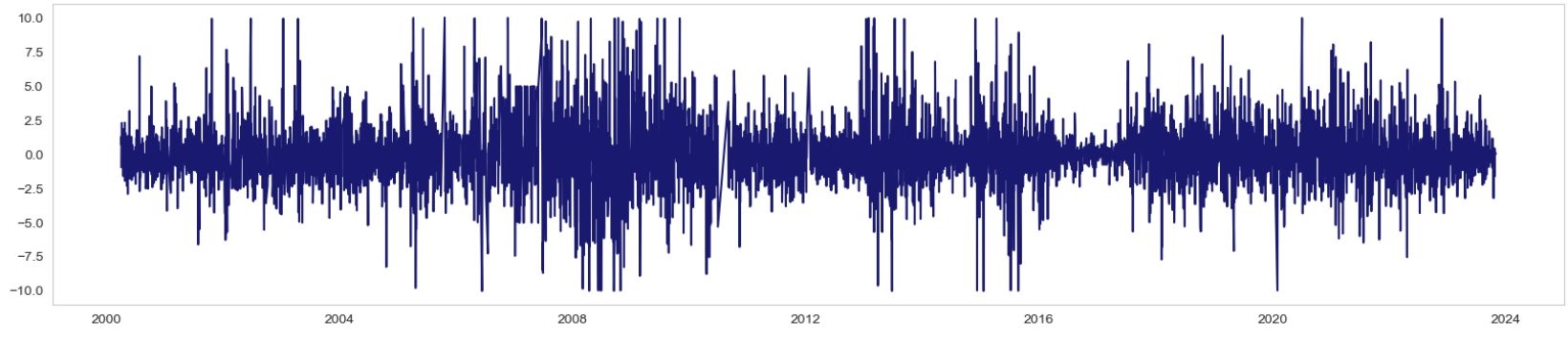
股票成交量变化图

```
plt.plot(df.index, df['vol'])
```



股票涨跌幅日变化

```
plt.plot(df.index,df['pct_chg'])
```



K线图绘制

导入mplfinance库

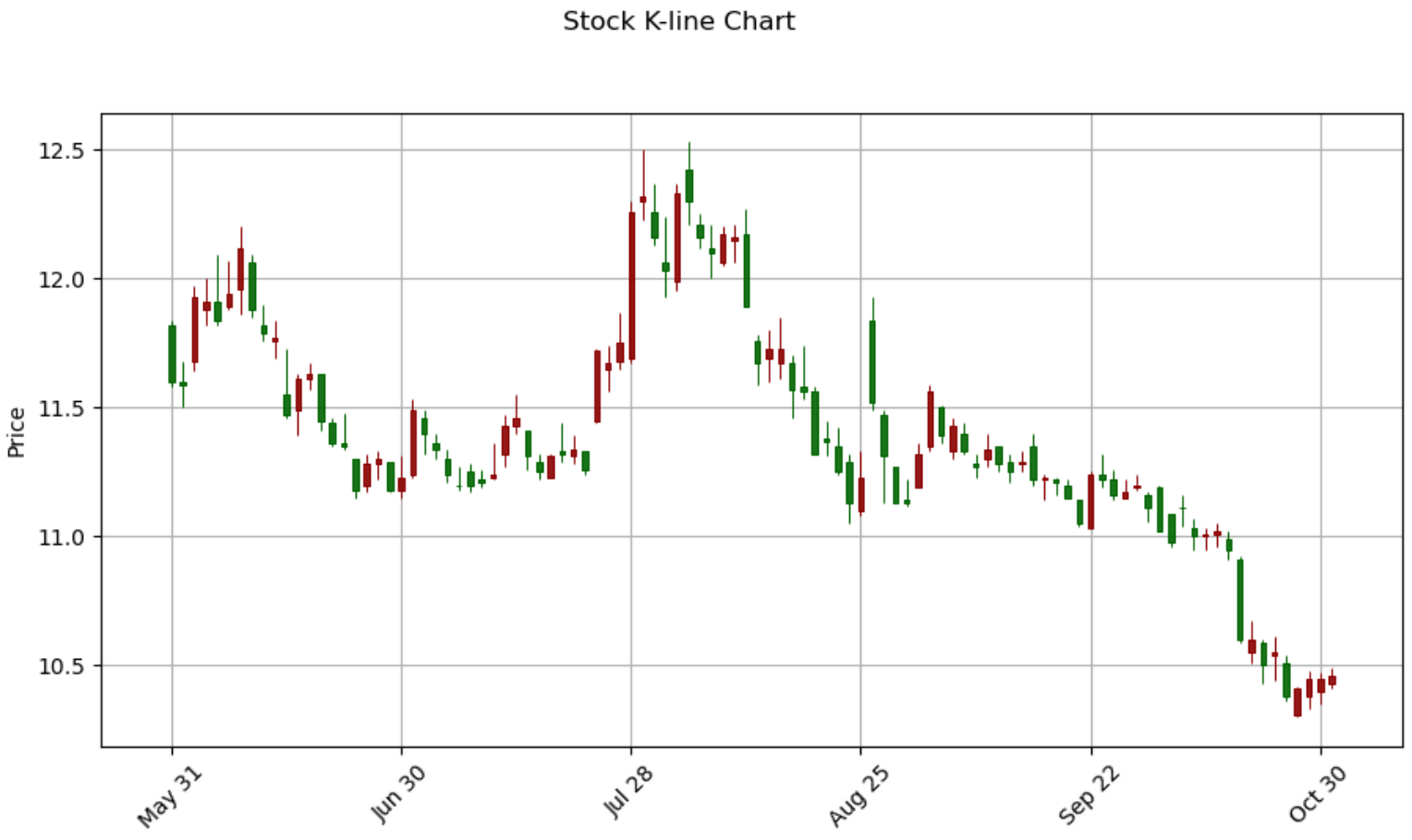
```
import mplfinance as mpf
```

时间范围选择

```
# 设置所需的时间范围，此处选择最近5个月的数据
start_date = '2023-5-31'
end_date = '2023-10-31'
df_plot = df.loc[start_date:end_date].copy()
```

画图

```
mpf.plot(df_plot, type='candle', style=s,title='Stock K-line Chart',figratio=
(20,10),returnfig=True)
mpf.show()
```



数据分析报告

```
from ydata_profiling import ProfileReport
profile=ProfileReport(df,title='Profiling Reader')
profile.to_file("report.html")
```

Overview

Overview | Alerts 28 | Reproduction

Dataset statistics

Number of variables	24
Number of observations	5557
Missing cells	1912
Missing cells (%)	1.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.1 MiB
Average record size in memory	200.0 B

Variable types

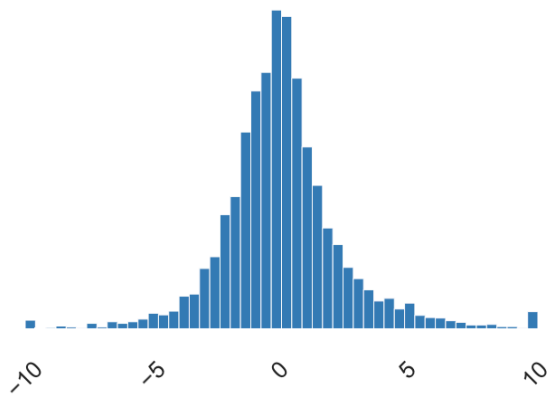
Numeric	24

涨跌幅pct_chg

pct_chg
Real number (ℝ)

`HIGH CORRELATION` `ZEROS`

Distinct	2219	**Minimum**	-10.02
Distinct (%)	39.9%	**Maximum**	10.07
Missing	0	**Zeros**	134
Missing (%)	0.0%	**Zeros (%)**	2.4%
Infinite	0	**Negative**	2778
Infinite (%)	0.0%	**Negative (%)**	50.0%
Mean	0.046327875	**Memory size**	86.8 KiB



Quantile statistics

Minimum	-10.02
5-th percentile	-3.38436
Q1	-1.17
median	0
Q3	1.05
95-th percentile	4.152
Maximum	10.07
Range	20.09
Interquartile range (IQR)	2.22

Descriptive statistics

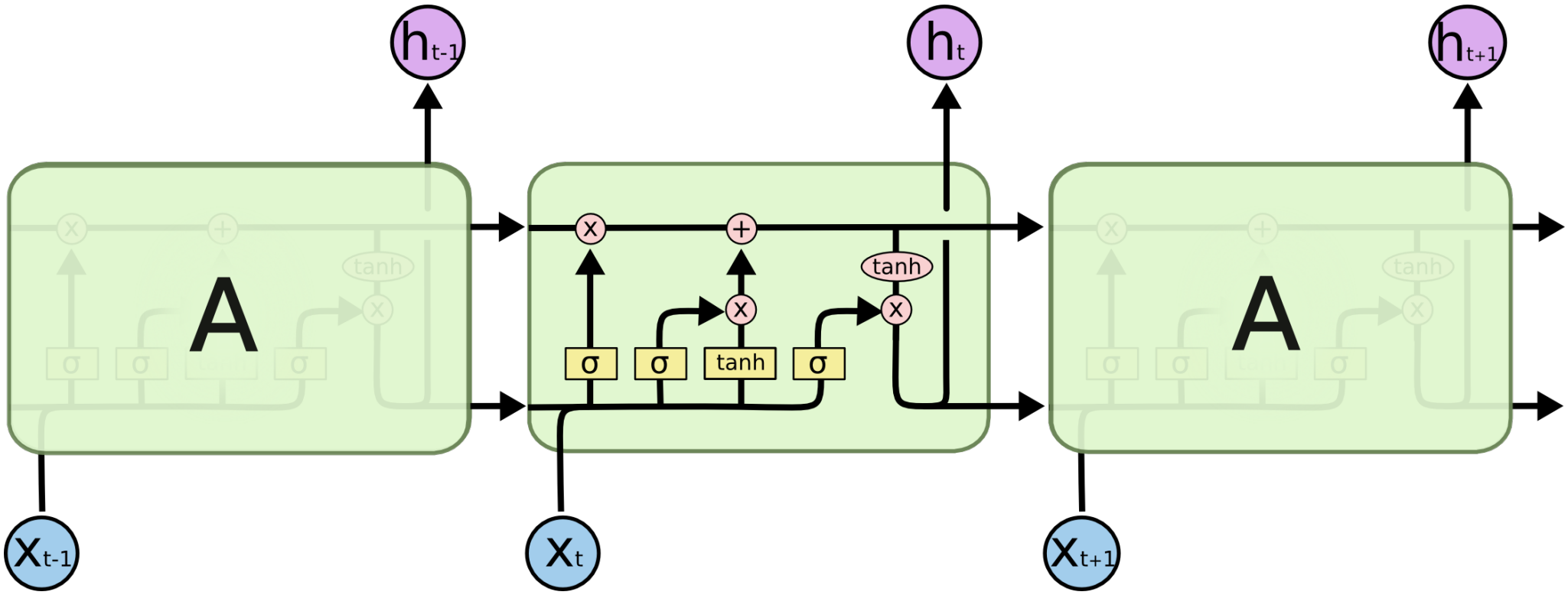
Standard deviation	2.3645935
Coefficient of variation (CV)	51.040406
Kurtosis	3.3609155
Mean	0.046327875
Median Absolute Deviation (MAD)	1.13
Skewness	0.42336572
Sum	257.444
Variance	5.5913026
Monotonicity	Not monotonic

涨跌幅大致位于-10%~+10%之间，呈现正态分布

3.建模

建模算法

LSTM（Long Short-Term Memory）神经网络模型，该模型可以处理时间序列数据，具有记忆能力。



LSTM算法原理

LSTM（Long Short-Term Memory）是一种长短期记忆网络，是一种特殊的RNN（循环神经网络）。与传统的RNN相比，LSTM更加适用于处理和预测时间序列中间隔较长的重要事件。

传统的RNN结构可以看做是多个重复的神经元构成的“回路”，每个神经元都接受输入信息并产生输出，然后将输出再次作为下一个神经元的输入，依次传递下去。这种结构能够在序列数据上学习短时依赖关系，但是由于梯度消失和梯度爆炸问题，RNN在处理长序列时难以达到很好的性能。

而LSTM通过引入记忆细胞、输入门、输出门和遗忘门的概念，能够有效地解决长序列问题。记忆细胞负责保存重要信息，输入门决定要不要将当前输入信息写入记忆细胞，遗忘门决定要不要遗忘记忆细胞中的信息，输出门决定要不要将记忆细胞的信息作为当前的输出。这些门的控制能够有效地捕捉序列中重要的长时间依赖性，并且能够解决梯度问题。

> - 输入门（input gate）: 决定了当前输入信息是否写入记忆细胞，从而控制输入信息对记忆细胞的影响。
> - 遗忘门（forget gate）: 决定了记忆细胞中的信息是否被遗忘，从而控制记忆细胞中保存的信息会不会消失。
> - 输出门（output gate）: 决定了记忆细胞中的信息是否输出，从而控制记忆细胞中保存的信息会不会对后面的网络层造成影响。
> - 通过计算权重矩阵和输入信号的点积，并通过激活函数计算出每个门的输出值，再乘上记忆细胞的值来进行最终计算。

为什么选择使用LSTM模型

- 金融市场具有时间序列性，即某一时刻的股票价格信息受到历史数据的影响，LSTM通过其记忆细胞的概念，能够记录历史信息并应用于当前预测，因此更适合处理时间序列数据。
- 金融市场具有不确定性和复杂性，很多内外部因素会影响股票的变化，LSTM能够捕捉到长时间依赖性并适应性地学习，更能够适应市场的多变性。
- 金融市场的长期趋势和短期波动等需要同时考虑，LSTM的结构可以同时考虑长期和短期信息，并且能够通过控制输入门、输出门、遗忘门和输入门来控制长期和短期信息的重要性，更好地处理金融数据。

需要注意的是，金融市场数据非常复杂且不确定，LSTM单独使用并不能保证高精度预测，通常需要结合其他技术如统计模型，深度学习等等来提高预测精度。

建模步骤与过程

导入相关依赖库

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout, BatchNormalization
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.optimizers import Adam
from keras.optimizers.legacy.rmsprop import RMSProp
```

数据集的准备及划分

将历史交易数据划分为训练集和测试集，使用时间序列中的滚动窗口方法

```
# 定义训练集和测试集的比例（时间轴前90%的数据用于训练，后10%的数据用于测试）
train_ratio = 0.90

# 计算划分的索引
train_size = int(len(scaled_data) * train_ratio)
test_size = len(scaled_data) - train_size

# 划分训练集和测试集
train_data = scaled_data[:train_size, :]
test_data = scaled_data[train_size:, :]

def create_dataset(dataset, time_steps=1):
    data_x, data_y = [], []
    for i in range(len(dataset) - time_steps):
        data_x.append(dataset[i:(i + time_steps), :])
        data_y.append(dataset[i + time_steps, -1])
    return np.array(data_x), np.array(data_y)
```

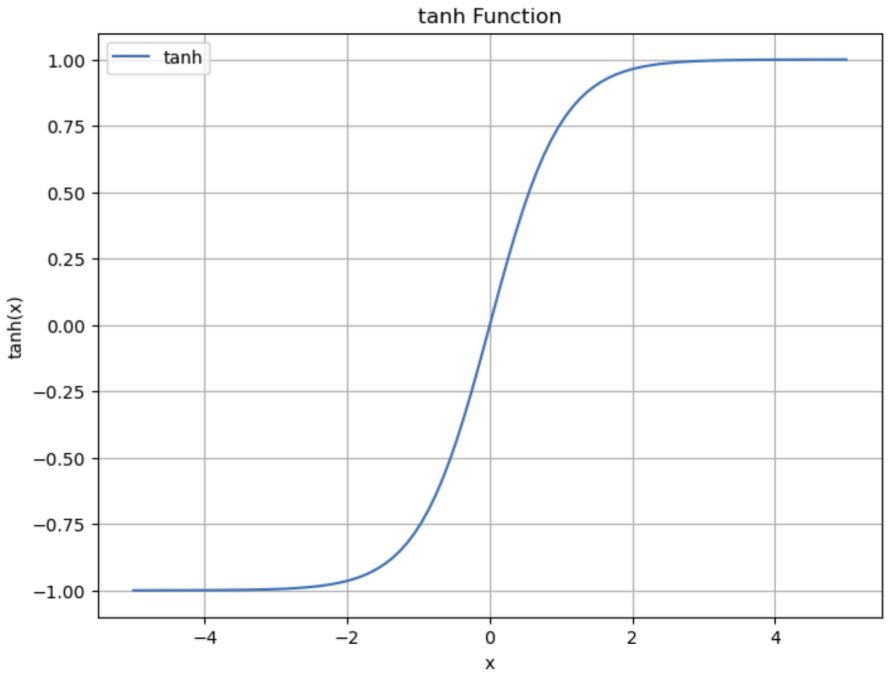
使用过去5天的数据来预测未来1天的数据

```
# 定义时间步长
time_steps = 5

# 创建训练集和测试集的数据集
train_x, train_y = create_dataset(train_data, time_steps)
test_x, test_y = create_dataset(test_data, time_steps)

# 将输入数据重塑为LSTM所需的三维形状
train_x = np.reshape(train_x, (train_x.shape[0], train_x.shape[1], train_x.shape[2]))
test_x = np.reshape(test_x, (test_x.shape[0], test_x.shape[1], test_x.shape[2]))
```

构建LSTM模型

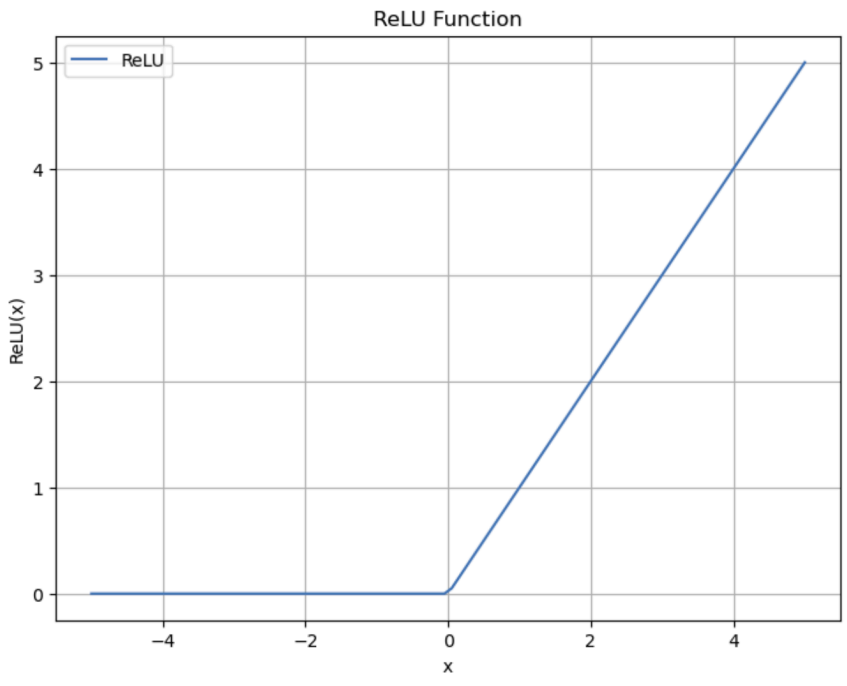
激活函数的选择

tanh



- tanh函数的输出范围介于-1和1之间，将输入值映射到[-1, 1]的区间。
- $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$。
- tanh函数具有S形曲线的特点，对于较大或较小的输入值，它会产生较大的梯度，在梯度下降中有助于更快地进行学习。
- tanh函数在输入为负时会输出负值，在输入为正时会输出正值，因此具有零中心化的特性，有助于模型的收敛。

`relu`



- ReLU函数的定义：当输入值大于0时，输出为输入值本身，否则输出为0。
- $ReLU(x) = max(0, x)$
- ReLU函数的优点是计算简单且具有稀疏激活性，能够更有效地处理大规模数据和深层网络。
- 缺点:对于负数输入，输出恒为0，产生“神经元死亡”现象，即该神经元无法更新权重。

本实验选择tanh作为激活函数

损失函数的选择

- 均方误差（MSE）：适用于回归问题，计算预测值与真实值之间的平方差的均值。
- 均绝对误差（MAE）：适用于回归问题，计算预测值与真实值之间的绝对差的均值。
- 二进制交叉熵（Binary Cross Entropy）：适用于二分类问题，计算预测概率与真实标签之间的交叉熵。
- 多类交叉熵（Categorical Cross Entropy）：适用于多分类问题，计算预测概率与真实标签之间的交叉熵。

本实验使用使用MSE作为损失函数

模型构建

```
# 定义LSTM模型
model = Sequential([
    LSTM(units=64, activation='tanh',return_sequences=True,input_shape=(time_steps,
train_x.shape[2])),
    Dropout(0.2),# 减少模型的过拟合
    LSTM(units=64,return_sequences=True),
    Dropout(0.2),
    LSTM(units=64),
    Dropout(0.2),
    Dense(units=1,activation='tanh')
])
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 5, 64)             18432

 dropout (Dropout)           (None, 5, 64)             0

 lstm_4 (LSTM)               (None, 5, 64)             33024

 dropout_1 (Dropout)         (None, 5, 64)             0

 lstm_5 (LSTM)               (None, 64)                33024

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 84,545
Trainable params: 84,545
Non-trainable params: 0
_________________________________________________________________
```

优化器的选择

```
# optimizers
adam_optimizer = Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
rmsprop_optimizer = RMSProp(learning_rate=0.001, decay=0.9)
```

Adam是一种自适应学习率优化器，使用一阶矩估计（均值）和二阶矩估计（方差）来自适应地调整每个参数的学习率。

- 一阶矩估计（均值）：Adam使用指数衰减的移动平均来估计每个参数梯度的一阶矩（均值），通过beta_1控制一阶矩的衰减速度，较高的值可以使梯度的衰减速度变慢。
- 二阶矩估计（方差）：Adam使用指数衰减的移动平均来估计每个参数梯度的二阶矩（方差），通过beta_2控制二阶矩的衰减速度，较高的值可以使梯度平方的衰减速度变慢。

RMSProp是一种基于梯度平方的自适应学习率优化器，使用了梯度平方的移动平均来调整每个参数的学习率，以适应不同特征的梯度变化。

- 梯度平方估计：RMSProp使用指数衰减的移动平均来估计每个参数梯度平方的二阶矩，通过衰减因子decay来控制历史梯度信息的衰减速度，较高的衰减因子将更快地降低过去梯度的影响。

本模型使用RMSProp优化器，相比较于Adam优化器，损失函数值下降更快，预测结果更加准确

模型的编译

```
model.compile(optimizer=rmsprop_optimizer, loss='mse')
```

模型训练

```
history=model.fit(train_x, train_y, validation_data=(test_x,test_y),epochs=100,
batch_size=64,callbacks=[es,mc])
```

EarlyStopping

早停：当损失函数值连续10个epoch不再下降时停止训练

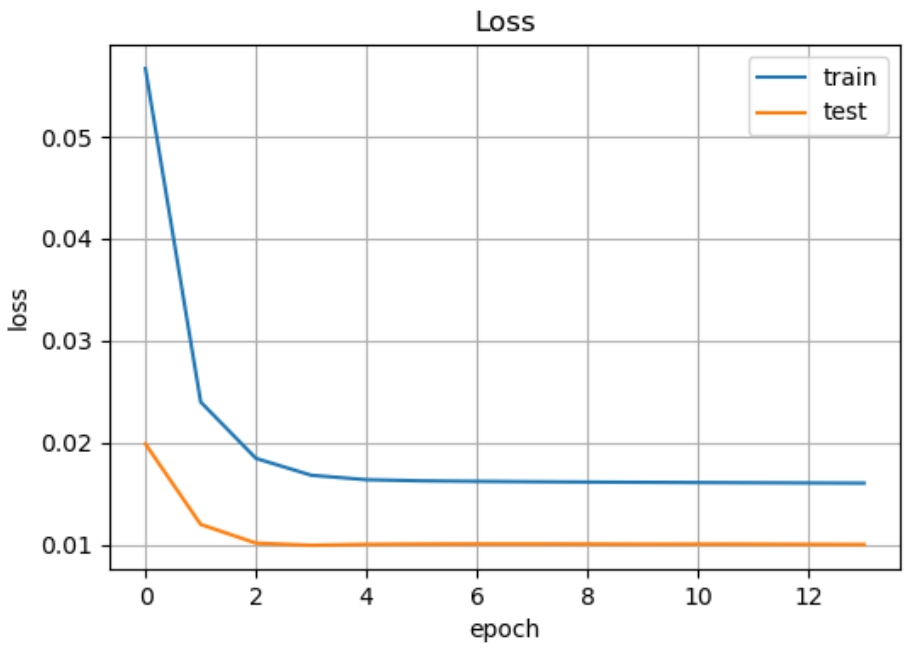
```
es = EarlyStopping(monitor='val_loss',patience=10,mode='min',verbose=1)
```

ModelCheckpoint

保存训练过程中损失值最低的模型

```
mc =
ModelCheckpoint('best_model.h5',monitor='val_loss',mode='min',save_best_only=True,verbose=1)
```

损失函数变化图



模型验证

导入记录的最优模型进行预测

```
model=keras.models.load_model('best_model.h5')
```

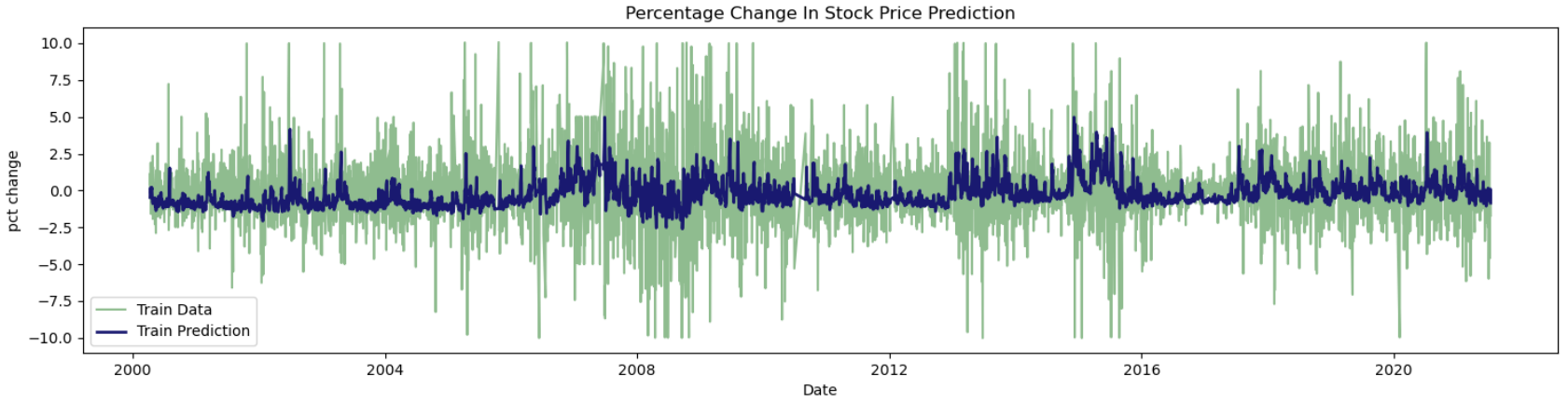
结果预测

```
train_pred = model.predict(train_x)
test_pred = model.predict(test_x)
```

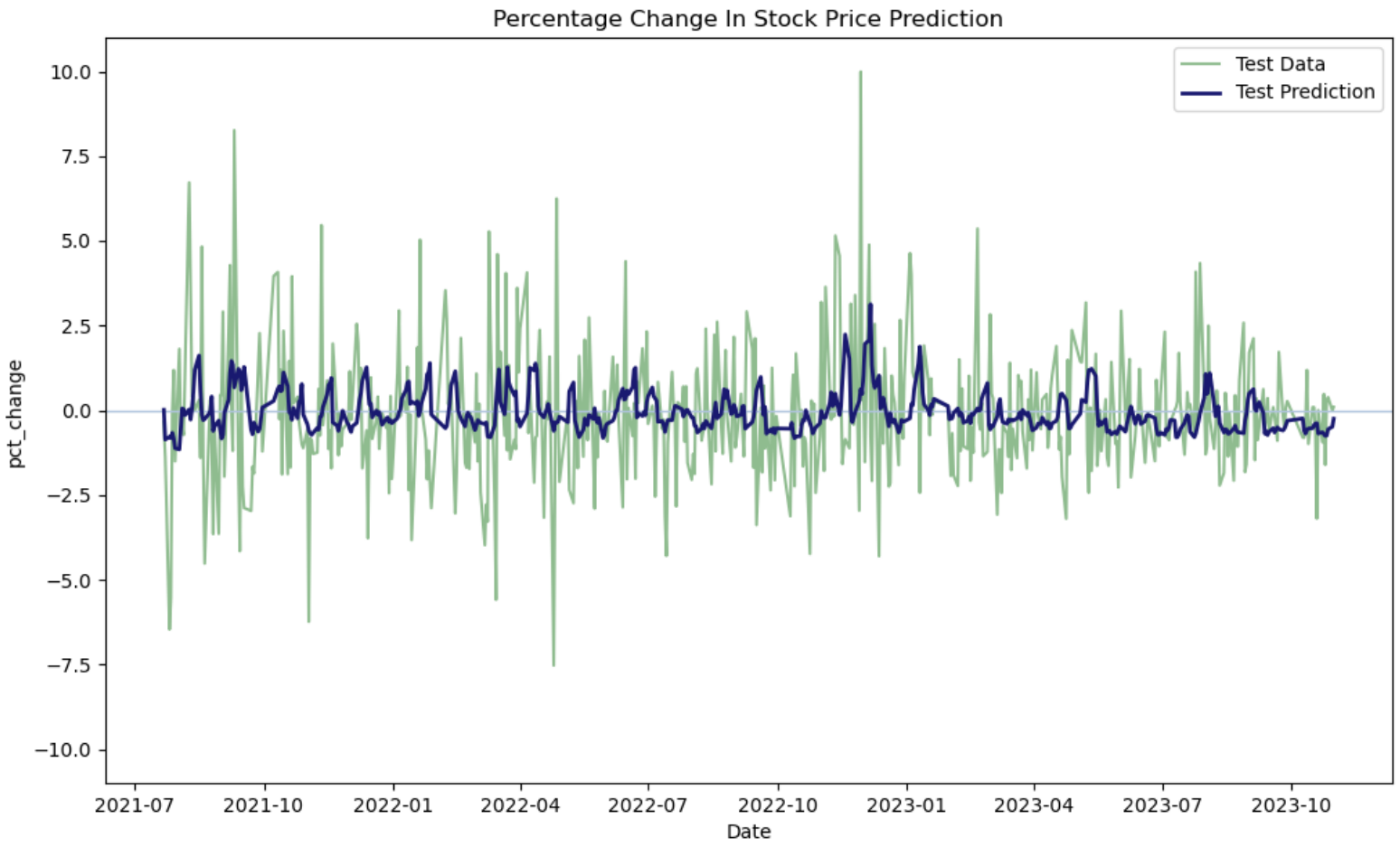
反归一化预测结果

```
train_pred = scaler.inverse_transform(np.concatenate((train_x[:, -1, :-1], train_pred),
axis=1))[:, -1]
test_pred = scaler.inverse_transform(np.concatenate((test_x[:, -1, :-1], test_pred), axis=1))
[:, -1]
```

训练集的真实值和预测值



测试集的真实值和预测值



模型评估

依赖库导入

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,mean_squared_error
```

均方根误差**RMSE**=1.9629347947973415

$$RMSE = sqrt(1/N * \Sigma(yi - \hat{yi})^2) \tag{1}$$

```
rmse = np.sqrt(mean_squared_error(test_pred, data[train_size + time_steps:, -1]))
```

平均绝对误差**MAE**=1.4254108480803112

$$MAE = 1/N * \Sigma|yi - \hat{yi}| \tag{2}$$

```
mae = mean_absolute_error(test_pred, data[train_size + time_steps:, -1])
```

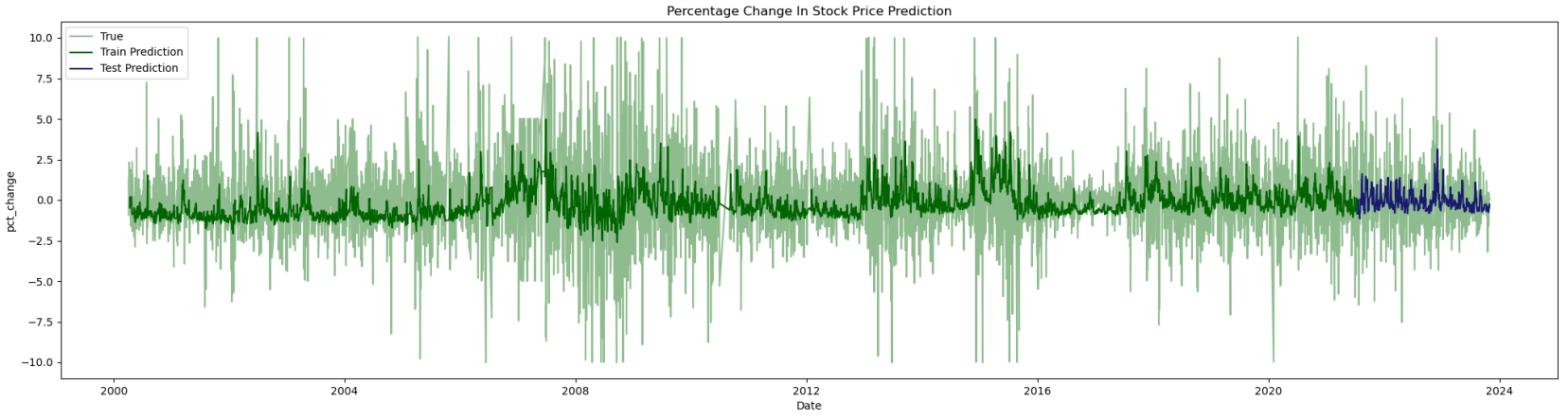
均方误差**MSE**=3.8531130086260807

$$MSE = 1/N * \Sigma(yi - \hat{yi})^2 \tag{3}$$

```
mse=mean_squared_error(test_pred, data[train_size + time_steps:,-1])
```

上述评价指标表明模型的预测误差相对较小，说明模型在预测数据时具有一定的准确性和精度。

4.知识发现与商业决策分析



股票市场分析

股票市场分析分为两部分：基本面分析和技术分析。

基本面分析涉及根据当前的商业环境和财务业绩分析公司未来的盈利能力，通过研究公司的基本经营情况、财务状况和产业竞争力等因素来评估公司的内在价值和长期发展趋势，目标是找到低估或高估的股票以及潜在的投资机会。

技术分析包括阅读图表和使用统计数据来确定股票市场的趋势，主要关注市场的供需关系和投资者的情绪以及价格走势的趋势和转折点，目标是捕捉短期价格波动和市场趋势以获取短期交易机会。

基本面分析和技术分析在股票分析中可以相互补充。基本面分析提供了对公司长期价值的评估，可以帮助投资者选择具有潜力的股票和行业。技术分析则提供了对短期价格走势的观察和预测，可以帮助投资者确定具体的买入和卖出时机。

本模型将重点放在技术分析部分，基于LSTM时间序列模型，对股票信息作多值量化分类，将股票跌涨幅预测转化为多维函数拟合问题。将股票的历史基本交易信息作为特征输入，并对股票价格的相对变化进行预测，其结果表明该模型具有相对较好的预测效果。

投资决策

股票趋势预测通常将股票走势分为上涨、下跌和横盘。股票的跌涨幅直接影响到投资回报率，预测股票跌涨幅有助于投资者进行风险管理和投资决策。

趋势交易策略

股票预测套利根据股票市场的涨跌趋势进行买入卖出操作，利用市场价格的不一致性或短期波动来实现风险低、收益稳定的交易。若当前市场处于上升趋势，投资者应适当增加仓位；若当前市场处于下降趋势，则应适当减少仓位或做空操作。

振荡交易策略

根据股票市场的波动幅度进行交易操作。在市场波动较大时，采用高抛低吸的策略，即在抛空行情中及时买入，而在牛市行情中适当卖出。

时间序列异常检测

由于股市具有不稳定性与可变性，可能给投资者带来重大风险。不稳定的例子包括系统性风险造成的市场崩溃和人为的大规模宣传造成的股价异常波动。时间严重性异常检测可以捕捉股市交易价格中的异常点，从而帮助投资者调整策略，降低投资风险，为股市预测提供便利。此外，该模型还可用于对多个财务时间序列数据集建模，并捕捉感兴趣公司中的异常。

预测股票价格趋势能够优化买卖时机，减少错误决策，确定合适的交易规模

股票预测复杂性

股票预测是一个复杂的任务。由于市场走势受到多种因素的影响，包括基本面因素、宏观经济因素和市场情绪等，是众多变量博弈的结果。因此股票预测具有阶段性与风险性。

有效市场假说

有效市场假说（Efficient Market Hypothesis，EMH）是金融经济学中的一个理论，提出了关于资本市场的假设。根据有效市场假说，资本市场是高度有效的，即市场上的股票价格已经充分反映了所有可得到的信息，投资者无法通过分析市场数据和信息来获得超额利润。其核心思想是市场参与者是理性的、富有信息的，并且会快速地将所有可得到的信息反映在股票价格中。因此，有效市场假说质疑了技术分析和基本面分析等投资策略的有效性。

复杂性因素

股票价格受到多种因素的影响，如公司基本面、市场需求、行业趋势、竞争态势等。同时股票市场受到众多随机因素的影响，如经济状况、政治事件、自然灾害、市场情绪等，因此具有随机性与不确定性。这些因素之间可能存在复杂的相互关系，需要全面考虑和分析。

股票价格的变化通常呈现非线性关系，不同时间段内的价格变化可能呈现出不同的模式。股票市场的数据通常存在噪声和不完整性，包括价格波动、交易量的波动、信息不对称等。这些因素会干扰模型的建立和预测的准确性。在建立股票预测模型时，存在过度拟合的问题，即模型在训练数据上表现良好，但在新数据上的预测能力较差。

股票市场具有短期波动和长期趋势两个层面。短期波动常常受到市场情绪和投机行为的影响，而长期趋势受到公司业绩和宏观经济因素的影响。准确预测这两个层面的变化都是具有挑战性的。

5.感想

案例分析感想

数据分析的效率

LSTM模型能够处理大量历史数据，并从中提取有用的信息。在股票分析中，这一点尤其重要，因为股票价格往往受到历史市场表现、新闻事件、公司业绩等多种因素的影响。通过使用LSTM模型，我们可以更高效地处理和分析这些数据，从而更好地预测未来的股票价格。

预测能力

LSTM模型在股票预测方面表现出了很好的效果。通过学习历史数据中的模式和趋势，模型可以生成对未来股票价格的预测。虽然这些预测并不总是100%准确，但它们可以为我们提供一种参考，帮助我们做出更明智的投资决策。

投资组合优化

除了预测单个股票的价格，LSTM模型还可以用于优化投资组合。通过分析各种股票的历史表现和相关性，模型可以确定最佳的投资组合配置，以实现更高的回报和更低的风险。

交易策略

基于LSTM模型的预测结果，我们可以制定出更有效的交易策略。例如，当模型预测某只股票即将上涨时，我们可以考虑购买它；当预测下跌时，则考虑卖出。这种策略可能无法保证100%的盈利，但它可以增加我们的投资成功率。

技术限制

虽然LSTM模型在股票分析中具有很多优点，但它们仍然受到一些技术限制。例如，模型可能无法捕捉到某些突发事件或非线性趋势的影响，这可能导致预测结果的不准确。此外，模型的训练和调优也需要一定的时间和计算资源。

人类专家的角色

尽管机器学习模型在股票分析中发挥着重要作用，但人类专家的角色仍然不可替代。人类专家可以提供更深入的市场洞察、行业知识和经验判断力，这些是机器学习模型难以模仿的。因此，将人类专家和机器学习模型结合起来，可以更好地应对股票市场的挑战。

课程感想

理论与实践结合

这门课不仅介绍了大数据商业分析的基本概念、技术和方法，还结合了大量的实际案例，通过理论与实际相结合，从而更好地理解了理论知识的应用。通过实际操作，学会了如何运用大数据技术解决商业问题。

数据分析的实用性

在课程中学习到许多实用的数据分析工具和方法，这些方法不仅可以帮助我更好地理解数据，还可以用于实际应用，例如更好地理解客户需求、市场趋势和竞争状况。同时这些数据处理和分析的方法不仅可以用于商业分析，也可以用于我们专业课，具有极高的实用性。

小组分工

姓名	学号	分工
邓栩瀛	21307035	数据处理，模型构建，报告撰写
肖瑶	22323103	PPT制作，知识发现与商业决策分析，报告撰写
徐子雅	22306066	PPT制作，感想，报告撰写