

上市公司股价数据分析

要实现对上市公司股价进行分析，可以使用Python的tushare接口来获取和处理股票数据。以下是一个可能的流程：

1. 获取数据：使用tushare接口获取指定上市公司的历史股价数据，包括开盘价、收盘价、最高价、最低价、成交量等信息。
2. 持久化数据：将获取到的股价数据存储在本地数据库中，以便后续的数据分析和处理。可以使用SQLite、MySQL等数据库。
3. 数据预处理：对获取到的股价数据进行预处理，包括数据清洗、缺失值填充、异常值处理等。可以使用pandas库进行数据预处理。
4. 分析数据：
 - a. 输出该股票所有收盘比开盘上涨3%以上的日期
 - b. 输出该股票所有开盘比前日收盘跌幅超过2%的日期
 - c. 计算收益：计算每日收益率和累计收益率
 - d. 计算每天的收益变化：计算每日收益率差异值
5. 绘图显示股票的价格、均线、RSI等

1. 通过tushare接口获取上市公司最近半年的历史股价数据，并将数据存储在本地数据库中

```
In [1]: import tushare as ts
import pandas as pd
import numpy as np
import sqlite3
from datetime import datetime, timedelta
import warnings
import matplotlib
import matplotlib.pyplot as plt

warnings.filterwarnings("ignore")
font = {'family': 'Microsoft YaHei', 'weight': 'bold', 'size': '12'}
matplotlib.rc("font", **font)

# 设置tushare token
ts.set_token('d4ade4dc2337bb63733c3ec90deb49bfe013ac6b81452bdba73641e7')
pro = ts.pro_api()

# 获取最近半年的日期范围
end_date = datetime.now().strftime('%Y%m%d')
start_date = (datetime.now() - timedelta(days=180)).strftime('%Y%m%d')

# 获取股票数据 (以平安银行000001.SZ为例)
df = pro.daily(ts_code='000001.SZ', start_date=start_date, end_date=end_date)

# 按日期升序排列
```

```
df = df.sort_values('trade_date').reset_index(drop=True)

print(f"获取数据时间范围: {start_date} 至 {end_date}")
print(f"共获取 {len(df)} 条记录")
print(df.head(10))
```

获取数据时间范围: 20251031 至 20260429

共获取 119 条记录

	ts_code	trade_date	open	high	low	close	pre_close	change	\
0	000001.SZ	20251031	11.38	11.40	11.30	11.32	11.38	-0.06	
1	000001.SZ	20251103	11.34	11.44	11.30	11.43	11.32	0.11	
2	000001.SZ	20251104	11.42	11.64	11.40	11.59	11.43	0.16	
3	000001.SZ	20251105	11.59	11.60	11.50	11.52	11.59	-0.07	
4	000001.SZ	20251106	11.50	11.58	11.47	11.51	11.52	-0.01	
5	000001.SZ	20251107	11.52	11.58	11.50	11.55	11.51	0.04	
6	000001.SZ	20251110	11.52	11.64	11.45	11.63	11.55	0.08	
7	000001.SZ	20251111	11.62	11.69	11.57	11.67	11.63	0.04	
8	000001.SZ	20251112	11.70	11.79	11.65	11.68	11.67	0.01	
9	000001.SZ	20251113	11.68	11.72	11.57	11.70	11.68	0.02	

	pct_chg	vol	amount
0	-0.5272	970192.93	1099179.193
1	0.9717	952326.43	1084422.504
2	1.3998	1503007.48	1737289.514
3	-0.6040	794926.05	918112.474
4	-0.0868	766585.46	882799.462
5	0.3475	734850.55	848177.984
6	0.6926	827208.33	957555.601
7	0.3439	886508.87	1033284.776
8	0.0857	1141325.38	1337963.805
9	0.1712	978991.03	1139765.204

```
In [2]: # 将数据存储到SQLite本地数据库
conn = sqlite3.connect('stock_data.db')
df.to_sql('stock_daily', conn, if_exists='replace', index=False)

# 验证数据已存储
df_check = pd.read_sql('SELECT * FROM stock_daily LIMIT 5', conn)
print("数据库中存储的数据: ")
print(df_check)
conn.close()
```

数据库中存储的数据:

	ts_code	trade_date	open	high	low	close	pre_close	change	\
0	000001.SZ	20251031	11.38	11.40	11.30	11.32	11.38	-0.06	
1	000001.SZ	20251103	11.34	11.44	11.30	11.43	11.32	0.11	
2	000001.SZ	20251104	11.42	11.64	11.40	11.59	11.43	0.16	
3	000001.SZ	20251105	11.59	11.60	11.50	11.52	11.59	-0.07	
4	000001.SZ	20251106	11.50	11.58	11.47	11.51	11.52	-0.01	

	pct_chg	vol	amount
0	-0.5272	970192.93	1099179.193
1	0.9717	952326.43	1084422.504
2	1.3998	1503007.48	1737289.514
3	-0.6040	794926.05	918112.474
4	-0.0868	766585.46	882799.462

2. 对获取到的股价数据进行预处理，包括数据清洗、缺失值填充、异常值处理等

```
In [3]: # 从数据库读取数据
conn = sqlite3.connect('stock_data.db')
df = pd.read_sql('SELECT * FROM stock_daily', conn)
conn.close()

# 查看数据基本信息
print("数据基本信息：")
print(df.info())
print("\n缺失值统计：")
print(df.isnull().sum())

# 缺失值填充（使用前向填充）
df = df.fillna(method='ffill')

# 异常值处理：去除价格为0或负数的记录
df = df[(df['open'] > 0) & (df['close'] > 0) & (df['high'] > 0) & (df['low'] > 0)]

# 重置索引
df = df.reset_index(drop=True)

print(f"\n预处理后数据量：{len(df)} 条")
print("\n预处理后数据统计描述：")
print(df[['open', 'high', 'low', 'close', 'vol']].describe())
```

数据基本信息:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119 entries, 0 to 118
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   ts_code     119 non-null    object
 1   trade_date  119 non-null    object
 2   open        119 non-null    float64
 3   high        119 non-null    float64
 4   low         119 non-null    float64
 5   close       119 non-null    float64
 6   pre_close   119 non-null    float64
 7   change      119 non-null    float64
 8   pct_chg     119 non-null    float64
 9   vol         119 non-null    float64
10  amount      119 non-null    float64
dtypes: float64(9), object(2)
memory usage: 10.4+ KB
None
```

缺失值统计:

```
ts_code      0
trade_date   0
open         0
high         0
low          0
close        0
pre_close    0
change       0
pct_chg      0
vol          0
amount       0
dtype: int64
```

预处理后数据量: 119 条

预处理后数据统计描述:

	open	high	low	close	vol
count	119.000000	119.000000	119.000000	119.000000	1.190000e+02
mean	11.238739	11.312437	11.173529	11.242941	9.039411e+05
std	0.316660	0.317607	0.316829	0.318421	3.652367e+05
min	10.520000	10.680000	10.430000	10.450000	4.061041e+05
25%	10.960000	11.035000	10.905000	10.960000	6.883779e+05
50%	11.160000	11.230000	11.110000	11.170000	8.340826e+05
75%	11.520000	11.595000	11.480000	11.530000	1.040694e+06
max	11.810000	11.990000	11.740000	11.850000	3.401276e+06

3. 筛选出该股票收盘价比前一日开盘价上涨超过3%的所有日期

```
In [4]: # 收盘价比前一日开盘价上涨超过3%的日期
df['prev_open'] = df['open'].shift(1)
df['close_vs_prev_open'] = (df['close'] - df['prev_open']) / df['prev_open'] * 100
```

```

rise_3pct = df[df['close_vs_prev_open'] > 3]
print("收盘价比前一日开盘价上涨超过3%的日期：")
print(rise_3pct[['trade_date', 'prev_open', 'close', 'close_vs_prev_open']].to_stri

```

收盘价比前一日开盘价上涨超过3%的日期：

trade_date	prev_open	close	close_vs_prev_open
20260325	10.52	10.94	3.992395
20260427	10.98	11.38	3.642987

4. 筛选出该股票开盘价比前一日收盘价下跌超过2%的所有日期

```

In [5]: # 开盘价比前一日收盘价下跌超过2%的日期
df['prev_close'] = df['close'].shift(1)
df['open_vs_prev_close'] = (df['open'] - df['prev_close']) / df['prev_close'] * 100

drop_2pct = df[df['open_vs_prev_close'] < -2]
print("开盘价比前一日收盘价下跌超过2%的日期：")
print(drop_2pct[['trade_date', 'prev_close', 'open', 'open_vs_prev_close']].to_stri

```

开盘价比前一日收盘价下跌超过2%的日期：

Empty DataFrame

Columns: [trade_date, prev_close, open, open_vs_prev_close]

Index: []

5. 计算该股票的每日收益率和一段时间内的累计收益率

```

In [6]: # 计算每日收益率
df['daily_return'] = df['close'].pct_change()

# 计算累计收益率
df['cumulative_return'] = (1 + df['daily_return']).cumprod() - 1

print("每日收益率和累计收益率：")
print(df[['trade_date', 'close', 'daily_return', 'cumulative_return']].to_string(in

print(f"\n期间累计收益率：{df['cumulative_return'].iloc[-1]*100:.2f}%")

```

每日收益率和累计收益率：

trade_date	close	daily_return	cumulative_return
20251031	11.32	NaN	NaN
20251103	11.43	0.009717	0.009717
20251104	11.59	0.013998	0.023852
20251105	11.52	-0.006040	0.017668
20251106	11.51	-0.000868	0.016784
20251107	11.55	0.003475	0.020318
20251110	11.63	0.006926	0.027385
20251111	11.67	0.003439	0.030919
20251112	11.68	0.000857	0.031802
20251113	11.70	0.001712	0.033569
20251114	11.75	0.004274	0.037986
20251117	11.67	-0.006809	0.030919
20251118	11.59	-0.006855	0.023852
20251119	11.80	0.018119	0.042403
20251120	11.85	0.004237	0.046820
20251121	11.69	-0.013502	0.032686
20251124	11.60	-0.007699	0.024735
20251125	11.80	0.017241	0.042403
20251126	11.69	-0.009322	0.032686
20251127	11.71	0.001711	0.034452
20251128	11.61	-0.008540	0.025618
20251201	11.69	0.006891	0.032686
20251202	11.64	-0.004277	0.028269
20251203	11.55	-0.007732	0.020318
20251204	11.49	-0.005195	0.015018
20251205	11.53	0.003481	0.018551
20251208	11.52	-0.000867	0.017668
20251209	11.43	-0.007812	0.009717
20251210	11.33	-0.008749	0.000883
20251211	11.37	0.003530	0.004417
20251212	11.35	-0.001759	0.002650
20251215	11.51	0.014097	0.016784
20251216	11.48	-0.002606	0.014134
20251217	11.53	0.004355	0.018551
20251218	11.64	0.009540	0.028269
20251219	11.62	-0.001718	0.026502
20251222	11.52	-0.008606	0.017668
20251223	11.56	0.003472	0.021201
20251224	11.54	-0.001730	0.019435
20251225	11.56	0.001733	0.021201
20251226	11.54	-0.001730	0.019435
20251229	11.56	0.001733	0.021201
20251230	11.48	-0.006920	0.014134
20251231	11.41	-0.006098	0.007951
20260105	11.50	0.007888	0.015901
20260106	11.67	0.014783	0.030919
20260107	11.64	-0.002571	0.028269
20260108	11.51	-0.011168	0.016784
20260109	11.46	-0.004344	0.012367
20260112	11.48	0.001745	0.014134
20260113	11.47	-0.000871	0.013251
20260114	11.36	-0.009590	0.003534
20260115	11.31	-0.004401	-0.000883
20260116	11.19	-0.010610	-0.011484

20260119	11.12	-0.006256	-0.017668
20260120	11.16	0.003597	-0.014134
20260121	11.07	-0.008065	-0.022085
20260122	11.07	0.000000	-0.022085
20260123	10.99	-0.007227	-0.029152
20260126	10.96	-0.002730	-0.031802
20260127	10.94	-0.001825	-0.033569
20260128	10.84	-0.009141	-0.042403
20260129	10.96	0.011070	-0.031802
20260130	10.83	-0.011861	-0.043286
20260202	10.86	0.002770	-0.040636
20260203	10.84	-0.001842	-0.042403
20260204	10.97	0.011993	-0.030919
20260205	11.09	0.010939	-0.020318
20260206	11.05	-0.003607	-0.023852
20260209	11.07	0.001810	-0.022085
20260210	11.06	-0.000903	-0.022968
20260211	11.07	0.000904	-0.022085
20260212	10.96	-0.009937	-0.031802
20260213	10.91	-0.004562	-0.036219
20260224	10.91	0.000000	-0.036219
20260225	10.86	-0.004583	-0.040636
20260226	10.87	0.000921	-0.039753
20260227	10.90	0.002760	-0.037102
20260302	10.85	-0.004587	-0.041519
20260303	10.88	0.002765	-0.038869
20260304	10.71	-0.015625	-0.053887
20260305	10.81	0.009337	-0.045053
20260306	10.82	0.000925	-0.044170
20260309	10.76	-0.005545	-0.049470
20260310	10.81	0.004647	-0.045053
20260311	10.89	0.007401	-0.037986
20260312	10.94	0.004591	-0.033569
20260313	10.93	-0.000914	-0.034452
20260316	10.92	-0.000915	-0.035336
20260317	11.03	0.010073	-0.025618
20260318	10.96	-0.006346	-0.031802
20260319	10.88	-0.007299	-0.038869
20260320	10.77	-0.010110	-0.048587
20260323	10.45	-0.029712	-0.076855
20260324	10.88	0.041148	-0.038869
20260325	10.94	0.005515	-0.033569
20260326	10.94	0.000000	-0.033569
20260327	11.02	0.007313	-0.026502
20260330	10.99	-0.002722	-0.029152
20260331	11.08	0.008189	-0.021201
20260401	11.15	0.006318	-0.015018
20260402	11.27	0.010762	-0.004417
20260403	11.12	-0.013310	-0.017668
20260407	11.03	-0.008094	-0.025618
20260408	11.22	0.017226	-0.008834
20260409	11.10	-0.010695	-0.019435
20260410	11.09	-0.000901	-0.020318
20260413	11.07	-0.001803	-0.022085
20260414	11.17	0.009033	-0.013251
20260415	11.21	0.003581	-0.009717

20260416	11.09	-0.010705	-0.020318
20260417	11.01	-0.007214	-0.027385
20260420	11.06	0.004541	-0.022968
20260421	11.08	0.001808	-0.021201
20260422	10.98	-0.009025	-0.030035
20260423	11.00	0.001821	-0.028269
20260424	11.00	0.000000	-0.028269
20260427	11.38	0.034545	0.005300
20260428	11.46	0.007030	0.012367

期间累计收益率：1.24%

6. 计算该股票每日收益率与前一日收益率之间的差异，并输出差异值

```
In [7]: # 计算每日收益率变化 (当日收益率 - 前一日收益率)
df['return_diff'] = df['daily_return'].diff()

print("每日收益率变化 (差异值): ")
print(df[['trade_date', 'daily_return', 'return_diff']].dropna().to_string(index=False))
```


每日收益率变化 (差异值):

trade_date	daily_return	return_diff
20251104	0.013998	4.280936e-03
20251105	-0.006040	-2.003794e-02
20251106	-0.000868	5.171634e-03
20251107	0.003475	4.343294e-03
20251110	0.006926	3.451168e-03
20251111	0.003439	-3.487026e-03
20251112	0.000857	-2.582483e-03
20251113	0.001712	8.554307e-04
20251114	0.004274	2.561176e-03
20251117	-0.006809	-1.108201e-02
20251118	-0.006855	-4.667359e-05
20251119	0.018119	2.497425e-02
20251120	0.004237	-1.388178e-02
20251121	-0.013502	-1.773940e-02
20251124	-0.007699	5.803222e-03
20251125	0.017241	2.494027e-02
20251126	-0.009322	-2.656341e-02
20251127	0.001711	1.103290e-02
20251128	-0.008540	-1.025057e-02
20251201	0.006891	1.543032e-02
20251202	-0.004277	-1.116777e-02
20251203	-0.007732	-3.454799e-03
20251204	-0.005195	2.537154e-03
20251205	0.003481	8.676093e-03
20251208	-0.000867	-4.348591e-03
20251209	-0.007812	-6.945197e-03
20251210	-0.008749	-9.364064e-04
20251211	0.003530	1.227936e-02
20251212	-0.001759	-5.289465e-03
20251215	0.014097	1.585593e-02
20251216	-0.002606	-1.670335e-02
20251217	0.004355	6.961830e-03
20251218	0.009540	5.184929e-03
20251219	-0.001718	-1.125854e-02
20251222	-0.008606	-6.887639e-03
20251223	0.003472	1.207807e-02
20251224	-0.001730	-5.202326e-03
20251225	0.001733	3.463206e-03
20251226	-0.001730	-3.463206e-03
20251229	0.001733	3.463206e-03
20251230	-0.006920	-8.653517e-03
20251231	-0.006098	8.228542e-04
20260105	0.007888	1.398538e-02
20260106	0.014783	6.894791e-03
20260107	-0.002571	-1.735330e-02
20260108	-0.011168	-8.597691e-03
20260109	-0.004344	6.824336e-03
20260112	0.001745	6.089249e-03
20260113	-0.000871	-2.616281e-03
20260114	-0.009590	-8.719155e-03
20260115	-0.004401	5.188827e-03
20260116	-0.010610	-6.208671e-03
20260119	-0.006256	4.354494e-03
20260120	0.003597	9.852708e-03

20260121	-0.008065	-1.166164e-02
20260122	0.000000	8.064516e-03
20260123	-0.007227	-7.226739e-03
20260126	-0.002730	4.496985e-03
20260127	-0.001825	9.049368e-04
20260128	-0.009141	-7.315950e-03
20260129	0.011070	2.021088e-02
20260130	-0.011861	-2.293142e-02
20260202	0.002770	1.463140e-02
20260203	-0.001842	-4.611704e-03
20260204	0.011993	1.383424e-02
20260205	0.010939	-1.053696e-03
20260206	-0.003607	-1.454578e-02
20260209	0.001810	5.416808e-03
20260210	-0.000903	-2.713297e-03
20260211	0.000904	1.807501e-03
20260212	-0.009937	-1.084093e-02
20260213	-0.004562	5.374722e-03
20260224	0.000000	4.562044e-03
20260225	-0.004583	-4.582951e-03
20260226	0.000921	5.503762e-03
20260227	0.002760	1.839079e-03
20260302	-0.004587	-7.347046e-03
20260303	0.002765	7.352133e-03
20260304	-0.015625	-1.838998e-02
20260305	0.009337	2.496207e-02
20260306	0.000925	-8.411999e-03
20260309	-0.005545	-6.470356e-03
20260310	0.004647	1.019213e-02
20260311	0.007401	2.753715e-03
20260312	0.004591	-2.809187e-03
20260313	-0.000914	-5.505445e-03
20260316	-0.000915	-8.363008e-07
20260317	0.010073	1.098817e-02
20260318	-0.006346	-1.641959e-02
20260319	-0.007299	-9.529419e-04
20260320	-0.010110	-2.811024e-03
20260323	-0.029712	-1.960187e-02
20260324	0.041148	7.086049e-02
20260325	0.005515	-3.563362e-02
20260326	0.000000	-5.514706e-03
20260327	0.007313	7.312614e-03
20260330	-0.002722	-1.003494e-02
20260331	0.008189	1.091159e-02
20260401	0.006318	-1.871573e-03
20260402	0.010762	4.444642e-03
20260403	-0.013310	-2.407200e-02
20260407	-0.008094	5.216147e-03
20260408	0.017226	2.531927e-02
20260409	-0.010695	-2.792094e-02
20260410	-0.000901	9.794286e-03
20260413	-0.001803	-9.025256e-04
20260414	0.009033	1.083685e-02
20260415	0.003581	-5.452403e-03
20260416	-0.010705	-1.428575e-02
20260417	-0.007214	3.491022e-03

```
20260420      0.004541  1.175503e-02
20260421      0.001808 -2.733008e-03
20260422     -0.009025 -1.083359e-02
20260423      0.001821  1.084676e-02
20260424      0.000000 -1.821494e-03
20260427      0.034545  3.454545e-02
20260428      0.007030 -2.751558e-02
```

7. 使用matplotlib绘图工具，将股票的价格、均线、RSI等指标绘制成图表

```
In [8]: # 计算均线
df['MA5'] = df['close'].rolling(window=5).mean()
df['MA10'] = df['close'].rolling(window=10).mean()
df['MA20'] = df['close'].rolling(window=20).mean()

# 计算RSI (14日)
delta = df['close'].diff()
gain = delta.where(delta > 0, 0)
loss = (-delta).where(delta < 0, 0)
avg_gain = gain.rolling(window=14).mean()
avg_loss = loss.rolling(window=14).mean()
rs = avg_gain / avg_loss
df['RSI'] = 100 - (100 / (1 + rs))

# 绘制股票价格和均线图
fig, axes = plt.subplots(3, 1, figsize=(14, 12), sharex=True)

# 子图1: 股价和均线
axes[0].plot(df['trade_date'], df['close'], label='收盘价', linewidth=1.5)
axes[0].plot(df['trade_date'], df['MA5'], label='MA5', linewidth=1)
axes[0].plot(df['trade_date'], df['MA10'], label='MA10', linewidth=1)
axes[0].plot(df['trade_date'], df['MA20'], label='MA20', linewidth=1)
axes[0].set_title('股票价格与均线')
axes[0].set_ylabel('价格')
axes[0].legend()
axes[0].grid(True, alpha=0.3)
tick_step = max(1, len(df) // 10)
axes[0].set_xticks(range(0, len(df), tick_step))

# 子图2: 成交量
axes[1].bar(df['trade_date'], df['vol'], color='steelblue', alpha=0.7)
axes[1].set_title('成交量')
axes[1].set_ylabel('成交量')
axes[1].grid(True, alpha=0.3)
axes[1].set_xticks(range(0, len(df), tick_step))

# 子图3: RSI指标
axes[2].plot(df['trade_date'], df['RSI'], label='RSI(14)', color='purple', linewidth=1)
axes[2].axhline(y=70, color='r', linestyle='--', alpha=0.5, label='超买线(70)')
axes[2].axhline(y=30, color='g', linestyle='--', alpha=0.5, label='超卖线(30)')
axes[2].set_title('RSI指标')
axes[2].set_ylabel('RSI')
axes[2].set_xlabel('日期')
```

```

axes[2].legend()
axes[2].grid(True, alpha=0.3)
axes[2].set_xticks(range(0, len(df), tick_step))

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

