Proof

February 13, 2025

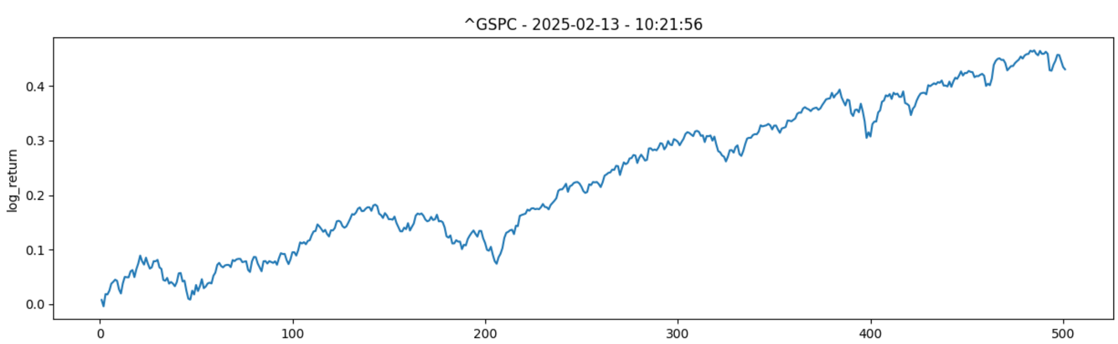
[41]:
```
import numpy as np
import yfinance as yf
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
```

1 Set up

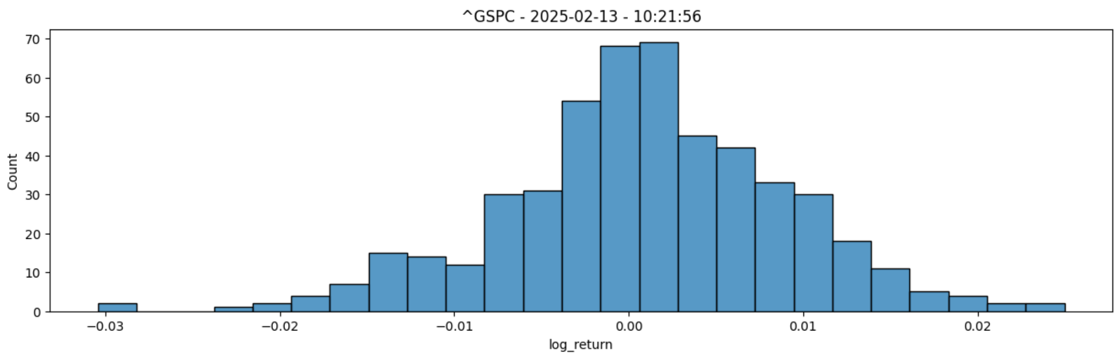
[42]:
```
START = "2023-01-01"
END = "2025-01-01"
TICKER = "^GSPC"
TITLE = TICKER + " - " +(datetime.now()).strftime("%Y-%m-%d - %H:%M:%S")
```

[43]:
```
data = pd.read_csv("data.csv")

data["log_return"] = np.log(data["Close"] / data["Close"].shift(1))

data.dropna(inplace=True)
```

[44]:
```
fig = plt.figure(figsize=(15,4))
sns.lineplot(data["log_return"].cumsum()).set_title(TITLE)
plt.show()
```



[45]:
```
fig = plt.figure(figsize=(15,4))
sns.histplot(data["log_return"]).set_title(TITLE)
plt.show()
```

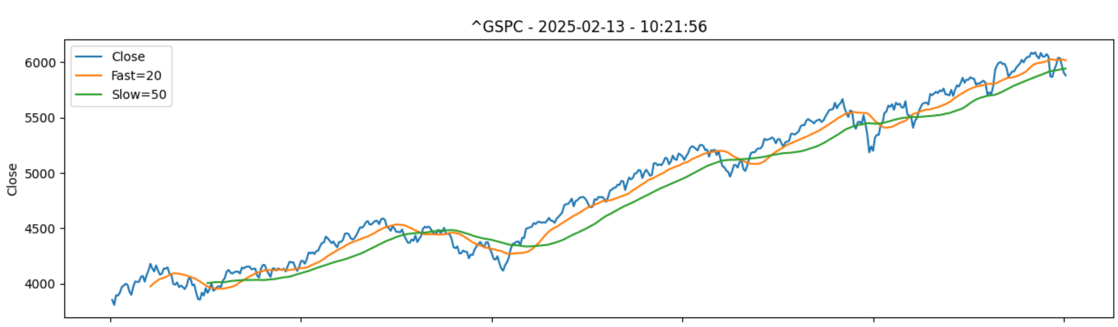


1.1 SMA

[46]:
```
def sma(data, t):
    return data.shift(1).rolling(t).mean()
```

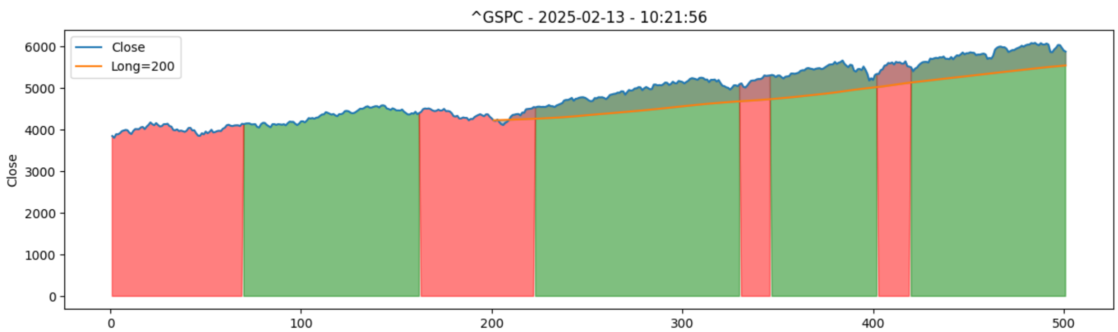
[47]:
```
Fast = 20
Slow = 50
Long = 200
data["SMA_fast"] = sma(data["Close"],Fast)
data["SMA_slow"] = sma(data["Close"],Slow)
data["SMA_long"] = sma(data["Close"],Long)
```

[48]:
```
fig = plt.figure(figsize=(15,4))
ax1 = sns.lineplot(data["Close"],label="Close")
ax2 = sns.lineplot(data["SMA_fast"],label=f"{Fast=}")
ax3 = sns.lineplot(data["SMA_slow"],label=f"{Slow=}")
ax1.set_title(TITLE)
plt.legend()
plt.show()
```



[49]:
```
data["bull"] = data["SMA_fast"] > data["SMA_slow"]
```

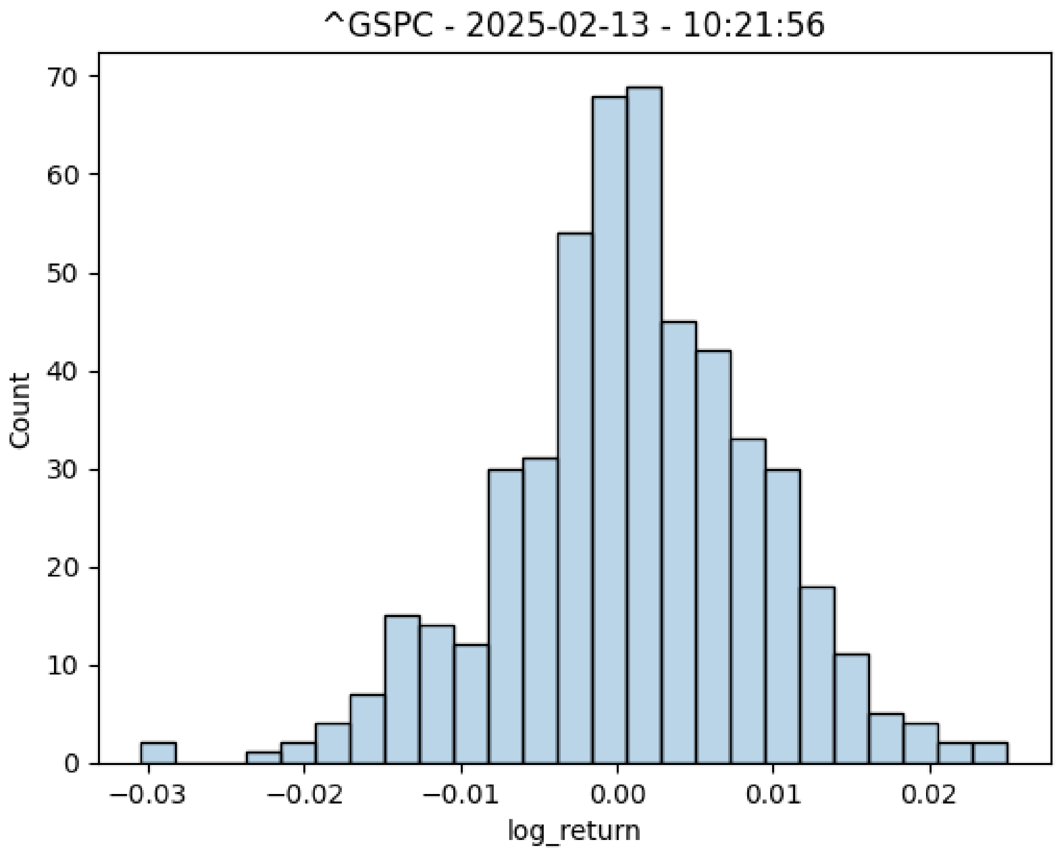
[50]:
```
fig = plt.figure(figsize=(15,4))
plt.fill_between(data.index, data["Close"], 0,␣
↪where=data["bull"],color="green",alpha=0.5,interpolate=True)
plt.fill_between(data.index, data["Close"], 0,␣
↪where=~data["bull"],color="red",alpha=0.5,interpolate=True)
plt.fill_between(data.index, data["Close"],␣
↪data["SMA_long"],color="grey",alpha=0.5,interpolate=True)
ax1 = sns.lineplot(data["Close"], label="Close")
ax1 = sns.lineplot(data["SMA_long"], label=f"{Long=}")
ax1.set_title(TITLE)
plt.show()
```



2 Bull and bear markets

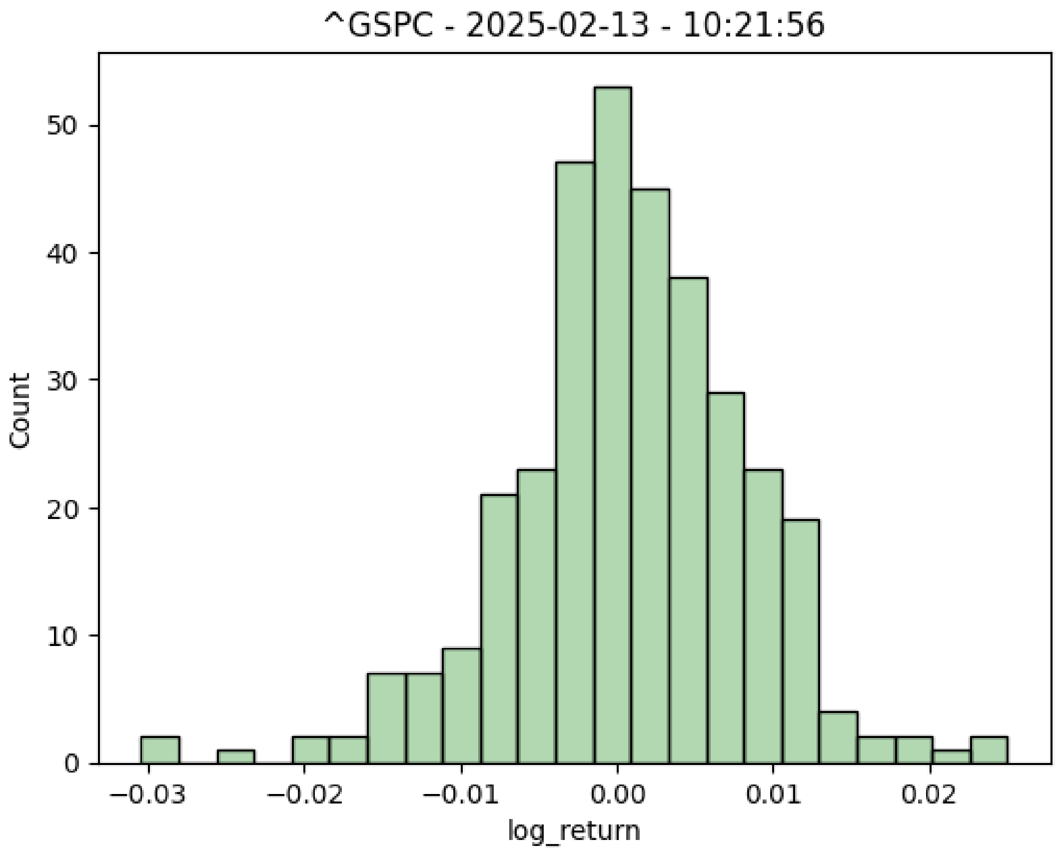
[51]:
```
sns.histplot(data["log_return"],alpha=0.3).set_title(TITLE)
```

[51]: `Text(0.5, 1.0, '^GSPC - 2025-02-13 - 10:21:56')`



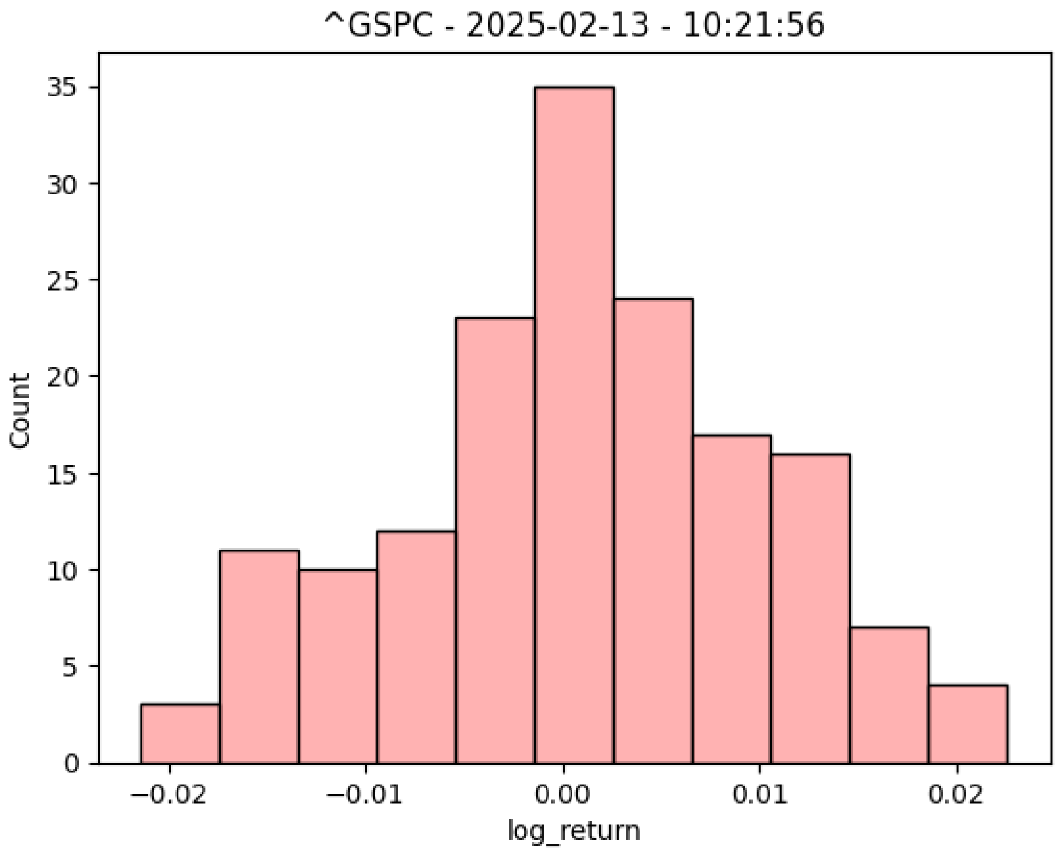
[52]:
```
sns.histplot(data["log_return"].where(data["bull"]),color="green",alpha=0.3).
    ↪set_title(TITLE)
```

[52]: Text(0.5, 1.0, '^GSPC - 2025-02-13 - 10:21:56')



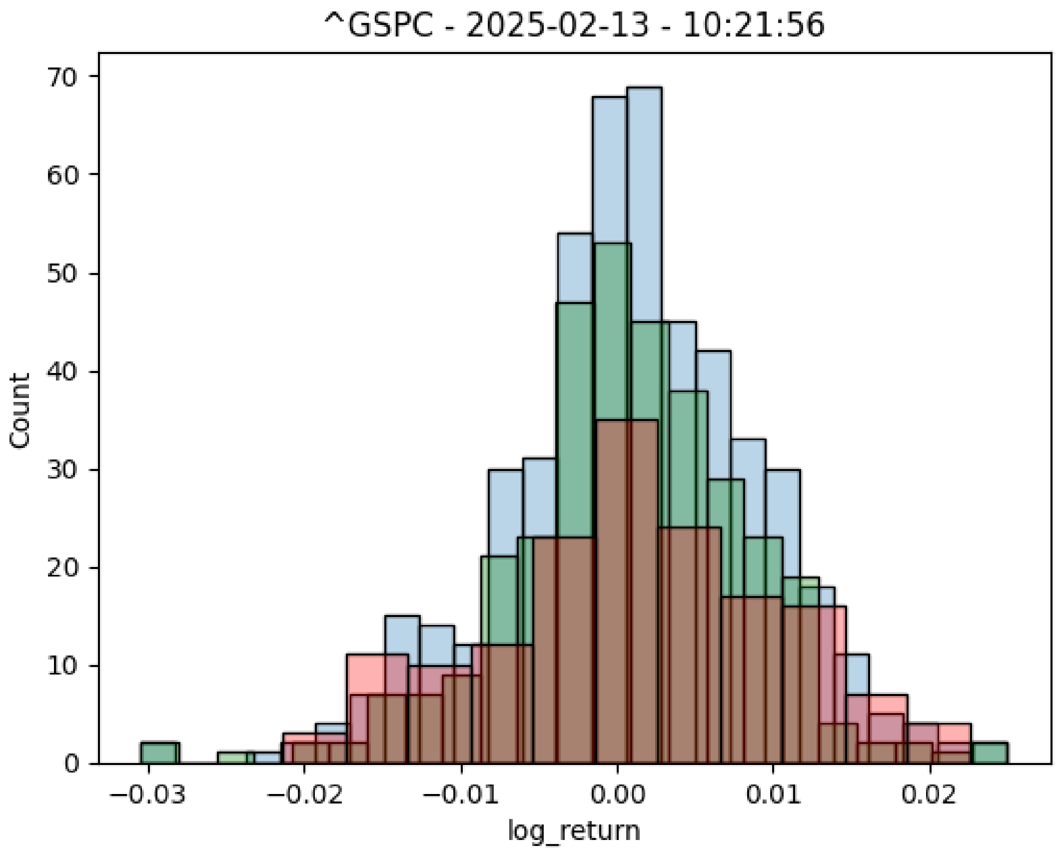
[53]:
```
sns.histplot(data["log_return"].where(~data["bull"]),color="red",alpha=0.3).
  ↪set_title(TITLE)
```

[53]: Text(0.5, 1.0, '^GSPC - 2025-02-13 - 10:21:56')



[54]:
```
sns.histplot(data["log_return"],alpha=0.3).set_title(TITLE)
sns.histplot(data["log_return"].where(data["bull"]),color="green",alpha=0.3).
↪set_title(TITLE)
sns.histplot(data["log_return"].where(~data["bull"]),color="red",alpha=0.3).
↪set_title(TITLE)
```

[54]:
```
Text(0.5, 1.0, '^GSPC - 2025-02-13 - 10:21:56')
```



3 1 - Descriptive statistics

[55]:
```
from scipy import stats
```

[56]:
```
def jarque_bera(data):
    # Reject Null if 0.05>
    return stats.jarque_bera(data.dropna()).pvalue


def describe_returns(data):
    mean = data.mean()
    std = data.std()
    skew = data.skew()
    kurt = data.kurt()
    jarque = jarque_bera(data)

    return pd.DataFrame({"Mean %" : [mean], "Standard deviation %": [std],␣
 ↪"Skew": [skew], "Kurtosis": [kurt], "Jarque-bera p value":[jarque]})
```

[57]:
```
both = describe_returns(data['log_return'])
bull = describe_returns(data['log_return'].where(data['bull']))
bear = describe_returns(data['log_return'].where(~data['bull']))
```

[58]:
```
combined = pd.concat([both,bull,bear])
combined.index = ["Both", "Bull", "Bear"]
combined["Mean %"] *= 100
combined["Standard deviation %"] *= 100

round(combined.T,4)
```

[58]:

	Both	Bull	Bear
Mean %	0.0859	0.0724	0.1141
Standard deviation %	0.8108	0.7567	0.9156
Skew	-0.2853	-0.4006	-0.1720
Kurtosis	0.7527	1.6462	-0.3532
Jarque-bera p value	0.0001	0.0000	0.4158

4 Students t-test on Bull and bear markets

This is a test for the null hypothesis that 2 independent samples have identical average (expected) values. **This test assumes that the populations have identical variances by default.**

The p-value quantifies the probability of observing as or more extreme values assuming the null hypothesis, that the samples are drawn from populations with the same population means, is true. A p-value larger than a chosen threshold (e.g. 5% or 1%) indicates that our observation is not so unlikely to have occurred by chance. Therefore, we do not reject the null hypothesis of equal population means. If the p-value is smaller than our threshold, then we have evidence against the null hypothesis of equal population means.

ttest_ind underestimates p for unequal variances:

[59]:
```
stats.ttest_ind(data["log_return"],data["log_return"].
 ↪where(data["bull"]),nan_policy="omit").pvalue
```

[59]:
```
np.float64(0.8081878403710501)
```

0.8>0.05 - Accept null

[60]:
```
stats.ttest_ind(data["log_return"],data["log_return"].
 ↪where(~data["bull"]),nan_policy="omit").pvalue
```

[60]:
```
np.float64(0.709493872061443)
```

0.71 > 0.05 - Accept null

5 Type errors

[61]:
```
true_positive = (data["bull"] & (data["log_return"] > 0)).value_counts().iloc[1]
true_negative = (~data["bull"] & (data["log_return"] < 0)).value_counts().
 ↪iloc[1]

false_postive = (data["bull"] & (data["log_return"] < 0)).value_counts().iloc[1]
false_negative = (~data["bull"] & (data["log_return"] > 0)).value_counts().
 ↪iloc[1]
```

6 Precision

TP + (TP + FP)

[62]:
```
true_positive / (true_positive + false_postive)
```

[62]:
```
np.float64(0.5545722713864307)
```

7 Accuracy

(TP + TN) / all

[63]:
```
(true_positive + true_negative) / (true_positive + true_negative +␣
 ↪false_negative + false_postive)
```

[63]:
```
np.float64(0.5149700598802395)
```

7.1 Balanced accuracy

[64]:
```
TPR = true_positive / (true_positive + false_negative)
TNR = true_negative / (true_negative + false_postive)

(TPR + TNR) / 2
```

[64]:
```
np.float64(0.4940853264382676)
```

8 Recall

[65]:
```
true_positive / (true_positive + false_negative)
```

[65]:
```
np.float64(0.6714285714285714)
```

9 F1 score

[66]:
```
(2*true_positive) / (2*(true_positive + false_postive + false_negative))
```

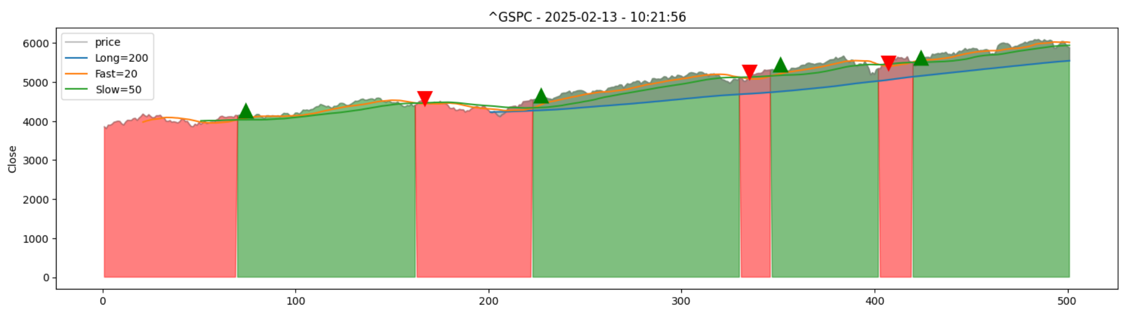
[66]: np.float64(0.4361948955916473)

10 Back testing

[67]:
```
last = 0
# [Date, Buy, Balance, Position]
equity = [(data.index[0],0,10_000,0)]
for index, row in data.iterrows():

    buy = 0
    position = equity[-1][3]
    balance = equity[-1][2]
    if row["bull"] and not last: # Buy
        buy = 1
        position = balance // row["Open"]
        balance -= row["Open"] * position

    if not row["bull"] and last: # Sell
        buy = -1
        balance += row["Open"] * position
        position = 0

    equity.append((index, buy, balance, position))
    last = row["bull"]

equity = pd.DataFrame(equity,columns=["Date","buy","balance","position"])

equity = equity.set_index("Date")

data = data.join(equity)

data["value"] = (data["position"] * data["Close"]) + data["balance"]
```

[68]:
```
fig, ax1 = plt.subplots(figsize=(15,4))

plt.fill_between(data.index, data["Close"], 0,␣
 ↪where=data["bull"],color="green",alpha=0.5,interpolate=True)
plt.fill_between(data.index, data["Close"], 0,␣
 ↪where=~data["bull"],color="red",alpha=0.5,interpolate=True)
plt.fill_between(data.index, data["Close"],␣
 ↪data["SMA_long"],color="grey",alpha=0.5,interpolate=True)
```

```python
ax1 = sns.lineplot(data["Close"],color="grey",alpha=0.5, label="price")
ax1 = sns.lineplot(data["SMA_long"],label=f"{Long=}")
ax1 = sns.lineplot(data["SMA_fast"], label=f"{Fast=}")
ax1 = sns.lineplot(data["SMA_slow"], label=f"{Slow=}")
# sns.lineplot(data["value"])

df_filt = data[data["buy"] != 0.0]

for index, value in df_filt.iterrows():
    if value["buy"] == -1.0:
        ax1.annotate(" ",(index,value["Close"]), color="red",fontsize=20)
    if value["buy"] == 1.0:
        ax1.annotate(" ",(index,value["Close"]), color="green",fontsize=20)

fig.tight_layout()
ax1.set_title(TITLE)
plt.show()
```



[69]:
```python
# data.to_excel("test.xlsx")
```

[70]:
```python
data["log_return"].sum()
```

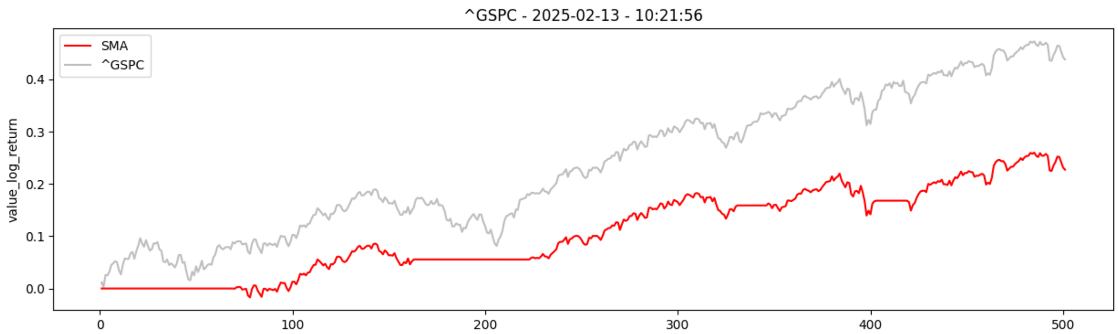
[70]: np.float64(0.43801103170967715)

[71]:
```python
data["value_log_return"] = np.log((data["value"] / data["value"].shift(1)))

data["value_log_return"].sum()
```

[71]: np.float64(0.22729803448569985)

[72]:
```python
fig, ax1 = plt.subplots(figsize=(15,4))
sns.lineplot(data["value_log_return"].
 ↪cumsum(),dashes=False,label="SMA",color="red").set_title(TITLE)
sns.lineplot(data["log_return"].cumsum(),dashes=False,label=TICKER,alpha=0.
 ↪5,color="grey",zorder=0)
```

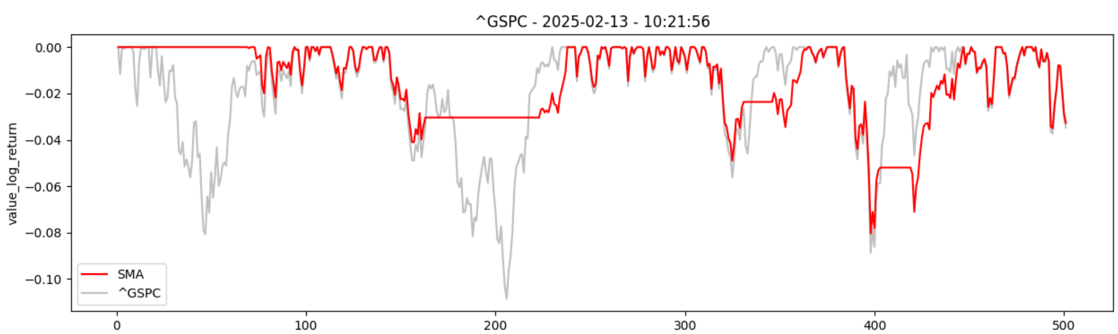
[72]: <Axes: title={'center': '^GSPC - 2025-02-13 - 10:21:56'}, ylabel='value_log_return'>



11 Drawdown

[73]:
```
fig, ax1 = plt.subplots(figsize=(15,4))
sns.lineplot(data["value_log_return"].cumsum() - data["value_log_return"].
↪cumsum().cummax(),dashes=False,label="SMA",color="red").set_title(TITLE)
sns.lineplot(data["log_return"].cumsum() - data["log_return"].cumsum().
↪cummax(),dashes=False,label=TICKER, alpha=0.5,color="grey",zorder=0)
```

[73]: <Axes: title={'center': '^GSPC - 2025-02-13 - 10:21:56'}, ylabel='value_log_return'>



12 Portfolio stats (annual)

[74]:
```
data["log_return"].sum() / 5
```

[74]: np.float64(0.08760220634193543)

[75]:
```
data["value_log_return"].sum() / 5
```

[75]:
```
np.float64(0.04545960689713997)
```

[76]:
```
data["log_return"].std() * np.sqrt(252*5)
```

[76]:
```
np.float64(0.2877027298666383)
```

[77]:
```
data["value_log_return"].std() * np.sqrt(252*5)
```

[77]:
```
np.float64(0.19620880942660335)
```

[78]:
```
((data["log_return"].mean()) / (data["log_return"].std())) * np.sqrt(252)
```

[78]:
```
np.float64(1.7089245619860631)
```

[79]:
```
(data["value_log_return"].mean() / data["value_log_return"].std()) * np.
↪sqrt(252)
```

[79]:
```
np.float64(1.3029417334047433)
```