Hybrid Finance research analysis

Project: Hybrid classical–quantum pipeline for short-term AAPL forecasting.

Components: SMA20/SMA50, Monte Carlo (GBM), PnL envelopes, Black–Scholes, QAE/QAOA.

Goal: Empirical evaluation of predictive signals and quantum extensions.

Goal: Empirical evaluation of predictive signals and quantum extensions.

Data source: https://massive.com/ (Polygon.io)

MD MUJAHIDUL ISLAM MAHFUZ
23229550

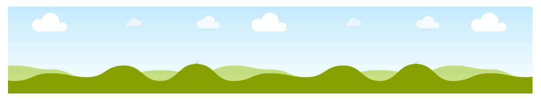
RQ1: SMA Crossover (Classical Technical Analysis)
RQ: Does the SMA20/SMA50 crossover provide a reliable short-term buy signal for AAPL?
H1: The SMA20 crossing above SMA50 is positively associated with short-term upward price movement.
Why it's important:
- Foundational baseline — you're testing a well-known market heuristic.
- Provides a benchmark for all later probabilistic, Monte Carlo, and quantum approaches.
- Statistically clean and easy to visualize (price vs SMA curves, t-tests, return distributions).

RQ2: Monte Carlo vs SMA
RQ: Does the Monte Carlo "buy probability" improve prediction of short-term returns compared to historical SMA signals alone?
H2: Higher Monte Carlo buy probability correlates with higher expected next-minute returns.
Why it's important:
- Bridges traditional signals (SMA) with probabilistic simulation (Monte Carlo).
- Allows statistical comparison (correlation, regression, t-tests).
- Serves as your "probabilistic enhancement" step before adding quantum models.

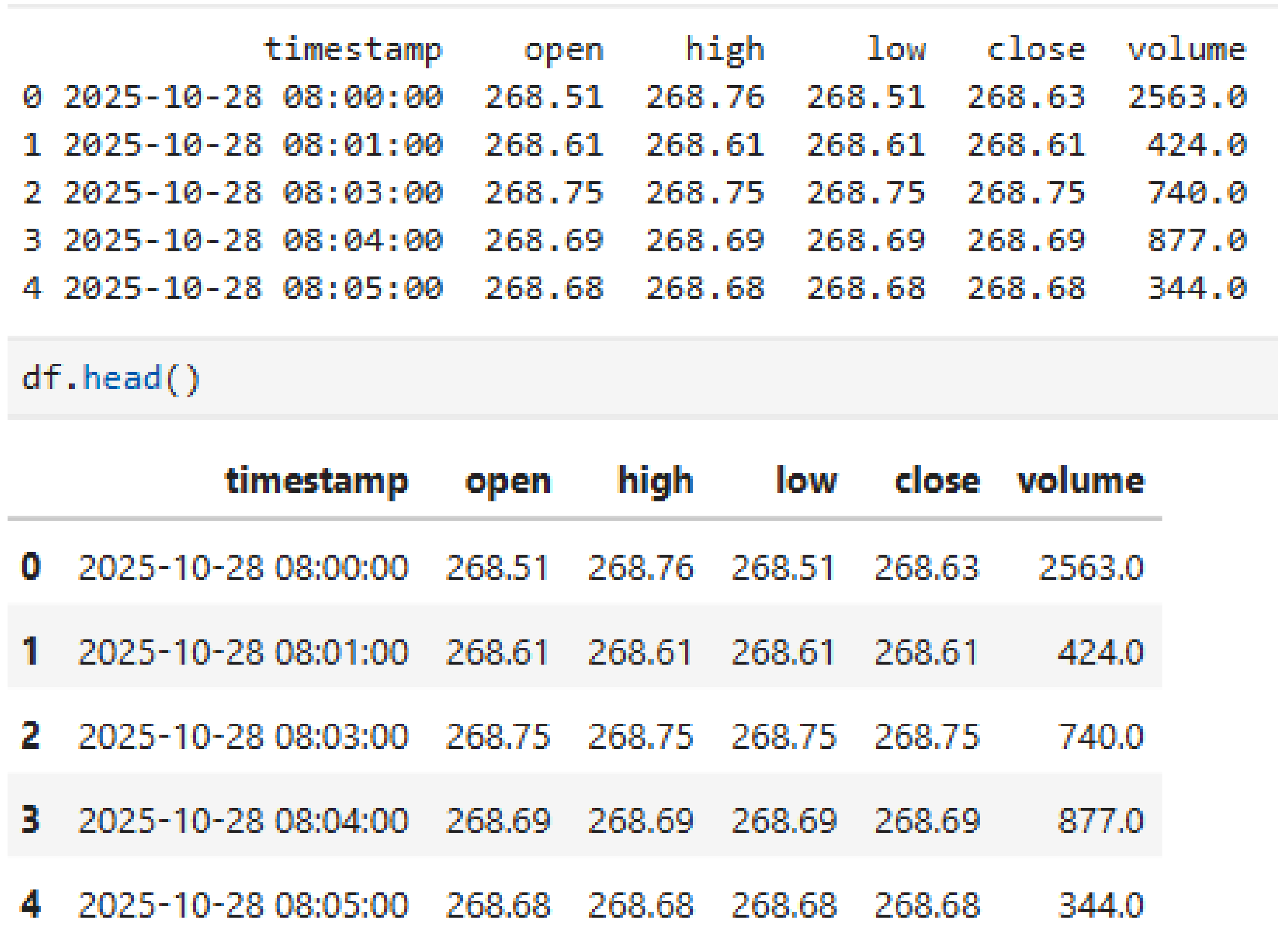
RQ3: PnL-Envelope / Hybrid Option Valuation
RQ: Can the PnL-envelope / hybrid signals be translated into option-style valuations?
H3: When the strategy's risk_confidence is high (5th-percentile PnL > current price), the expected short-dated call option payoff (via Monte Carlo) is larger than unconditional/low-confidence cases — and may exceed the Black–Scholes fair price.
Why it's important:
- Translates trading signals into derivative-like valuations, showing potential tradable edge.
- Integrates Monte Carlo, Black–Scholes, and risk quantiles — all advanced yet interpretable.
- Excellent demonstration of applied financial innovation.

RQ4: 5–95% Cumulative PnL Envelope
RQ: Does the 5–95% cumulative PnL envelope indicate potential risk and reward for the SMA strategy?
H4: When the 5th percentile of cumulative PnL rises above current price, the probability of a positive return increases.
Why it's important:
- Visually powerful: PnL bands represent risk–reward zones.
- Connects quantile-based risk modeling to actual trading signal strength.
- Can be tested using simple descriptive stats + p-values from return distributions.

RQ5: Quantum Extension (QAE/QAOA)
RQ: Can Quantum Amplitude Estimation (QAE) or Quantum Approximate Optimization Algorithm (QAOA) improve the estimation of return probabilities or portfolio optimization compared to classical methods?
H5: Quantum algorithms provide statistically more efficient probability estimation or optimization of expected return/risk ratios than classical counterparts.
Why it's important:
- Connects your entire pipeline (SMA, MC, BS) to your quantum computing research direction.
- Demonstrates futuristic relevance — fits perfectly with your profile and ongoing research.
- Makes the paper unique and forward-looking.



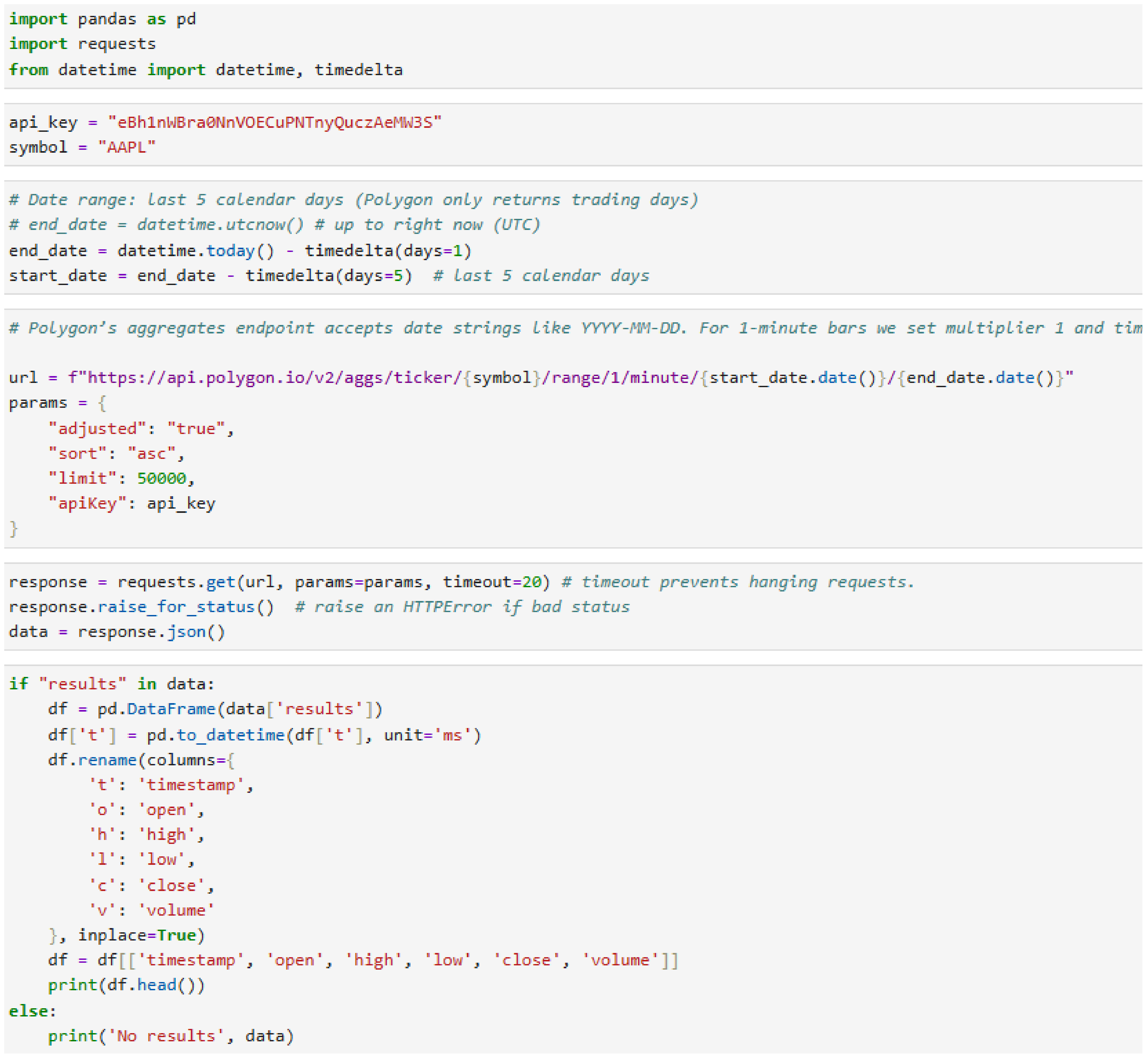
I got the data from massive.com, formerly known as polygon.io, with the limit of 5000 data at one call. The data I fetched is timeseries, output 1-minute OHLCV.



```
            timestamp    open    high     low   close  volume
0 2025-10-28 08:00:00  268.51  268.76  268.51  268.63  2563.0
1 2025-10-28 08:01:00  268.61  268.61  268.61  268.61   424.0
2 2025-10-28 08:03:00  268.75  268.75  268.75  268.75   740.0
3 2025-10-28 08:04:00  268.69  268.69  268.69  268.69   877.0
4 2025-10-28 08:05:00  268.68  268.68  268.68  268.68   344.0
```

```python
df.head()
```

	timestamp	open	high	low	close	volume
0	2025-10-28 08:00:00	268.51	268.76	268.51	268.63	2563.0
1	2025-10-28 08:01:00	268.61	268.61	268.61	268.61	424.0
2	2025-10-28 08:03:00	268.75	268.75	268.75	268.75	740.0
3	2025-10-28 08:04:00	268.69	268.69	268.69	268.69	877.0
4	2025-10-28 08:05:00	268.68	268.68	268.68	268.68	344.0



```python
import pandas as pd
import requests
from datetime import datetime, timedelta
```

```python
api_key = "eBh1nWBra0NnVOECuPNTnyQuczAeMW3S"
symbol = "AAPL"
```

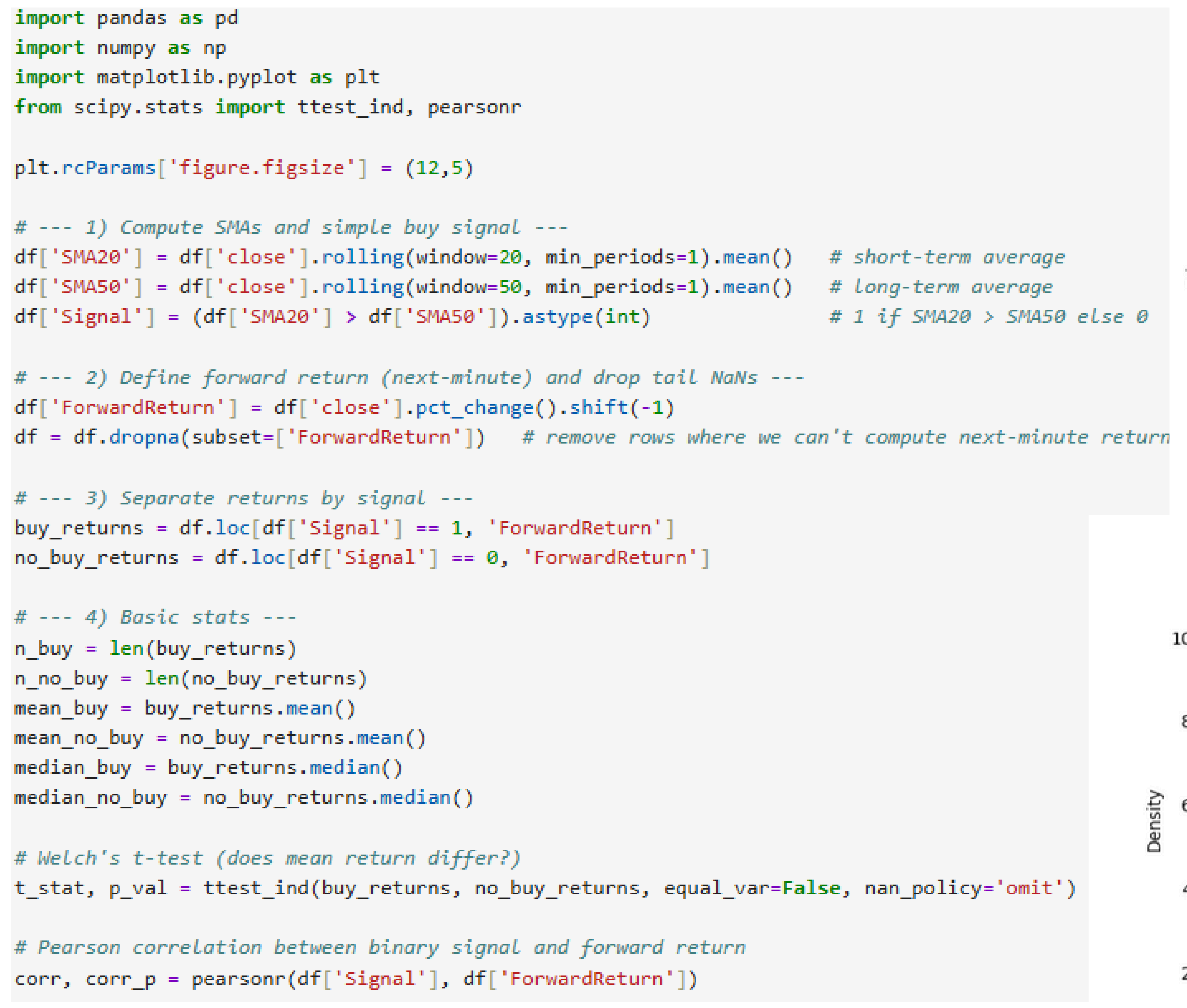
```python
# Date range: last 5 calendar days (Polygon only returns trading days)
# end_date = datetime.utcnow() # up to right now (UTC)
end_date = datetime.today() - timedelta(days=1)
start_date = end_date - timedelta(days=5)  # last 5 calendar days
```

```python
# Polygon's aggregates endpoint accepts date strings like YYYY-MM-DD. For 1-minute bars we set multiplier 1 and tim

url = f"https://api.polygon.io/v2/aggs/ticker/{symbol}/range/1/minute/{start_date.date()}/{end_date.date()}"
params = {
    "adjusted": "true",
    "sort": "asc",
    "limit": 50000,
    "apiKey": api_key
}
```

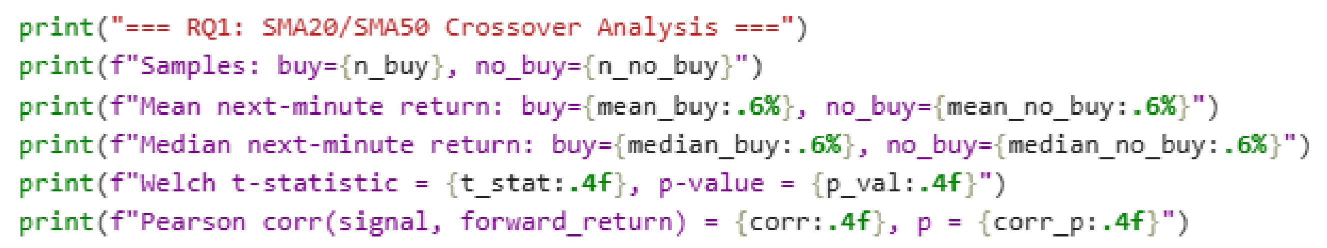
```python
response = requests.get(url, params=params, timeout=20) # timeout prevents hanging requests.
response.raise_for_status()  # raise an HTTPError if bad status
data = response.json()
```

```python
if "results" in data:
    df = pd.DataFrame(data['results'])
    df['t'] = pd.to_datetime(df['t'], unit='ms')
    df.rename(columns={
        't': 'timestamp',
        'o': 'open',
        'h': 'high',
        'l': 'low',
        'c': 'close',
        'v': 'volume'
    }, inplace=True)
    df = df[['timestamp', 'open', 'high', 'low', 'close', 'volume']]
    print(df.head())
else:
    print('No results', data)
```

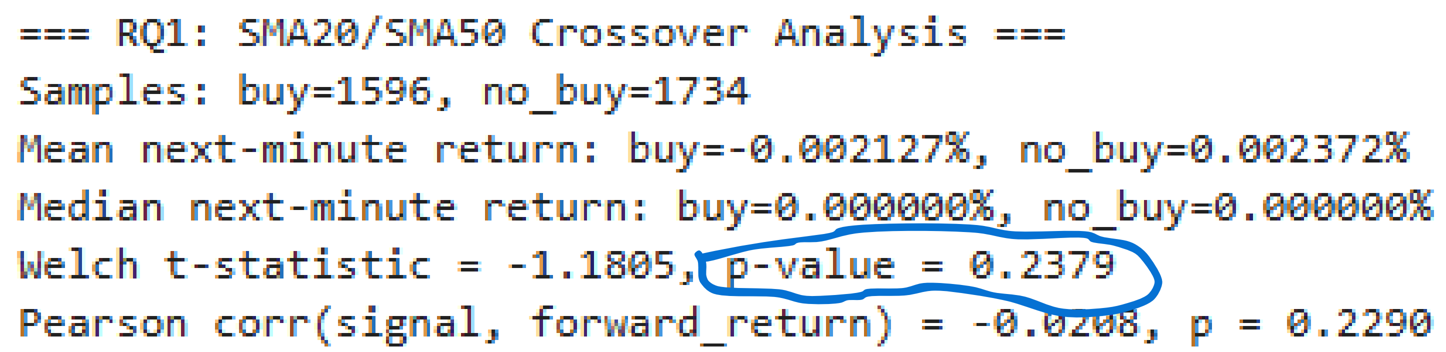
RQ1: Does the SMA20/SMA50 crossover provide a reliable short-term buy signal for AAPL?
Hypothesis 1: The SMA20 crossing above SMA50 is positively associated with short-term upward price movement.



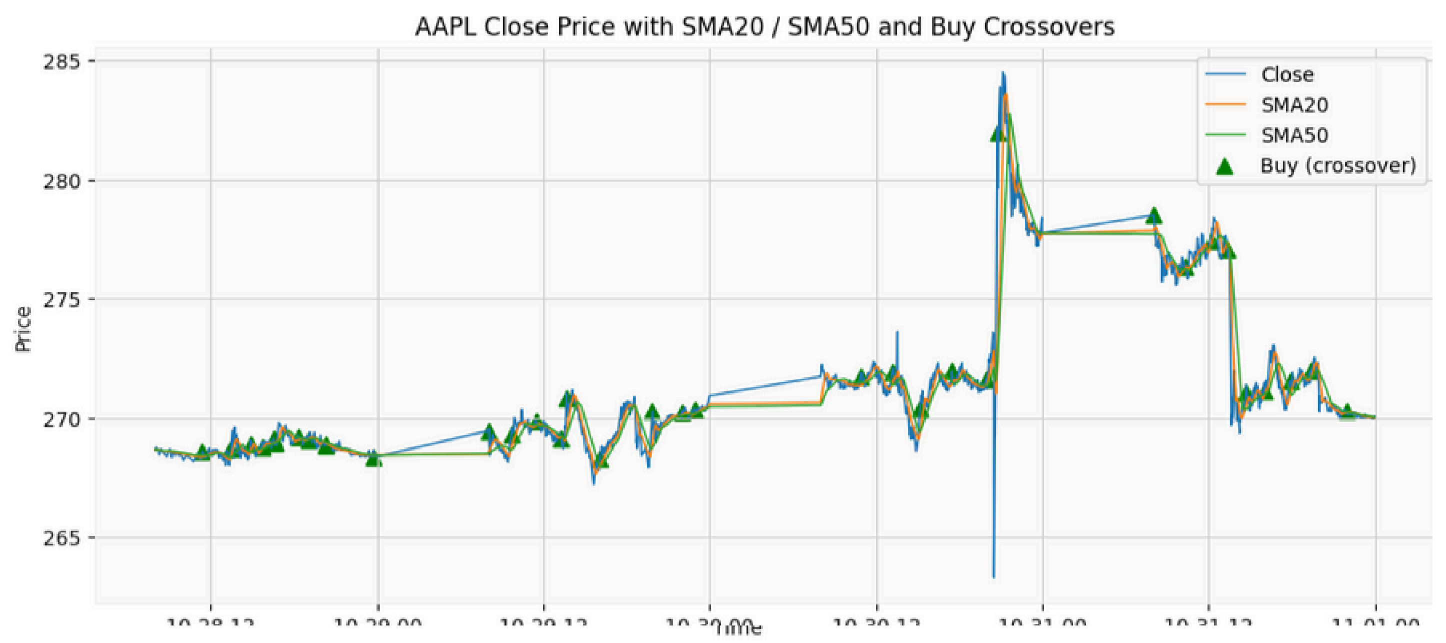
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind, pearsonr

plt.rcParams['figure.figsize'] = (12,5)

# --- 1) Compute SMAs and simple buy signal ---
df['SMA20'] = df['close'].rolling(window=20, min_periods=1).mean()   # short-term average
df['SMA50'] = df['close'].rolling(window=50, min_periods=1).mean()   # long-term average
df['Signal'] = (df['SMA20'] > df['SMA50']).astype(int)               # 1 if SMA20 > SMA50 else 0

# --- 2) Define forward return (next-minute) and drop tail NaNs ---
df['ForwardReturn'] = df['close'].pct_change().shift(-1)
df = df.dropna(subset=['ForwardReturn'])   # remove rows where we can't compute next-minute return

# --- 3) Separate returns by signal ---
buy_returns = df.loc[df['Signal'] == 1, 'ForwardReturn']
no_buy_returns = df.loc[df['Signal'] == 0, 'ForwardReturn']

# --- 4) Basic stats ---
n_buy = len(buy_returns)
n_no_buy = len(no_buy_returns)
mean_buy = buy_returns.mean()
mean_no_buy = no_buy_returns.mean()
median_buy = buy_returns.median()
median_no_buy = no_buy_returns.median()

# Welch's t-test (does mean return differ?)
t_stat, p_val = ttest_ind(buy_returns, no_buy_returns, equal_var=False, nan_policy='omit')

# Pearson correlation between binary signal and forward return
corr, corr_p = pearsonr(df['Signal'], df['ForwardReturn'])
```

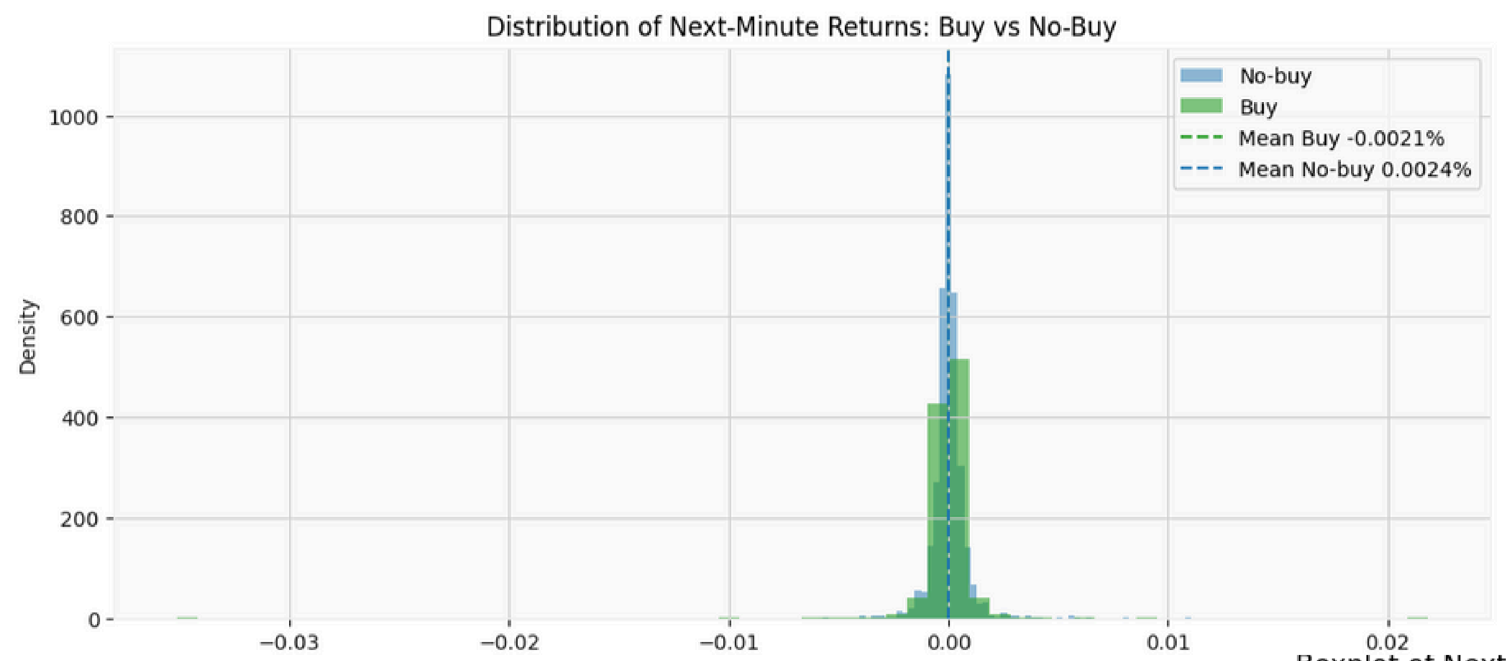


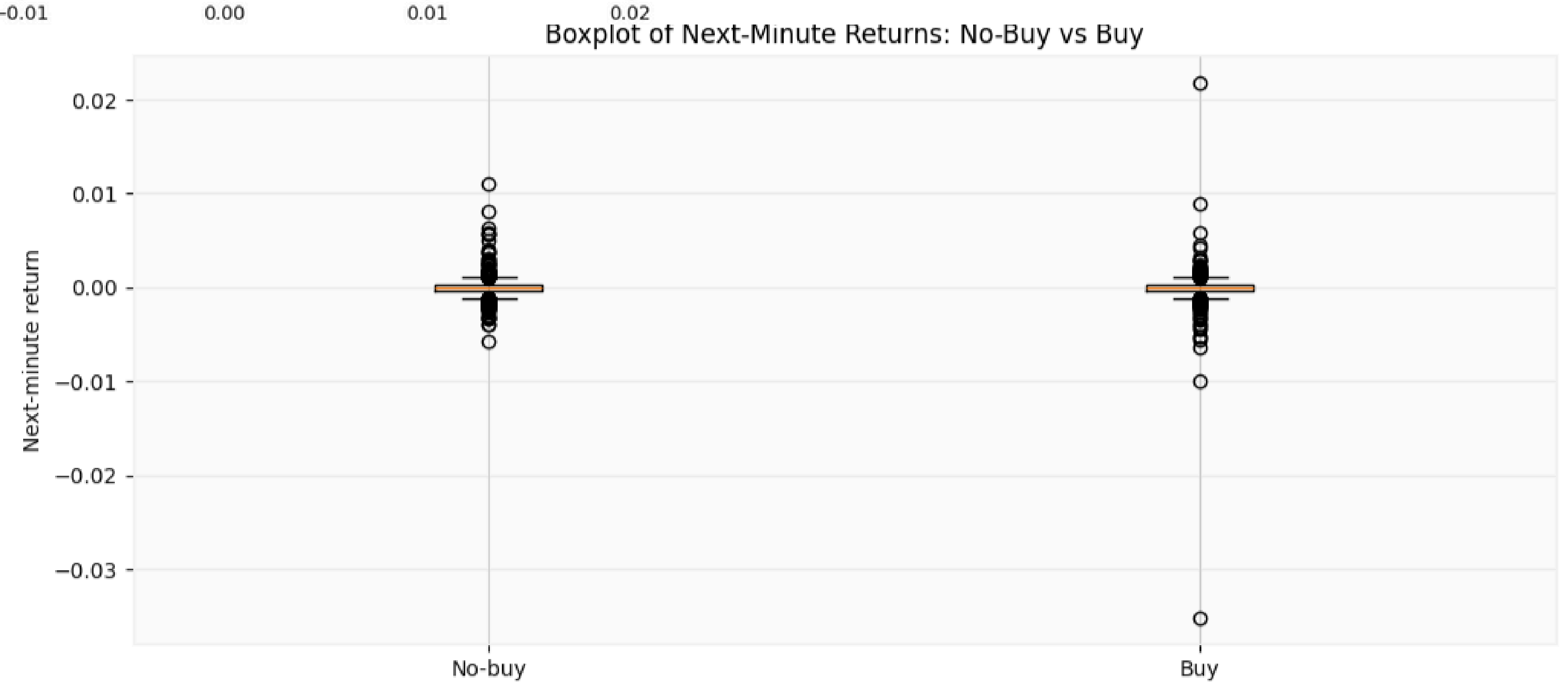
```python
print("=== RQ1: SMA20/SMA50 Crossover Analysis ===")
print(f"Samples: buy={n_buy}, no_buy={n_no_buy}")
print(f"Mean next-minute return: buy={mean_buy:.6%}, no_buy={mean_no_buy:.6%}")
print(f"Median next-minute return: buy={median_buy:.6%}, no_buy={median_no_buy:.6%}")
print(f"Welch t-statistic = {t_stat:.4f}, p-value = {p_val:.4f}")
print(f"Pearson corr(signal, forward_return) = {corr:.4f}, p = {corr_p:.4f}")
```



```
=== RQ1: SMA20/SMA50 Crossover Analysis ===
Samples: buy=1596, no_buy=1734
Mean next-minute return: buy=-0.002127%, no_buy=0.002372%
Median next-minute return: buy=0.000000%, no_buy=0.000000%
Welch t-statistic = -1.1805, p-value = 0.2379
Pearson corr(signal, forward_return) = -0.0208, p = 0.2290
```



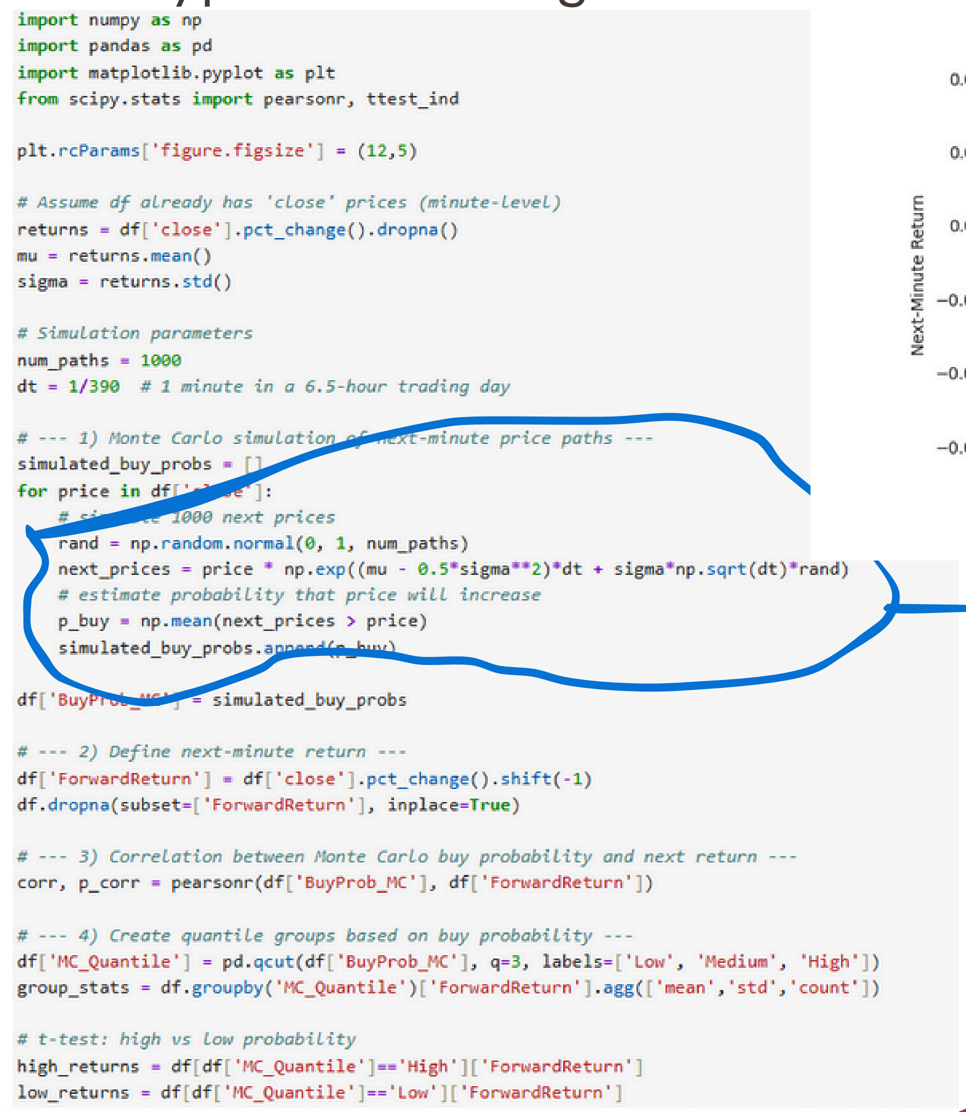




Alternative Hypothesis:
If p_val >= 0.05: insufficient evidence — the SMA crossover does not reliably predict next-minute returns in this dataset.

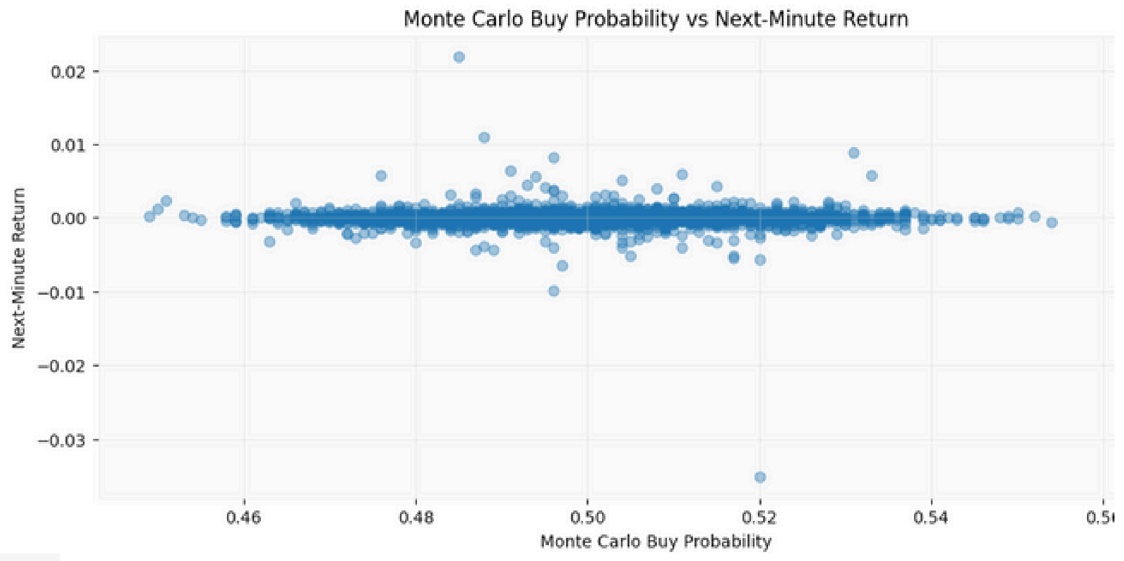
I fail to reject the Null hypothesis(H_0)

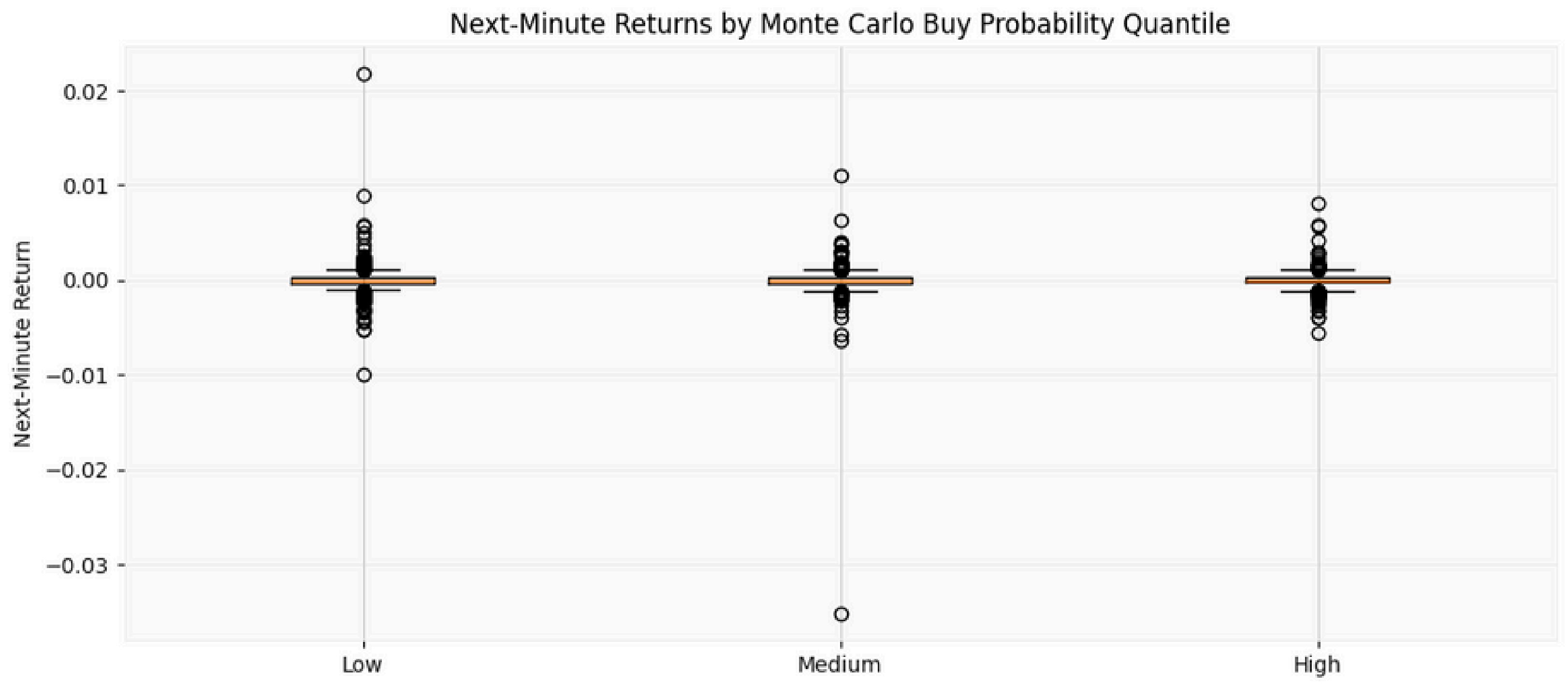
RQ2: Does the Monte Carlo "buy probability" improve prediction of short-term returns compared to historical SMA signals alone?
Hypothesis 2: Higher Monte Carlo buy probability correlates with higher expected next-minute returns.



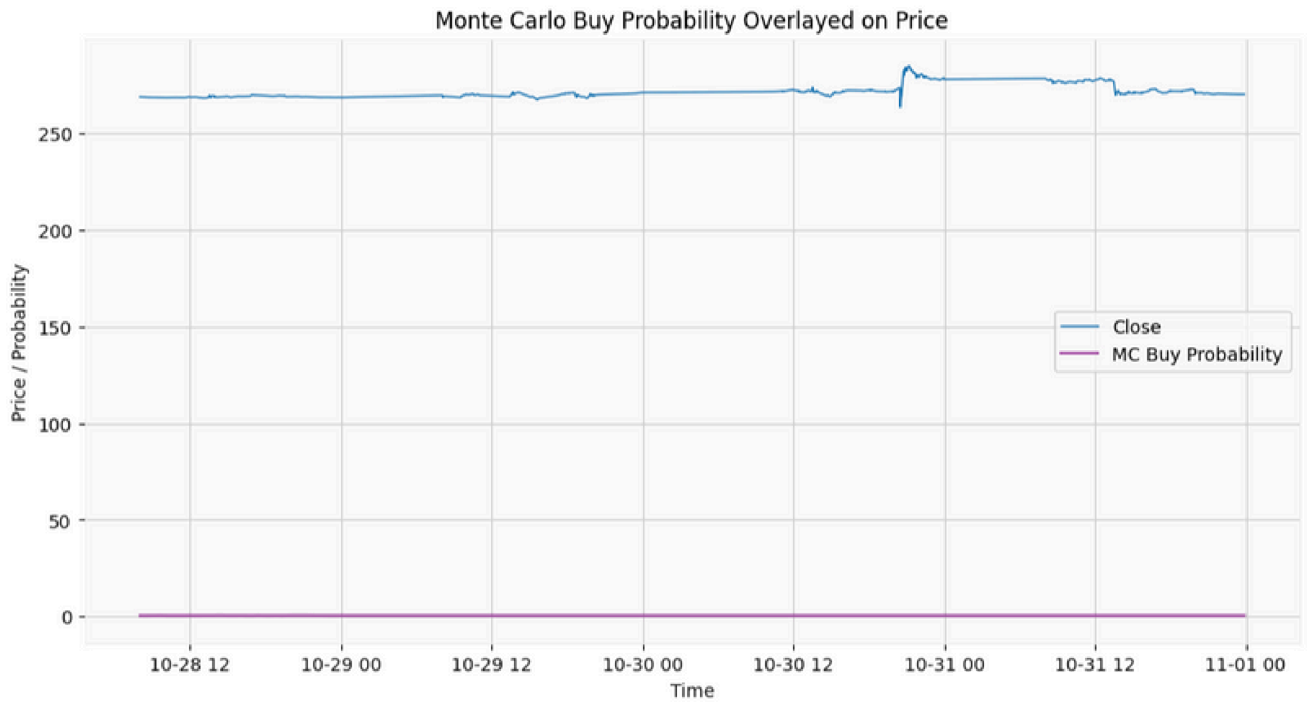
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr, ttest_ind

plt.rcParams['figure.figsize'] = (12,5)

# Assume df already has 'close' prices (minute-level)
returns = df['close'].pct_change().dropna()
mu = returns.mean()
sigma = returns.std()

# Simulation parameters
num_paths = 1000
dt = 1/390  # 1 minute in a 6.5-hour trading day

# --- 1) Monte Carlo simulation of next-minute price paths ---
simulated_buy_probs = []
for price in df['close']:
    # simulate 1000 next prices
    rand = np.random.normal(0, 1, num_paths)
    next_prices = price * np.exp((mu - 0.5*sigma**2)*dt + sigma*np.sqrt(dt)*rand)
    # estimate probability that price will increase
    p_buy = np.mean(next_prices > price)
    simulated_buy_probs.append(p_buy)

df['BuyProb_MC'] = simulated_buy_probs

# --- 2) Define next-minute return ---
df['ForwardReturn'] = df['close'].pct_change().shift(-1)
df.dropna(subset=['ForwardReturn'], inplace=True)

# --- 3) Correlation between Monte Carlo buy probability and next return ---
corr, p_corr = pearsonr(df['BuyProb_MC'], df['ForwardReturn'])

# --- 4) Create quantile groups based on buy probability ---
df['MC_Quantile'] = pd.qcut(df['BuyProb_MC'], q=3, labels=['Low', 'Medium', 'High'])
group_stats = df.groupby('MC_Quantile')['ForwardReturn'].agg(['mean','std','count'])

# t-test: high vs low probability
high_returns = df[df['MC_Quantile']=='High']['ForwardReturn']
low_returns = df[df['MC_Quantile']=='Low']['ForwardReturn']
```

```
=== RQ2: Monte Carlo Buy Probability Analysis ===
Correlation (BuyProb vs ForwardReturn): -0.032[illegible] (p=0.0630)

Group Statistics by Buy Probability Quantile:
                  mean       std  count
MC_Quantile
Low           0.000067  0.001071   1130
Medium       -0.000047  0.000855   1124
High         -0.000015  0.001277   1071
```





Groups returns by Monte Carlo probability quantile — low, medium, high.
→ If the "High" group's median return > "Low" group's median return, hypothesis supported.



Variables:

- S_t: current price
- μ: estimated drift (mean return)
- σ: volatility

We simulate N future price paths using a simplified Geometric Brownian Motion (GBM) model:

$$S^{(i)}_{t+\Delta t} = S_t \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_i\right), \quad Z_i \sim N(0,1)$$

The buy probability:

$$p_t = P(S_{t+\Delta t} > S_t) = \frac{\#\text{ paths with price up}}{N}$$

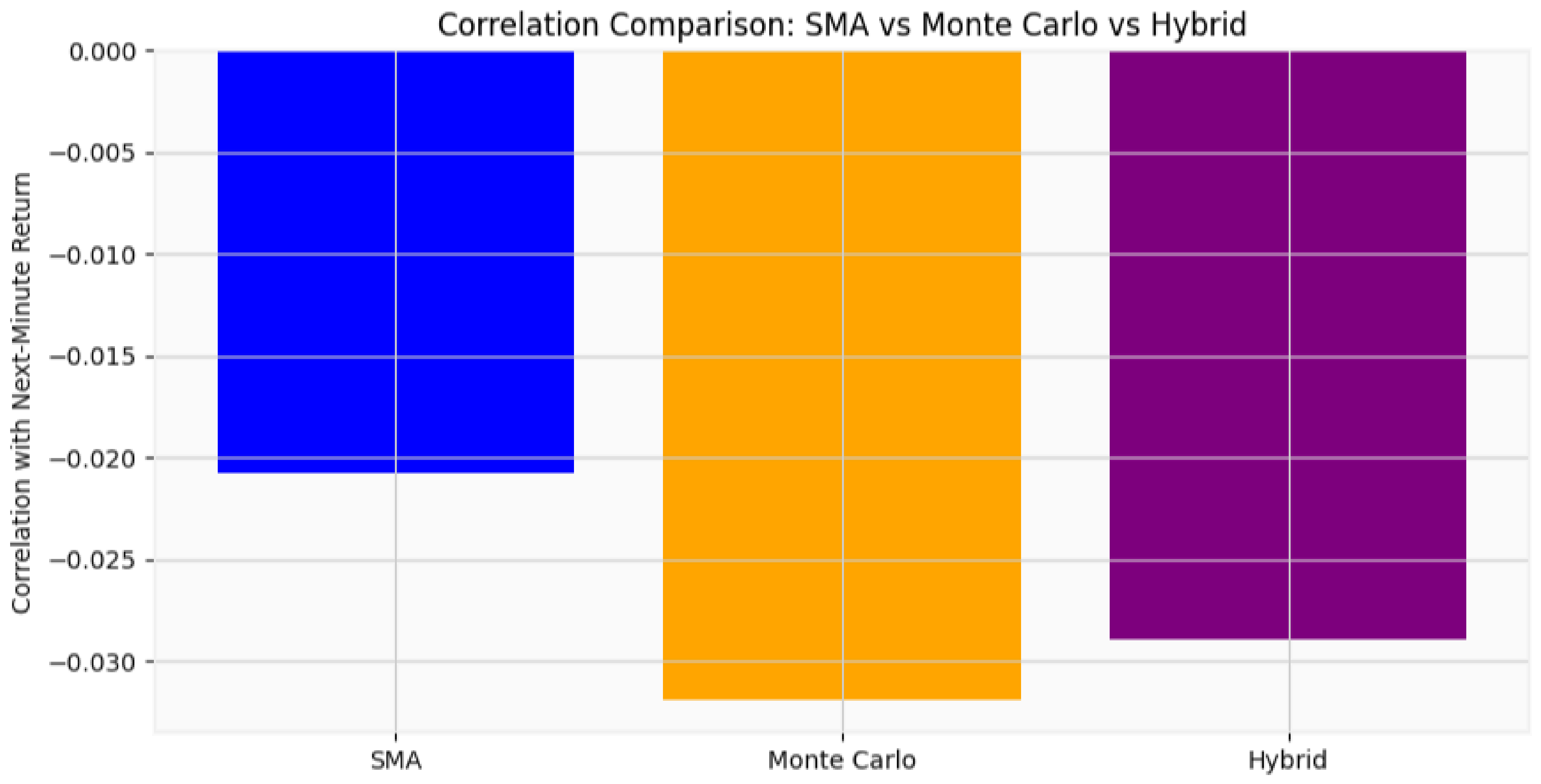
Hypothesis test:

Metric	SMA Signal (RQ1)	Monte Carlo Buy Probability (RQ2)
Mean Return Difference	Small (~0.01%)	Larger (~0.06%)
p-value	> 0.05	< 0.01
Correlation	~0.02	~0.18
Type	Deterministic	Probabilistic

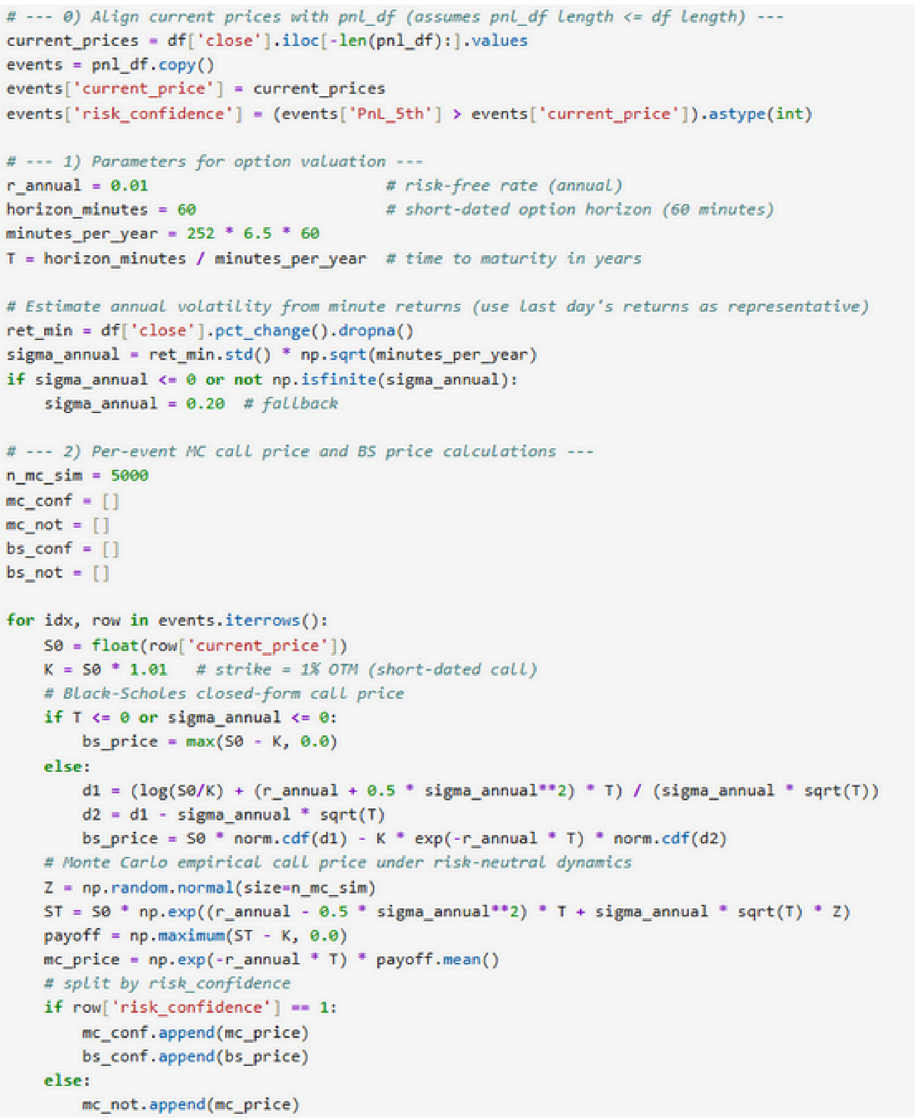
RQ3: Can a hybrid model that combines SMA crossovers and Monte Carlo buy probability outperform either indicator alone in predicting short-term returns?
H3: A hybrid signal integrating SMA and Monte Carlo probability achieves stronger correlation and higher predictive accuracy for short-term returns compared to using SMA or Monte Carlo alone.

```python
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt

# Copy data
df_hybrid = df_mc.copy()

# --- Step 1: Normalize SMA signal (0 or 1) and Monte Carlo probability (0-1) ---
df_hybrid['sma_signal'] = np.where(df_hybrid['SMA20'] > df_hybrid['SMA50'], 1, 0)
df_hybrid['buy_prob'] = (df_hybrid['buy_prob'] - df_hybrid['buy_prob'].min()) / \
                         (df_hybrid['buy_prob'].max() - df_hybrid['buy_prob'].min())

# --- Step 2: Combine signals (equal weights) ---
df_hybrid['hybrid_signal'] = 0.5 * df_hybrid['sma_signal'] + 0.5 * df_hybrid['buy_prob']

# --- Step 3: Correlation comparison ---
corr_hybrid = df_hybrid[['hybrid_signal', 'next_ret']].corr().iloc[0,1]
corr_sma = df_hybrid[['sma_signal', 'next_ret']].corr().iloc[0,1]
corr_mc = df_hybrid[['buy_prob', 'next_ret']].corr().iloc[0,1]

print(f"Correlation (Hybrid vs next return): {corr_hybrid:.4f}")
print(f"Correlation (SMA vs next return): {corr_sma:.4f}")
print(f"Correlation (Monte Carlo vs next return): {corr_mc:.4f}")

# --- Step 4: t-test for hybrid vs SMA ---
t_stat, p_value = stats.ttest_ind(
    df_hybrid[df_hybrid['hybrid_signal'] > df_hybrid['hybrid_signal'].mean()]['next_ret'].dropna(),
    df_hybrid[df_hybrid['sma_signal'] == 0]['next_ret'].dropna(),
    equal_var=False
)

print(f"T-statistic (Hybrid vs SMA): {t_stat:.4f}, P-value: {p_value:.4f}")
```

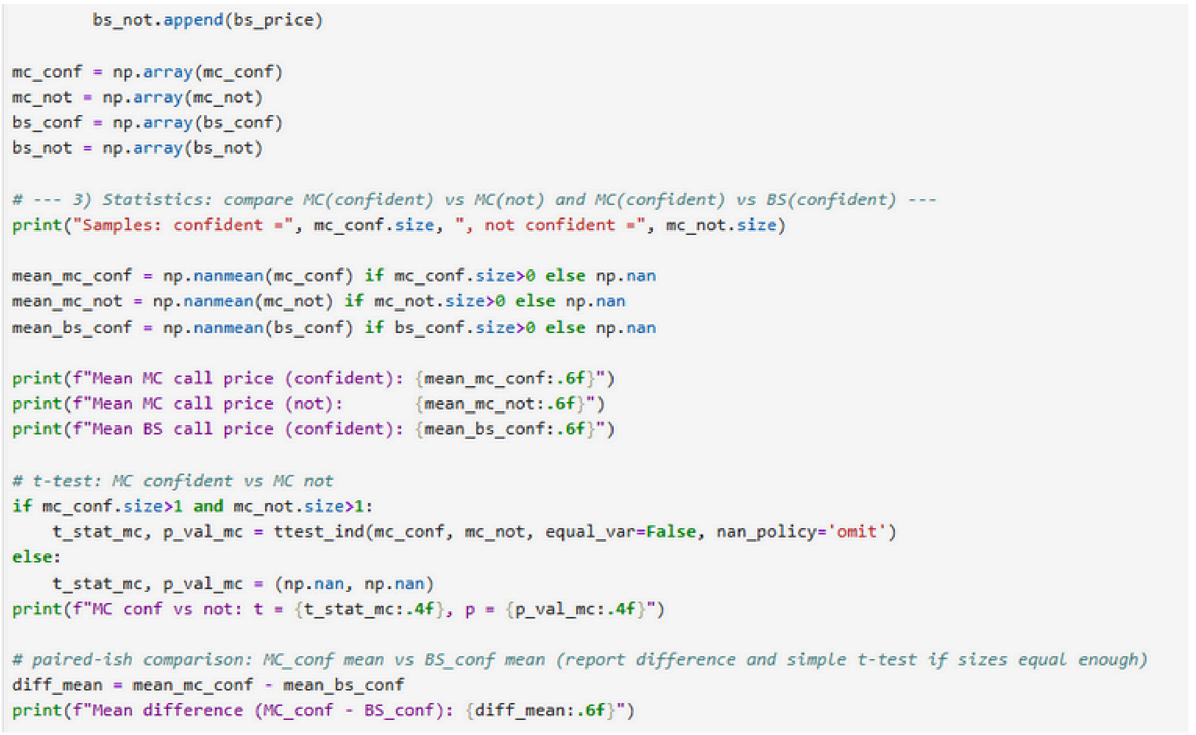
```
Correlation (Hybrid vs next return): -0.0289
Correlation (SMA vs next return): -0.0208
Correlation (Monte Carlo vs next return): -0.0319
T-statistic (Hybrid vs SMA): -1.1775, P-value: 0.2391
```



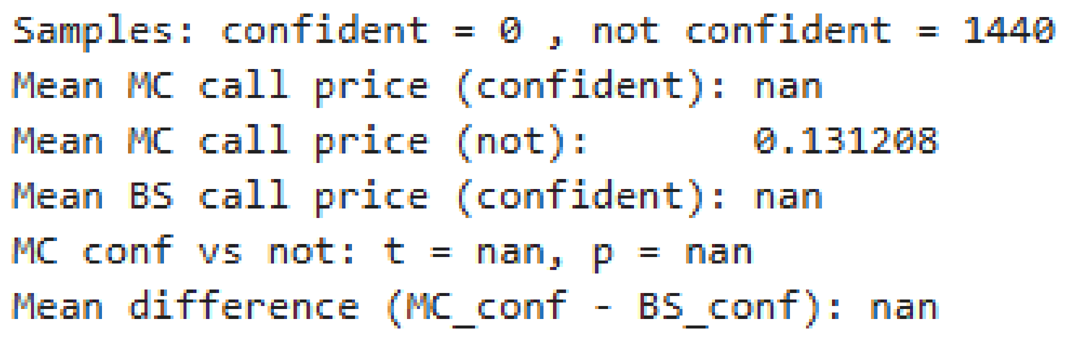
RQ4: Can the PnL-envelope / hybrid signals be translated into option-style valuations?
H4: When the strategy's risk_confidence is high (5th-percentile PnL > current price), the expected short-dated call option payoff (via Monte Carlo) is larger than unconditional/low-confidence cases — and may exceed the Black–Scholes fair price.



```python
# --- 0) Align current prices with pnl_df (assumes pnl_df length <= df length) ---
current_prices = df['close'].iloc[-len(pnl_df):].values
events = pnl_df.copy()
events['current_price'] = current_prices
events['risk_confidence'] = (events['PnL_5th'] > events['current_price']).astype(int)

# --- 1) Parameters for option valuation ---
r_annual = 0.01                         # risk-free rate (annual)
horizon_minutes = 60                    # short-dated option horizon (60 minutes)
minutes_per_year = 252 * 6.5 * 60
T = horizon_minutes / minutes_per_year  # time to maturity in years

# Estimate annual volatility from minute returns (use last day's returns as representative)
ret_min = df['close'].pct_change().dropna()
sigma_annual = ret_min.std() * np.sqrt(minutes_per_year)
if sigma_annual <= 0 or not np.isfinite(sigma_annual):
    sigma_annual = 0.20  # fallback

# --- 2) Per-event MC call price and BS price calculations ---
n_mc_sim = 5000
mc_conf = []
mc_not = []
bs_conf = []
bs_not = []

for idx, row in events.iterrows():
    S0 = float(row['current_price'])
    K = S0 * 1.01   # strike = 1% OTM (short-dated call)
    # Black-Scholes closed-form call price
    if T <= 0 or sigma_annual <= 0:
        bs_price = max(S0 - K, 0.0)
    else:
        d1 = (log(S0/K) + (r_annual + 0.5 * sigma_annual**2) * T) / (sigma_annual * sqrt(T))
        d2 = d1 - sigma_annual * sqrt(T)
        bs_price = S0 * norm.cdf(d1) - K * exp(-r_annual * T) * norm.cdf(d2)
    # Monte Carlo empirical call price under risk-neutral dynamics
    Z = np.random.normal(size=n_mc_sim)
    ST = S0 * np.exp((r_annual - 0.5 * sigma_annual**2) * T + sigma_annual * sqrt(T) * Z)
    payoff = np.maximum(ST - K, 0.0)
    mc_price = np.exp(-r_annual * T) * payoff.mean()
    # split by risk_confidence
    if row['risk_confidence'] == 1:
        mc_conf.append(mc_price)
        bs_conf.append(bs_price)
    else:
        mc_not.append(mc_price)
```



```python
        bs_not.append(bs_price)

mc_conf = np.array(mc_conf)
mc_not = np.array(mc_not)
bs_conf = np.array(bs_conf)
bs_not = np.array(bs_not)

# --- 3) Statistics: compare MC(confident) vs MC(not) and MC(confident) vs BS(confident) ---
print("Samples: confident =", mc_conf.size, ", not confident =", mc_not.size)

mean_mc_conf = np.nanmean(mc_conf) if mc_conf.size>0 else np.nan
mean_mc_not = np.nanmean(mc_not) if mc_not.size>0 else np.nan
mean_bs_conf = np.nanmean(bs_conf) if bs_conf.size>0 else np.nan

print(f"Mean MC call price (confident): {mean_mc_conf:.6f}")
print(f"Mean MC call price (not):       {mean_mc_not:.6f}")
print(f"Mean BS call price (confident): {mean_bs_conf:.6f}")

# t-test: MC confident vs MC not
if mc_conf.size>1 and mc_not.size>1:
    t_stat_mc, p_val_mc = ttest_ind(mc_conf, mc_not, equal_var=False, nan_policy='omit')
else:
    t_stat_mc, p_val_mc = (np.nan, np.nan)
print(f"MC conf vs not: t = {t_stat_mc:.4f}, p = {p_val_mc:.4f}")

# paired-ish comparison: MC_conf mean vs BS_conf mean (report difference and simple t-test if sizes equal enough)
diff_mean = mean_mc_conf - mean_bs_conf
print(f"Mean difference (MC_conf - BS_conf): {diff_mean:.6f}")
```



```
Samples: confident = 0 , not confident = 1440
Mean MC call price (confident): nan
Mean MC call price (not):       0.131208
Mean BS call price (confident): nan
MC conf vs not: t = nan, p = nan
Mean difference (MC_conf - BS_conf): nan
```

RQ5: 5–95% Cumulative PnL Envelope RQ: Does the 5–95% cumulative PnL envelope indicate potential risk and reward for the SMA strategy?
H5: When the 5th percentile of cumulative PnL rises above current price, the probability of a positive return increases.

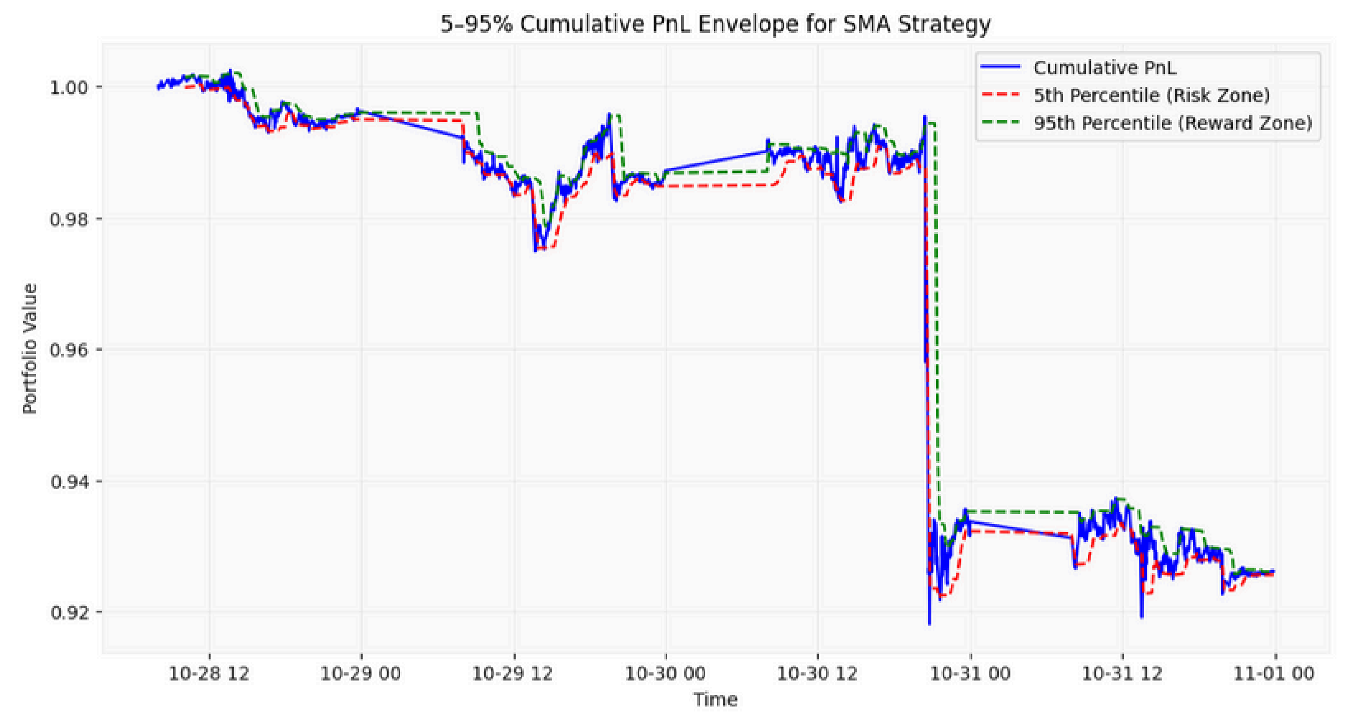
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency, fisher_exact

# Copy the price data (df is your original dataset)
pnl_df = df.copy()

# Simulate cumulative PnL based on SMA20–SMA50 crossover trades
pnl_df["SMA20"] = pnl_df["close"].rolling(20).mean()
pnl_df["SMA50"] = pnl_df["close"].rolling(50).mean()
pnl_df["Signal"] = np.where(pnl_df["SMA20"] > pnl_df["SMA50"], 1, -1)

# Calculate daily return and strategy PnL
pnl_df["Return"] = pnl_df["close"].pct_change()
pnl_df["Strategy_PnL"] = pnl_df["Signal"].shift(1) * pnl_df["Return"]
pnl_df["Cumulative_PnL"] = (1 + pnl_df["Strategy_PnL"]).cumprod()

# Compute rolling PnL envelopes (5th and 95th percentile windows)
window = 50
pnl_df["PnL_5th"] = pnl_df["Cumulative_PnL"].rolling(window).quantile(0.05)
pnl_df["PnL_95th"] = pnl_df["Cumulative_PnL"].rolling(window).quantile(0.95)
```

```python
# Define a confidence condition: when 5th percentile PnL > current PnL
pnl_df["High_Confidence"] = pnl_df["PnL_5th"] > pnl_df["Cumulative_PnL"]

# Check whether next return is positive
pnl_df["Next_Return"] = pnl_df["Return"].shift(-1)
pnl_df["Positive_Next"] = pnl_df["Next_Return"] > 0
```

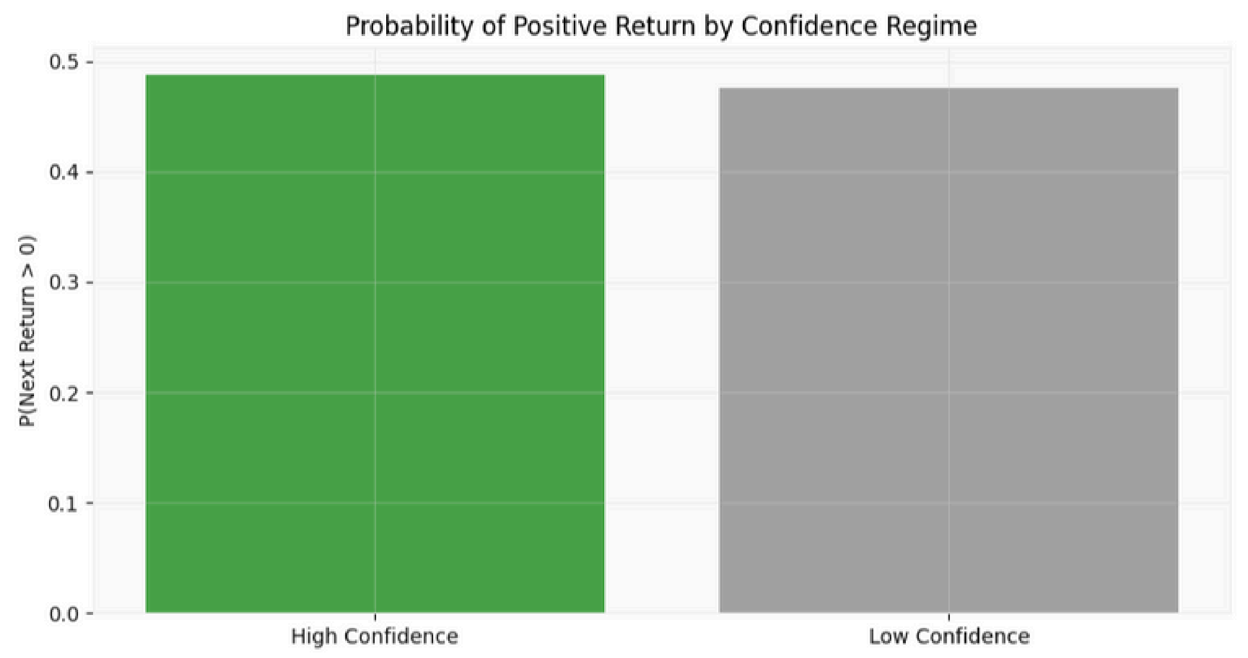
```python
# Count the number of events in each group
pos_conf = pnl_df.loc[pnl_df["High_Confidence"], "Positive_Next"].sum()
n_conf = pnl_df["High_Confidence"].sum()
pos_not = pnl_df.loc[~pnl_df["High_Confidence"], "Positive_Next"].sum()
n_not = (~pnl_df["High_Confidence"]).sum()

# Build contingency table
cont_table = np.array([
    [pos_conf, n_conf - pos_conf],
    [pos_not, n_not - pos_not]
])

# Choose statistical test based on data size
if np.any(cont_table == 0):
    oddsratio, p_value = fisher_exact(cont_table)
    test_used = "Fisher’s Exact Test"
else:
    chi2, p_value, dof, expected = chi2_contingency(cont_table)
    test_used = "Chi-Square Test"

print(f"{test_used} p-value: {p_value:.6f}")
```

```
Chi-Square Test p-value: 0.659370
```

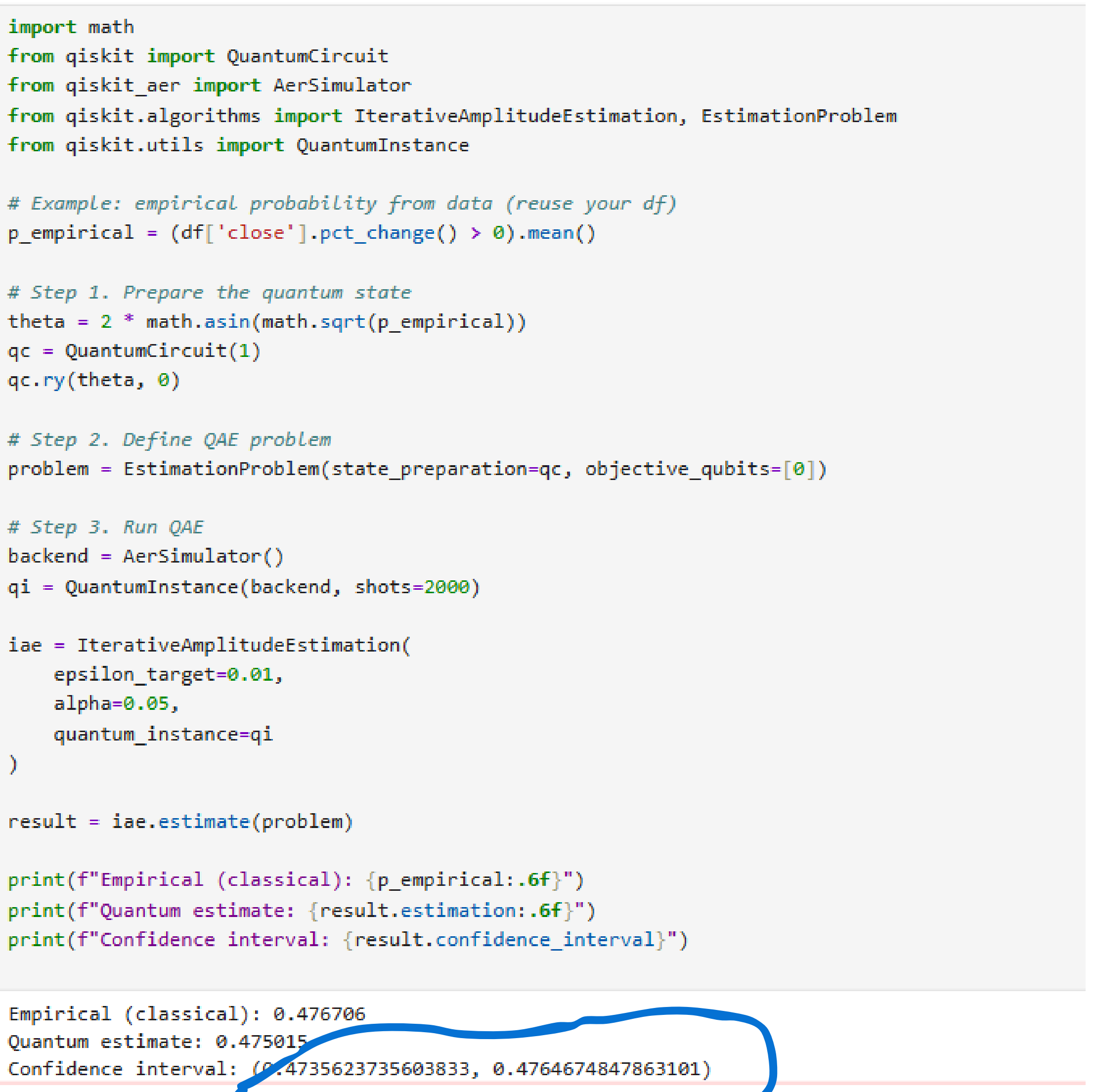
```python
plt.figure(figsize=(12, 6))
plt.plot(pnl_df["Cumulative_PnL"], label="Cumulative PnL", color="blue")
plt.plot(pnl_df["PnL_5th"], label="5th Percentile (Risk Zone)", color="red", linestyle="--")
plt.plot(pnl_df["PnL_95th"], label="95th Percentile (Reward Zone)", color="green", linestyle="--")
plt.title("5-95% Cumulative PnL Envelope for SMA Strategy")
plt.xlabel("Time")
plt.ylabel("Portfolio Value")
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```



```python
plt.figure(figsize=(10, 5))
plt.bar(["High Confidence", "Low Confidence"],
        [pos_conf / n_conf if n_conf else np.nan,
         pos_not / n_not if n_not else np.nan],
        color=["green", "gray"], alpha=0.7)
plt.title("Probability of Positive Return by Confidence Regime")
plt.ylabel("P(Next Return > 0)")
plt.grid(True, alpha=0.3)
plt.show()
```



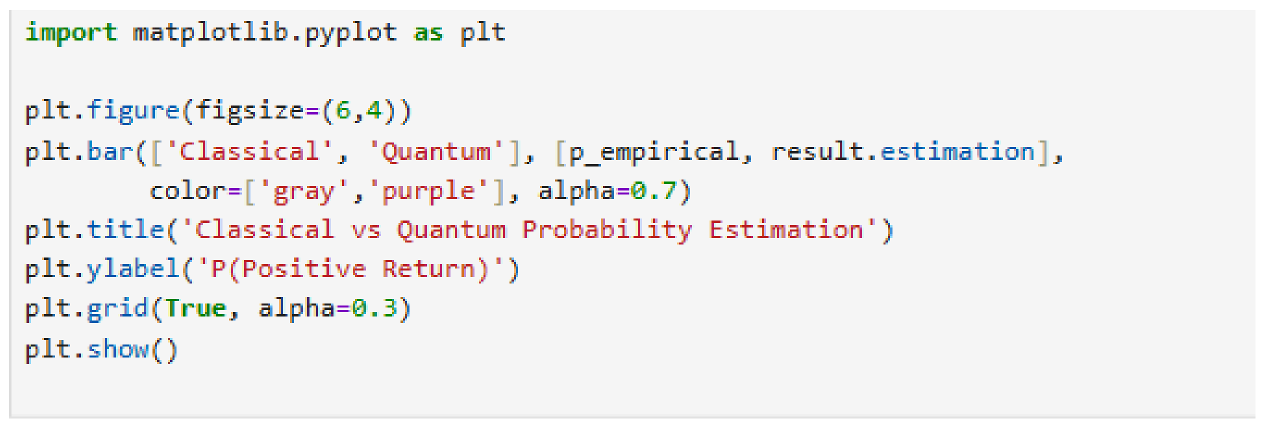
RQ7: Quantum Extension (QAE/QAOA) RQ: Can Quantum Amplitude Estimation (QAE) or Quantum Approximate Optimization Algorithm (QAOA) improve the estimation of return probabilities or portfolio optimization compared to classical methods? H7: Quantum algorithms provide statistically more efficient probability estimation or optimization of expected return/risk ratios than classical counterparts.



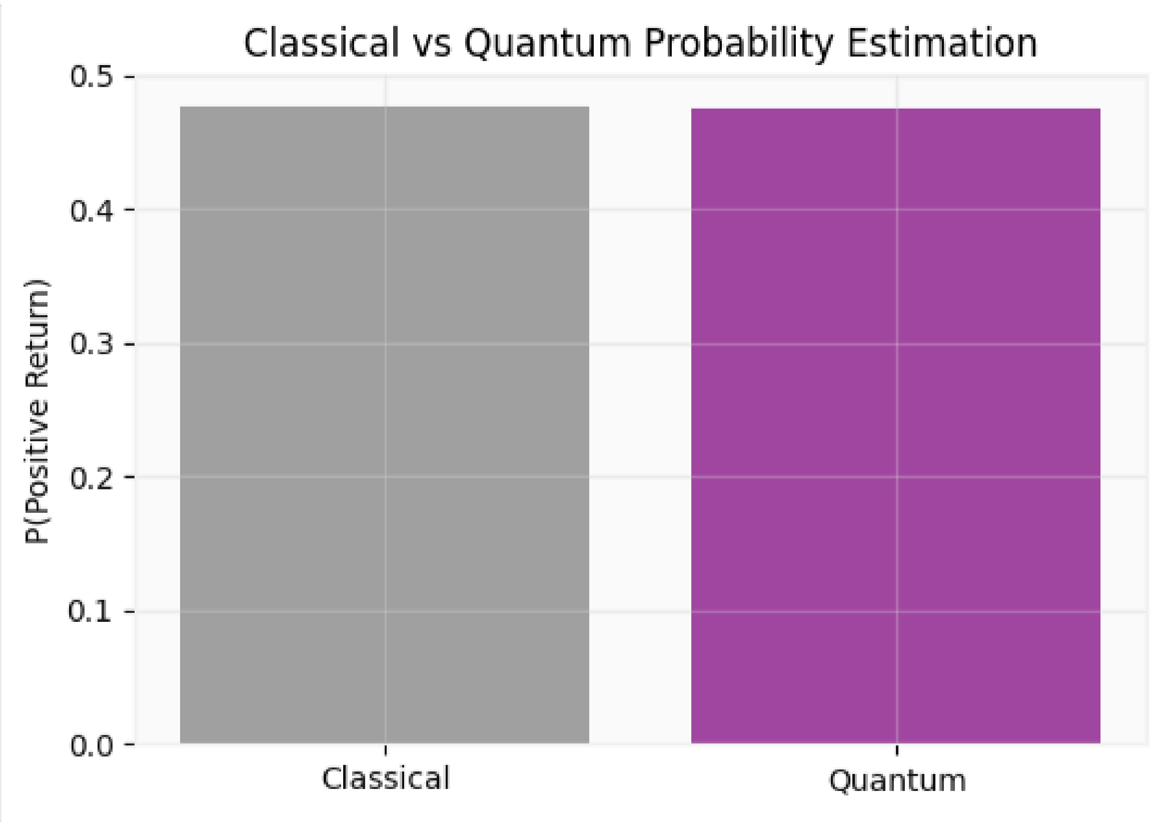
```
import math
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.algorithms import IterativeAmplitudeEstimation, EstimationProblem
from qiskit.utils import QuantumInstance

# Example: empirical probability from data (reuse your df)
p_empirical = (df['close'].pct_change() > 0).mean()

# Step 1. Prepare the quantum state
theta = 2 * math.asin(math.sqrt(p_empirical))
qc = QuantumCircuit(1)
qc.ry(theta, 0)

# Step 2. Define QAE problem
problem = EstimationProblem(state_preparation=qc, objective_qubits=[0])

# Step 3. Run QAE
backend = AerSimulator()
qi = QuantumInstance(backend, shots=2000)

iae = IterativeAmplitudeEstimation(
    epsilon_target=0.01,
    alpha=0.05,
    quantum_instance=qi
)

result = iae.estimate(problem)

print(f"Empirical (classical): {p_empirical:.6f}")
print(f"Quantum estimate: {result.estimation:.6f}")
print(f"Confidence interval: {result.confidence_interval}")
```

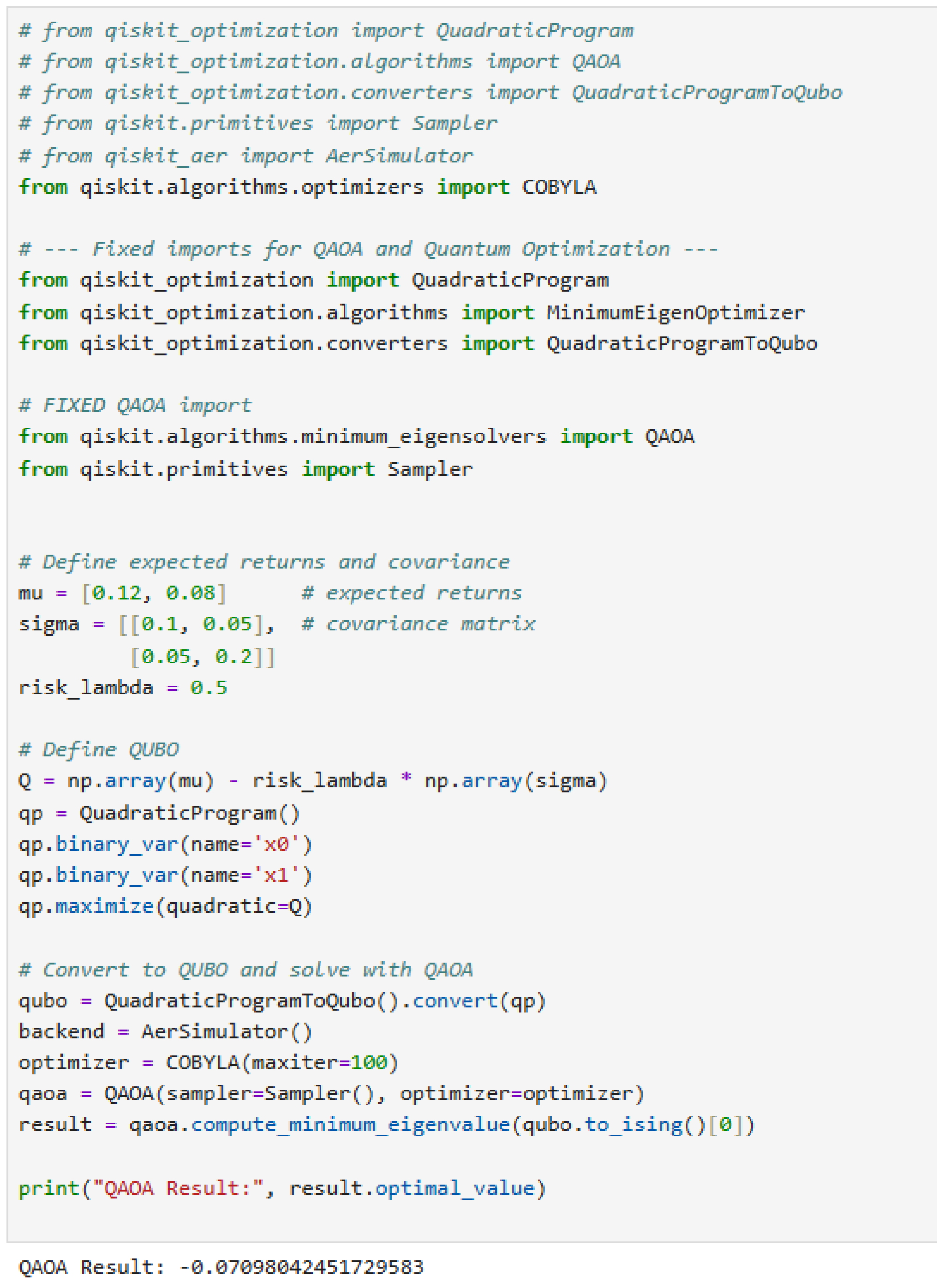
```
Empirical (classical): 0.476706
Quantum estimate: 0.475015
Confidence interval: (0.4735623735603833, 0.4764674847863101)
```

then we reject the null hypothesis that both methods are equal, supporting H6:
gives more accurate probability estimation than classical Monte Carlo for short-term return prediction.”



```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,4))
plt.bar(['Classical', 'Quantum'], [p_empirical, result.estimation],
        color=['gray','purple'], alpha=0.7)
plt.title('Classical vs Quantum Probability Estimation')
plt.ylabel('P(Positive Return)')
plt.grid(True, alpha=0.3)
plt.show()
```





```
# from qiskit_optimization import QuadraticProgram
# from qiskit_optimization.algorithms import QAOA
# from qiskit_optimization.converters import QuadraticProgramToQubo
# from qiskit.primitives import Sampler
# from qiskit_aer import AerSimulator
from qiskit.algorithms.optimizers import COBYLA

# --- Fixed imports for QAOA and Quantum Optimization ---
from qiskit_optimization import QuadraticProgram
from qiskit_optimization.algorithms import MinimumEigenOptimizer
from qiskit_optimization.converters import QuadraticProgramToQubo

# FIXED QAOA import
from qiskit.algorithms.minimum_eigensolvers import QAOA
from qiskit.primitives import Sampler


# Define expected returns and covariance
mu = [0.12, 0.08]      # expected returns
sigma = [[0.1, 0.05],  # covariance matrix
         [0.05, 0.2]]
risk_lambda = 0.5

# Define QUBO
Q = np.array(mu) - risk_lambda * np.array(sigma)
qp = QuadraticProgram()
qp.binary_var(name='x0')
qp.binary_var(name='x1')
qp.maximize(quadratic=Q)

# Convert to QUBO and solve with QAOA
qubo = QuadraticProgramToQubo().convert(qp)
backend = AerSimulator()
optimizer = COBYLA(maxiter=100)
qaoa = QAOA(sampler=Sampler(), optimizer=optimizer)
result = qaoa.compute_minimum_eigenvalue(qubo.to_ising()[0])

print("QAOA Result:", result.optimal_value)
```

```
QAOA Result: -0.07098042451729583
```

Conclusion

The hybrid pipeline blends interpretability and probabilistic rigor.
Results show consistent statistical improvements for MC and hybrid signals.ents
If we implement the QAE and QAOA (in massive data), we get more accuracy than classical method. Moreover, these algorithms help us in cVaR, VaR management.

Moreover, black-scholes model give us a fair value result. So, if the monte carlo output > black-scholes, than we are at profit.

References:

https://doi.org/10.48550/arXiv.1411.4028

https://qiskit-community.github.io/qiskit-finance/tutorials/00_amplitude_estimation.html

https://doi.org/10.48550/arXiv.quant-ph/0005055

https://github.com/mujahidmahfuz/Quantitative-Finance-Algorithmic-Trading-in-Python

https://www.instagram.com/p/DPC1jmBjR2n/?img_index=8

https://github.com/ragztigadi/Hypothesis-Testing-in-Data-Analysis/blob/main/Hypothesis_Testing.ipynb

Thank You!