Stock_Analysis_RJ-v2-TechRiskPort_Opt

October 11, 2025

[1]:
```
# ============================================================
#  Multi-Agent Financial Analysis System (Extended Agents)
#  - Adds TechnicalAgent, RiskAgent, PortfolioAgent, OptimizerAgent
#  - Keeps our existing agents intact and working
# ============================================================

# --- 1. Imports and Setup ----------------------------------
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta, timezone
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from textblob import TextBlob
import requests, json, re
from typing import Dict, Any, List, Optional

# Download tokenizer resources once per runtime (safe to call repeatedly)
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

# Helper - UTC timestamp (timezone-aware)
def utc_now_iso():
    """Return current UTC time in ISO format (no deprecation)."""
    return datetime.now(timezone.utc).isoformat()

# --- 2. Existing Agents (unchanged semantics, with small robustness fixes) ---

class FinancialsAgent:
    """Fetches key company metrics and plots price trends."""
    def act(self, ticker: str) -> Dict[str, Any]:
        t_obj = yf.Ticker(ticker)
        info = t_obj.info if hasattr(t_obj, "info") else {}
        # Get last 1 year of daily data
```

```
        data = yf.download(ticker, period="1y", auto_adjust=True,␣
↪progress=False)
        if data.empty:
            print(f"[FinancialsAgent] Warning: No price data for {ticker}")
            return {"type": "financials", "latest_price": None, "marketCap":␣
↪info.get("marketCap")}
        # Plot price
        plt.figure(figsize=(10,4))
        plt.plot(data.index, data['Close'])
        plt.title(f"{ticker} Close Price")
        plt.xlabel("Date")
        plt.ylabel("Close")
        plt.grid(True)
        plt.show()
        return {
            "type": "financials",
            "latest_price": float(data['Close'].iloc[-1]),
            "marketCap": info.get("marketCap"),
            "trailingPE": info.get("trailingPE"),
            "forwardPE": info.get("forwardPE"),
            "dividendYield": info.get("dividendYield"),
            "beta": info.get("beta"),
            "price_df": data  # include df for downstream agents
        }

class NewsAgent:
    """Fetches recent news headlines and performs sentiment using TextBlob."""
    def act(self, ticker: str) -> Dict[str, Any]:
        # Using Google News RSS (no API key required)
        url = f"https://news.google.com/rss/search?q={ticker}+stock"
        try:
            resp = requests.get(url, timeout=10)
            titles = re.findall(r"<title>(.*?)</title>", resp.text)
        except Exception as e:
            print(f"[NewsAgent] error fetching news for {ticker}: {e}")
            titles = []
        processed = []
        # Skip the RSS feed header titles[0..1] and limit to five items if␣
↪available
        for t in titles[2:7]:
            sent = TextBlob(t).sentiment.polarity
            cat = "earnings" if "earn" in t.lower() else "other"
            processed.append({"headline": t, "sentiment": round(sent,4),␣
↪"category": cat})
        return {"type": "news", "data": processed}

# --- 3. New Agent: TechnicalAgent ----------------------------------
```

```python
class TechnicalAgent:
    """Calculates basic technical indicators: SMA (20,50), EMA(20), RSI(14)."""
    def rsi(self, series: pd.Series, period: int = 14) -> pd.Series:
        delta = series.diff()
        up = delta.clip(lower=0)
        down = -1 * delta.clip(upper=0)
        ma_up = up.ewm(com=period-1, adjust=False).mean()
        ma_down = down.ewm(com=period-1, adjust=False).mean()
        rs = ma_up / ma_down
        return 100 - (100 / (1 + rs))

    def act(self, price_df: pd.DataFrame) -> Dict[str, Any]:
        if price_df is None or price_df.empty:
            return {"type": "technical", "indicators": {}}
        df = price_df.copy()
        df['SMA20'] = df['Close'].rolling(window=20).mean()
        df['SMA50'] = df['Close'].rolling(window=50).mean()
        df['EMA20'] = df['Close'].ewm(span=20, adjust=False).mean()
        df['RSI14'] = self.rsi(df['Close'], period=14)
        # Latest indicator snapshot
        last = df.iloc[-1].to_dict()
        indicators = {
            "SMA20": last.get('SMA20'),
            "SMA50": last.get('SMA50'),
            "EMA20": last.get('EMA20'),
            "RSI14": last.get('RSI14'),
            "close": last.get('Close')
        }

        print(f"df['SMA20'] >>> {df['SMA20']}")


        # Plot Close with SMA overlays
        plt.figure(figsize=(10,4))
        plt.plot(df.index, df['Close'], label='Close')
        plt.plot(df.index, df['SMA20'], label='SMA20', alpha=0.8)
        plt.plot(df.index, df['SMA50'], label='SMA50', alpha=0.8)
        plt.title("Price with SMA20 & SMA50")
        plt.legend()
        plt.grid(True)
        plt.show()
        return {"type": "technical", "indicators": indicators, "df": df}

# --- 4. New Agent: RiskAgent ----------------------------------
class RiskAgent:
    """Computes risk metrics: volatility (annualized), max drawdown, simple VaR.
    """
```

```
    def act(self, price_df: pd.DataFrame) -> Dict[str, Any]:
        if price_df is None or price_df.empty:
            return {"type": "risk", "metrics": {}}
        df = price_df.copy()
        df['ret'] = df['Close'].pct_change().fillna(0)
        vol_annual = df['ret'].std() * np.sqrt(252)  # annualized volatility
        # max drawdown
        cum = (1 + df['ret']).cumprod()
        running_max = cum.cummax()
        drawdown = (cum - running_max) / running_max
        max_dd = drawdown.min()
        # 95% historical VaR (simple)
        var95 = df['ret'].quantile(0.05)
        metrics = {
            "vol_annual": float(vol_annual),
            "max_drawdown": float(max_dd),
            "var95": float(var95)
        }
        # Plot rolling volatility (30d)
        df['vol30'] = df['ret'].rolling(window=30).std() * np.sqrt(252)
        plt.figure(figsize=(10,3))
        plt.plot(df.index, df['vol30'])
        plt.title("30-day Rolling Volatility (annualized)")
        plt.grid(True)
        plt.show()
        return {"type": "risk", "metrics": metrics, "df": df}

# --- 5. PortfolioAgent ----------------------------------------
class PortfolioAgent:
    """Simple portfolio sizing & hypothetical allocation suggestions.
       - Suggests a notional allocation based on risk and trend signals.
    """
    def act(self, ticker: str, technical: Dict[str, Any], risk: Dict[str, Any],
            latest_price: Optional[float], capital: float = 100000.0) ->␣
 ↪Dict[str, Any]:
        # Default safe outputs
        if latest_price is None:
            return {"type": "portfolio", "recommendation": "no_data", "size": 0}
        # Heuristic: if trend is up (SMA20 > SMA50) and RSI < 70 and var␣
 ↪moderate -> allocate more
        sma20 = technical.get('SMA20')
        sma50 = technical.get('SMA50')
        rsi = technical.get('RSI14')
        vol = risk.get('vol_annual', 0)
        # base allocation percent
        alloc_pct = 0.02  # 2% default
        if sma20 and sma50 and sma20 > sma50 and (rsi is None or rsi < 70):
```

```
            # trend is positive
            alloc_pct = 0.05
        if vol and vol > 1.0:  # very risky (edge case)
            alloc_pct = max(0.005, alloc_pct * 0.5)
        # position sizing
        notional = capital * alloc_pct
        shares = int(notional / latest_price) if latest_price > 0 else 0
        rec = {
            "allocation_pct": round(alloc_pct * 100, 3),
            "notional": round(notional, 2),
            "shares": shares,
            "price": latest_price,
            "rationale": f"Trend {'up' if sma20 and sma50 and sma20>sma50 else␣
↪'neutral/down'}, volatility {vol:.3f}"
        }
        return {"type": "portfolio", "recommendation": rec}

# --- 6. Existing EvaluatorAgent (expanded) --------------------
class EvaluatorAgent:
    """Assesses sentiment balance, technical signals, risk - returns score and␣
↪notes."""
    def evaluate(self,
                 news_data: List[Dict[str, Any]],
                 technical: Dict[str, Any],
                 risk: Dict[str, Any],
                 financials: Dict[str, Any]) -> Dict[str, Any]:
        notes = []
        score = 50  # neutral baseline
        # NEWS: average polarity
        news_sentiments = [n['sentiment'] for n in (news_data or [])]
        avg_news = np.mean(news_sentiments) if news_sentiments else 0.0
        if avg_news > 0.05:
            score += 20
            notes.append("News sentiment positive")
        elif avg_news < -0.05:
            score -= 15
            notes.append("News sentiment negative")
        else:
            notes.append("News sentiment neutral")

        # TECHNICAL: trend and RSI
        sma20 = technical.get('SMA20')
        sma50 = technical.get('SMA50')
        rsi = technical.get('RSI14')
        if sma20 and sma50 and sma20 > sma50:
            score += 10
            notes.append("Short-term trend above medium-term (SMA20 > SMA50)")
```

```
        else:
            notes.append("No confirmed short-term uptrend")
        if rsi is not None:
            if rsi > 70:
                score -= 10
                notes.append(f"RSI high ({rsi:.1f}) - possible overbought")
            elif rsi < 30:
                score += 5
                notes.append(f"RSI low ({rsi:.1f}) - possible oversold")

        # RISK: penalize for high vol or deep drawdown
        vol = risk.get('vol_annual', 0)
        max_dd = risk.get('max_drawdown', 0)
        if vol > 0.6:  # threshold heuristic
            score -= 10
            notes.append(f"High volatility ({vol:.2f})")
        if max_dd and abs(max_dd) > 0.5:
            score -= 15
            notes.append(f"Large historical drawdown ({max_dd:.2f})")

        # FINANCIALS: prefer reasonable forward PE if exists
        fpe = financials.get('forwardPE')
        if fpe is not None:
            if fpe < 20:
                score += 5
                notes.append("Forward PE attractive")
            elif fpe > 50:
                score -= 5
                notes.append("Forward PE elevated")

        # clamp score between 0 and 100
        score = max(0, min(100, int(score)))
        conclusion = "Positive" if score >= 65 else ("Neutral" if score >= 40␣
 ↪else "Negative")
        return {"score": score, "conclusion": conclusion, "notes": notes,␣
 ↪"avg_news": float(avg_news)}

# --- 7. New Agent: OptimizerAgent ---------------------------------
class OptimizerAgent:
    """Simple optimizer that adjusts portfolio allocation or suggests defensive␣
 ↪actions
       based on evaluator feedback. Returns refined recommendation string.
    """
    def act(self, portfolio_rec: Dict[str, Any], evaluator_feedback: Dict[str,␣
 ↪Any]) -> Dict[str, Any]:
        rec = portfolio_rec.copy()
        score = evaluator_feedback.get('score', 50)
```

```
        notes = evaluator_feedback.get('notes', [])
        # If negative sentiment & high risk => reduce allocation
        alloc_pct = rec.get('allocation_pct', 0)
        if score < 40:
            new_alloc_pct = max(0.0, alloc_pct * 0.3)
            reason = "Reduce: evaluator indicates weak outlook"
        elif score < 60:
            new_alloc_pct = max(0.0, alloc_pct * 0.7)
            reason = "Trim: evaluator indicates neutral/uncertain outlook"
        else:
            new_alloc_pct = alloc_pct
            reason = "Keep allocation"
        # adjust notional & shares
        notional = new_alloc_pct / 100 * 100000.0  # base capital 100k
        shares = int(notional / rec.get('price', 1)) if rec.get('price') else 0
        optimized = {
            "old_allocation_pct": alloc_pct,
            "new_allocation_pct": round(new_alloc_pct, 4),
            "new_notional": round(notional, 2),
            "new_shares": shares,
            "reason": reason,
            "evaluator_notes": notes
        }
        return {"type": "optimized_portfolio", "optimized": optimized}

# --- 8. InvestmentResearchAgent (coordinator) ------------------
class InvestmentResearchAgent:
    """Coordinator combining all agents and returning final report."""
    def __init__(self):
        self.fin_agent = FinancialsAgent()
        self.news_agent = NewsAgent()
        self.tech_agent = TechnicalAgent()
        self.risk_agent = RiskAgent()
        self.port_agent = PortfolioAgent()
        self.eval_agent = EvaluatorAgent()
        self.opt_agent = OptimizerAgent()
        self.memory = {}

    def run_full(self, ticker: str) -> Dict[str, Any]:
        print(f"\n{'='*80}\nRunning full analysis for {ticker} ...")
        results: Dict[str, Any] = {}

        # 1) Financials & Price DF
        fin = self.fin_agent.act(ticker)
        results['financials'] = fin
        price_df = fin.get('price_df')
```

```
        # 2) Technical indicators
        tech = self.tech_agent.act(price_df)
        results['technical'] = tech['indicators']

        # 3) Risk metrics
        risk = self.risk_agent.act(price_df)
        results['risk'] = risk['metrics']

        # 4) News & Sentiment
        news = self.news_agent.act(ticker)
        results['news'] = news['data']

        # 5) Evaluate combined signals
        eval_res = self.eval_agent.evaluate(
            news_data=results['news'],
            technical=results['technical'],
            risk=results['risk'],
            financials=fin
        )
        results['evaluation'] = eval_res

        # 6) Portfolio recommendation (base)
        portfolio_rec = self.port_agent.act(
            ticker,
            technical=results['technical'],
            risk=results['risk'],
            latest_price=fin.get('latest_price'),
            capital=100000.0
        )
        results['portfolio'] = portfolio_rec

        # 7) Optimize portfolio based on evaluator feedback
        optimized = self.opt_agent.act(portfolio_rec.get('recommendation',␣
↪portfolio_rec), eval_res)
        results['optimized'] = optimized

        # 8) Compose human-readable draft (keeps your original style)
        draft_lines = [
            f"Investment Research Draft for {ticker}",
            f"Generated: {utc_now_iso()}",
            "",
            f"Latest price: {fin.get('latest_price'):.2f}" if fin.
↪get('latest_price') is not None else "Latest price: N/A",
            "",
            "Key Financials:"
        ]
        for k in ['marketCap','trailingPE','forwardPE','dividendYield','beta']:
```

```
            draft_lines.append(f"- {k}: {fin.get(k)}")
        draft_lines.append("\nTechnical indicators:")
        for k,v in results['technical'].items():
            draft_lines.append(f"- {k}: {v}")
        draft_lines.append("\nRisk metrics:")
        for k,v in results['risk'].items():
            draft_lines.append(f"- {k}: {v}")
        draft_lines.append("\nRecent News Highlights:")
        for a in results['news']:
            draft_lines.append(f"- [{a['category']}] {a['headline']}␣
 ↪(sentiment={a['sentiment']})")
        draft_lines.append(f"\nEvaluator Conclusion: {eval_res['conclusion']}␣
 ↪(score={eval_res['score']})")
        draft_lines.append("\nEvaluator Notes:")
        for n in eval_res['notes']:
            draft_lines.append(f"- {n}")
        draft_lines.append("\nPortfolio Recommendation (base):")
        rec = portfolio_rec.get('recommendation')
        draft_lines.append(json.dumps(rec, indent=2))
        draft_lines.append("\nOptimized Allocation:")
        draft_lines.append(json.dumps(optimized.get('optimized'), indent=2))

        draft = "\n".join(draft_lines)
        results['draft'] = draft

        # 9) Memory store (keeps short summary)
        self.memory[ticker] = {"timestamp": utc_now_iso(), "conclusion":␣
 ↪eval_res['conclusion'], "score": eval_res['score']}
        print("\n--- FINAL REPORT ---\n")
        print(draft)
        print(f"\nMemory entries for {ticker}: {len(self.memory)}")
        return results

# --- 9. Run analysis for multiple tickers ---------------------
tickers = ["AAPL","TSLA","GOOG","NVDA","INTC","MSFT"]
ira = InvestmentResearchAgent()
all_results = {}
for t in tickers:
    try:
        all_results[t] = ira.run_full(t)
    except Exception as e:
        print(f"[Main] Error analyzing {t}: {e}")
        all_results[t] = {"error": str(e)}

# --- 10. Summary DataFrame ----------------------------------
summary = pd.DataFrame({
    t: {
```

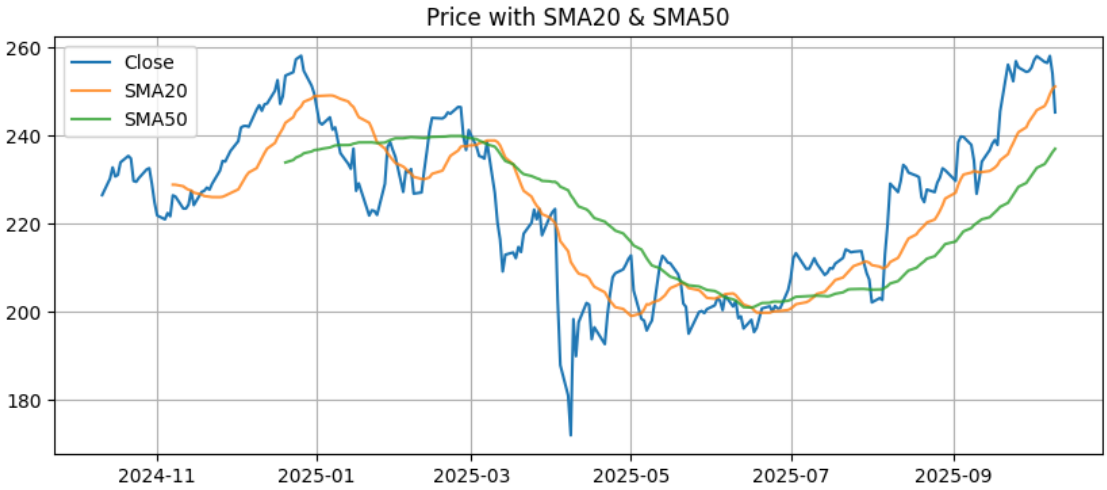
```
        "latest_price": all_results[t]['financials'].get('latest_price') if␣
 ↪'financials' in all_results[t] else None,
        "eval_score": all_results[t]['evaluation'].get('score') if 'evaluation'␣
 ↪in all_results[t] else None,
        "conclusion": all_results[t]['evaluation'].get('conclusion') if␣
 ↪'evaluation' in all_results[t] else None,
        "suggested_alloc_pct":␣
 ↪(all_results[t]['portfolio']['recommendation']['allocation_pct']
                                if 'portfolio' in all_results[t] and␣
 ↪all_results[t]['portfolio']['recommendation'] != "no_data"
                                else None)
    }
    for t in tickers
}).T

print("\n=== Summary of All Analyses ===")
display(summary)
```

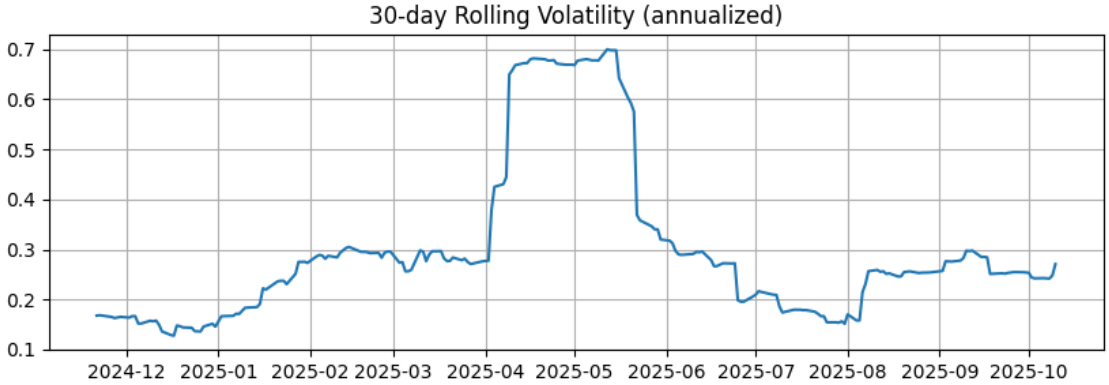
```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
================================================================================
Running full analysis for AAPL …
```


```
/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
```

```
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  "latest_price": float(data['Close'].iloc[-1]),

df['SMA20'] >>> Date
2024-10-11           NaN
2024-10-14           NaN
2024-10-15           NaN
2024-10-16           NaN
2024-10-17           NaN
                    ...
2025-10-06    246.697999
2025-10-07    247.804499
2025-10-08    249.367999
2025-10-09    250.568499
2025-10-10    251.128499
Name: SMA20, Length: 250, dtype: float64
```



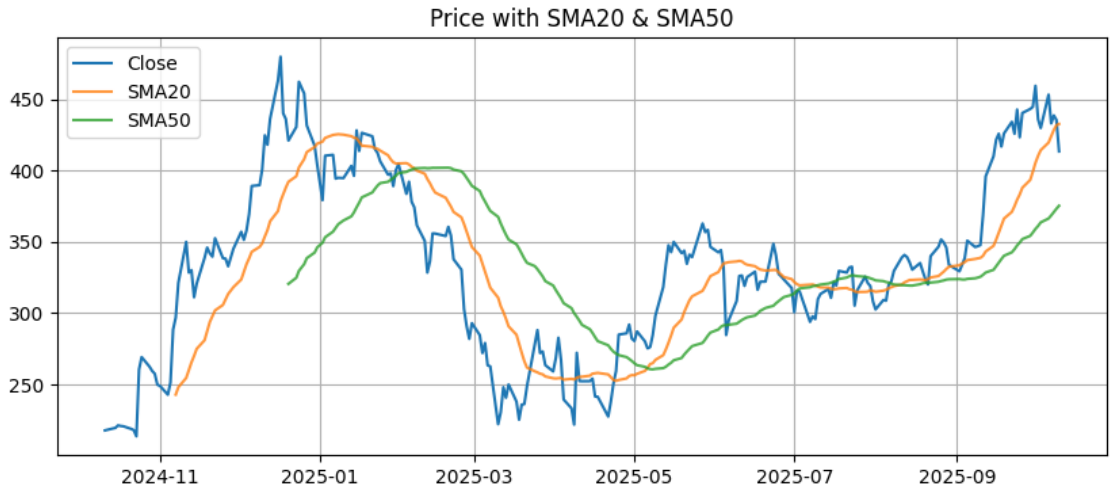


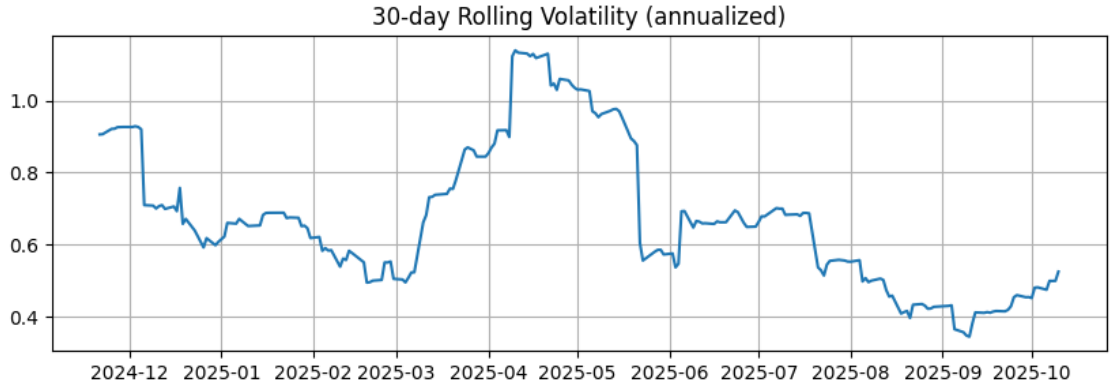
```
--- FINAL REPORT ---

Investment Research Draft for AAPL
Generated: 2025-10-11T05:59:42.155333+00:00

Latest price: 245.27

Key Financials:
- marketCap: 3639902470144
- trailingPE: 37.16212
- forwardPE: 29.515041
- dividendYield: 0.42
- beta: 1.094

Technical indicators:
- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.3248254080588367
- max_drawdown: -0.3336051616882118
- var95: -0.032104851429163916

Recent News Highlights:

Evaluator Conclusion: Neutral (score=50)

Evaluator Notes:
- News sentiment neutral
- No confirmed short-term uptrend

Portfolio Recommendation (base):
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 8,
  "price": 245.27000427246094,
  "rationale": "Trend neutral/down, volatility 0.325"
}

Optimized Allocation:
{
```

```
  "old_allocation_pct": 2.0,
  "new_allocation_pct": 1.4,
  "new_notional": 1400.0,
  "new_shares": 5,
  "reason": "Trim: evaluator indicates neutral/uncertain outlook",
  "evaluator_notes": [
    "News sentiment neutral",
    "No confirmed short-term uptrend"
  ]
}

Memory entries for AAPL: 1

================================================================================
Running full analysis for TSLA …
```


```
/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  "latest_price": float(data['Close'].iloc[-1]),

df['SMA20'] >>> Date
2024-10-11           NaN
2024-10-14           NaN
2024-10-15           NaN
2024-10-16           NaN
2024-10-17           NaN
                    …
2025-10-06    419.653000
2025-10-07    423.959000
2025-10-08    428.503999
```

```
2025-10-09    431.840500
2025-10-10    432.717999
Name: SMA20, Length: 250, dtype: float64
```

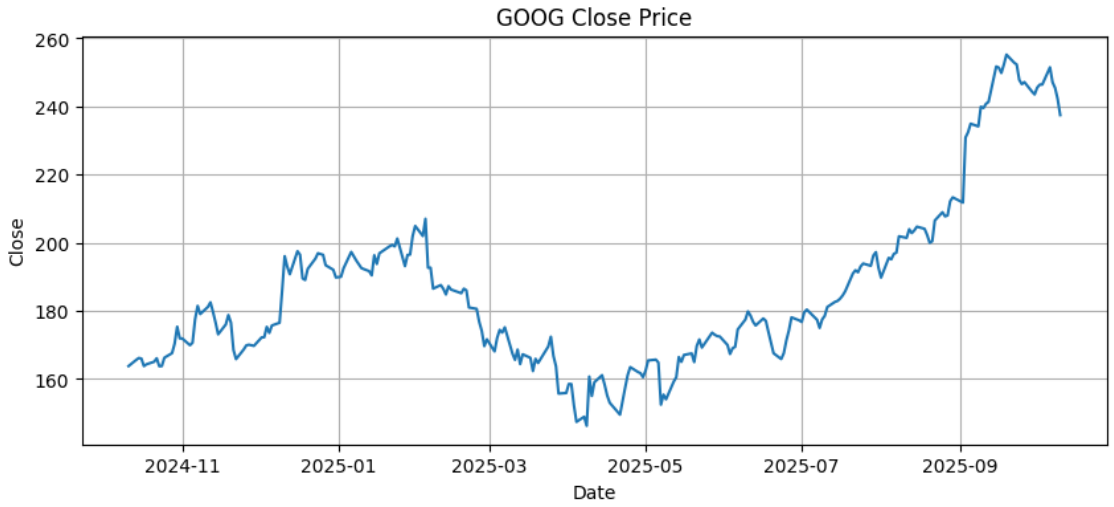





```
--- FINAL REPORT ---

Investment Research Draft for TSLA
Generated: 2025-10-11T05:59:46.244769+00:00

Latest price: 413.49

Key Financials:
- marketCap: 1374916706304
- trailingPE: 243.2294
- forwardPE: 127.62037
```

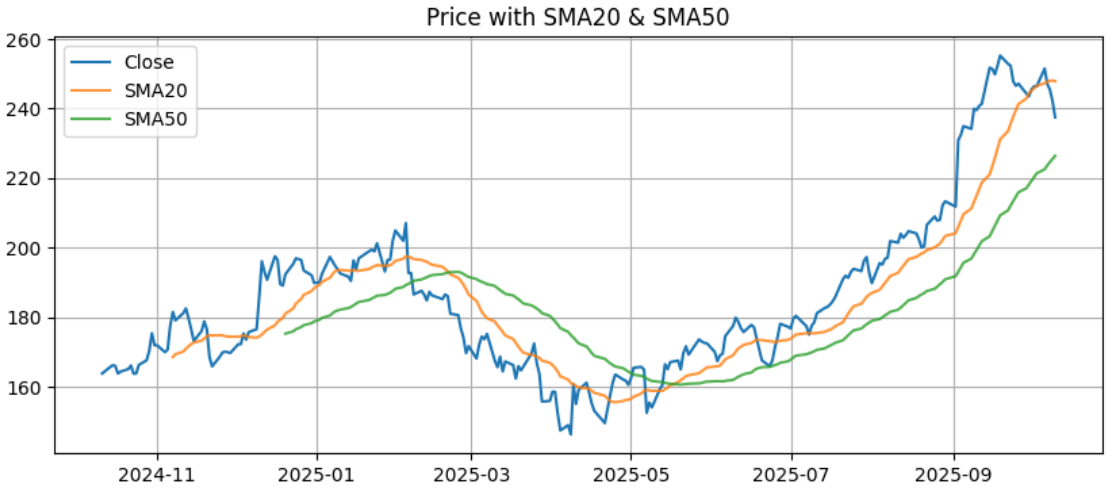
```
- dividendYield: None
- beta: 2.086

Technical indicators:
- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.7077031080423287
- max_drawdown: -0.5376568011858524
- var95: -0.056837875321660236

Recent News Highlights:
- [other] Why We Went Larger On Our Tesla Trade - Investor's Business Daily
(sentiment=0.0)
- [other] Tesla, Inc. ($TSLA) Stock: Shares Drop 2.86% as Musk Eyes Billions -
CoinCentral (sentiment=0.0)
- [earnings] Tesla’s 'Model 2' Is Here - What Does It Mean Ahead Of Earnings? -
Seeking Alpha (sentiment=-0.3125)
- [other] Tesla (TSLA) Stock Rises Over $450, Hits Record $1.5T Market Cap as Q3
Delivery Test Looms - CarbonCredits.com (sentiment=0.0)
- [other] Tesla’s FSD Faces a New Safety Probe. Options Data Tells Us TSLA Stock
Could Be Headed Here. - Yahoo Finance (sentiment=0.1364)

Evaluator Conclusion: Negative (score=20)

Evaluator Notes:
- News sentiment neutral
- No confirmed short-term uptrend
- High volatility (0.71)
- Large historical drawdown (-0.54)
- Forward PE elevated

Portfolio Recommendation (base):
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 4,
  "price": 413.489990234375,
  "rationale": "Trend neutral/down, volatility 0.708"
}

Optimized Allocation:
{
  "old_allocation_pct": 2.0,
```

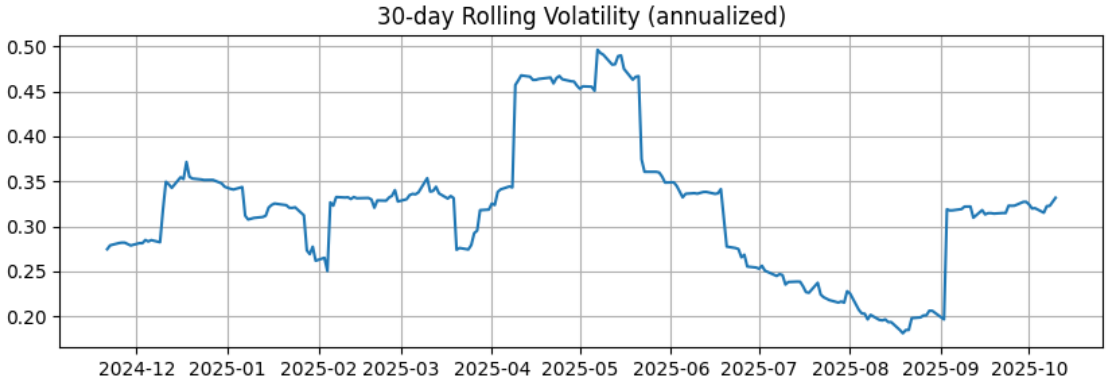
```
  "new_allocation_pct": 0.6,
  "new_notional": 600.0,
  "new_shares": 1,
  "reason": "Reduce: evaluator indicates weak outlook",
  "evaluator_notes": [
    "News sentiment neutral",
    "No confirmed short-term uptrend",
    "High volatility (0.71)",
    "Large historical drawdown (-0.54)",
    "Forward PE elevated"
  ]
}

Memory entries for TSLA: 2

================================================================================
Running full analysis for GOOG …
```




```
df['SMA20'] >>> Date
2024-10-11           NaN
2024-10-14           NaN
2024-10-15           NaN
2024-10-16           NaN
2024-10-17           NaN
                     …
2025-10-06    247.345000
2025-10-07    247.704500
2025-10-08    247.999500
2025-10-09    248.071001
2025-10-10    247.876501
```

```
Name: SMA20, Length: 250, dtype: float64

/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  "latest_price": float(data['Close'].iloc[-1]),
```






```
--- FINAL REPORT ---

Investment Research Draft for GOOG
Generated: 2025-10-11T05:59:47.792303+00:00

Latest price: 237.49

Key Financials:
- marketCap: 2866073305088
```

- trailingPE: 25.318764
- forwardPE: 26.535196
- dividendYield: 0.35
- beta: 1.0

Technical indicators:
- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.3182982894967636
- max_drawdown: -0.2935006055551587
- var95: -0.029754596327532006

Recent News Highlights:
- [other] Google Launches Gemini Enterprise As Tech Giants Race To Offer AI Agents - Investor's Business Daily (sentiment=0.0)
- [other] Alphabet Inc. (GOOG) Stock Sinks As Market Gains: Here's Why - Yahoo Finance (sentiment=-0.1)
- [other] Alphabet (NASDAQ:GOOG) Stock Price Down 1.9% - Here's Why - MarketBeat (sentiment=-0.1556)
- [other] Alphabet's stock could soar if Google's Gemini AI gets a TikTok-style makeover - MarketWatch (sentiment=0.0)
- [other] Alphabet: Undervalued Mag-7 Stock Hiding In Plain Sight (NASDAQ:GOOG) - Seeking Alpha (sentiment=-0.2143)

Evaluator Conclusion: Negative (score=35)

Evaluator Notes:
- News sentiment negative
- No confirmed short-term uptrend

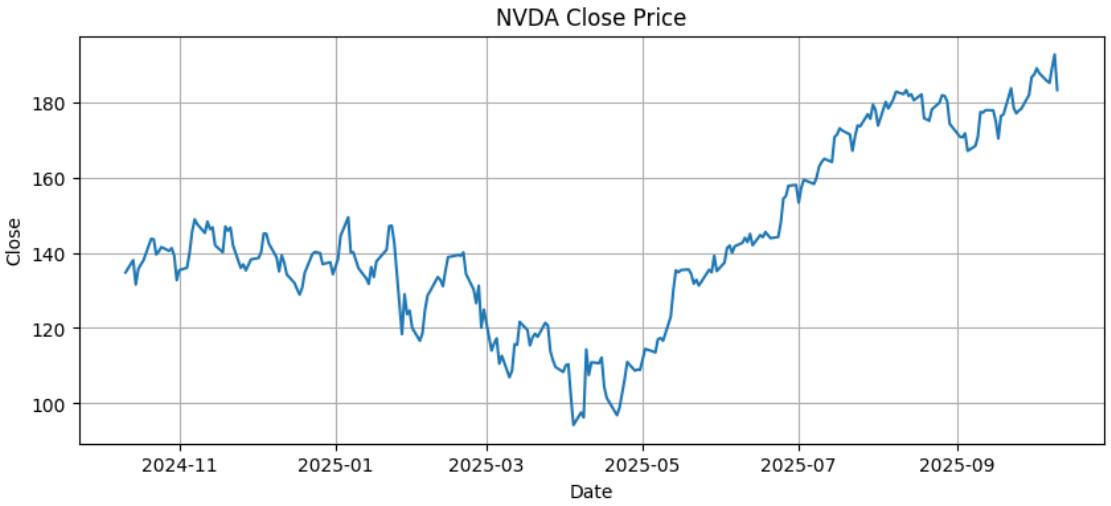
Portfolio Recommendation (base):

```
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 8,
  "price": 237.49000549316406,
  "rationale": "Trend neutral/down, volatility 0.318"
}
```


Optimized Allocation:

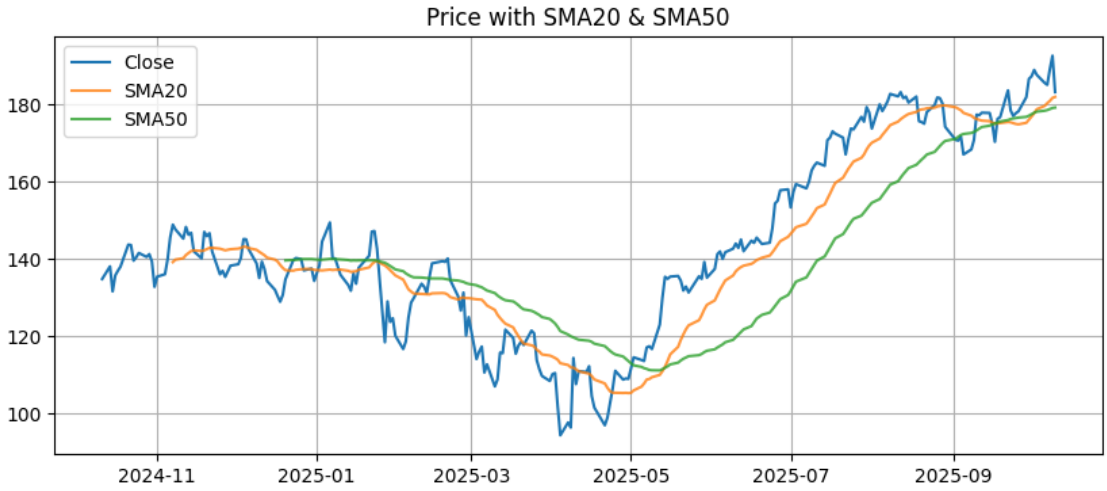
```
{
  "old_allocation_pct": 2.0,
  "new_allocation_pct": 0.6,
```

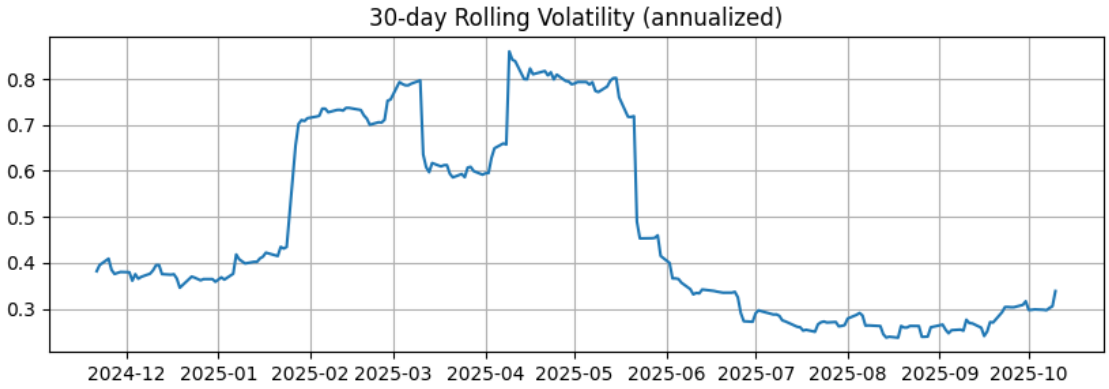
```
  "new_notional": 600.0,
  "new_shares": 2,
  "reason": "Reduce: evaluator indicates weak outlook",
  "evaluator_notes": [
    "News sentiment negative",
    "No confirmed short-term uptrend"
  ]
}

Memory entries for GOOG: 3

================================================================================
Running full analysis for NVDA …
```




```
df['SMA20'] >>> Date
2024-10-11           NaN
2024-10-14           NaN
2024-10-15           NaN
2024-10-16           NaN
2024-10-17           NaN
                     …
2025-10-06    179.575019
2025-10-07    180.289500
2025-10-08    180.879000
2025-10-09    181.649001
2025-10-10    181.916000
Name: SMA20, Length: 250, dtype: float64

/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
```

```
"latest_price": float(data['Close'].iloc[-1]),
```

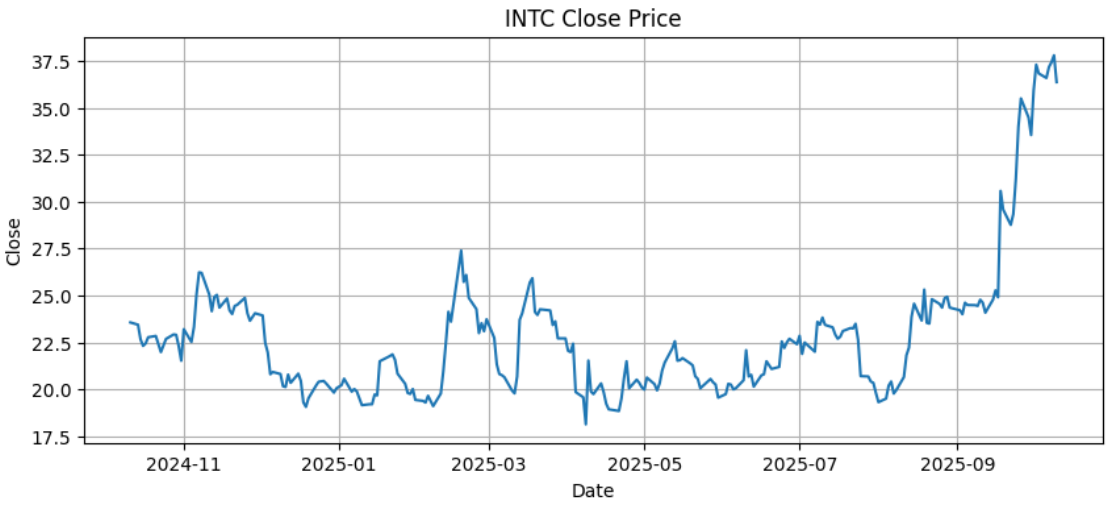




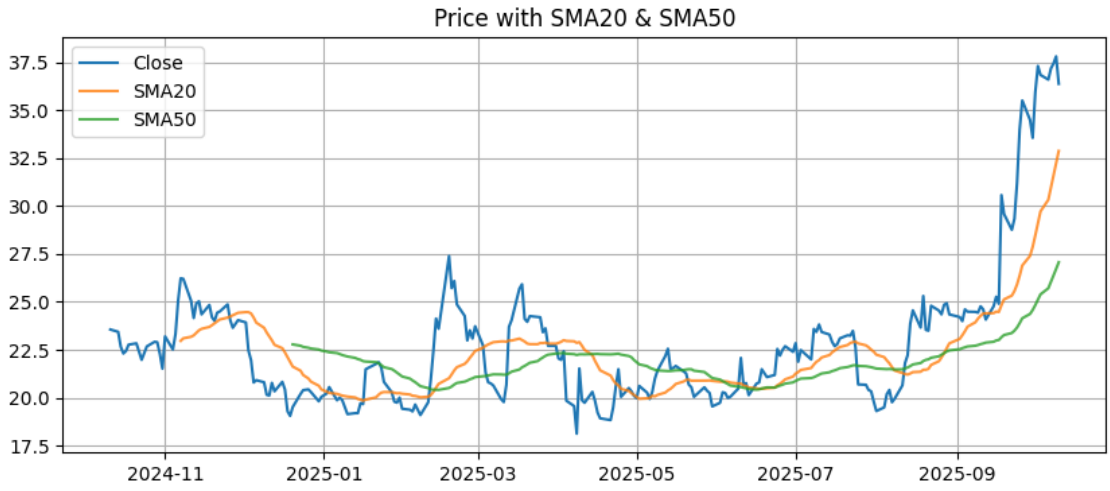

```
--- FINAL REPORT ---

Investment Research Draft for NVDA
Generated: 2025-10-11T05:59:49.195327+00:00

Latest price: 183.16

Key Financials:
- marketCap: 4459396595712
- trailingPE: 52.034092
- forwardPE: 44.456314
- dividendYield: 0.02
- beta: 2.123
```

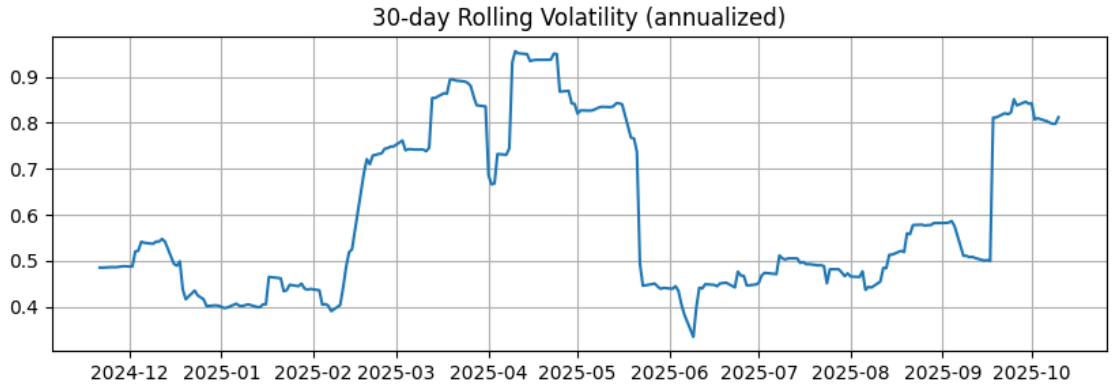
```
Technical indicators:
- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.49569069391769677
- max_drawdown: -0.3688102669914984
- var95: -0.04705952469839477

Recent News Highlights:
- [other] Nvidia Stock Spikes To Record High On UAE Trade Approval - Investor's
Business Daily (sentiment=0.08)
- [other] Prediction: This Unstoppable Stock Will Join Nvidia, Microsoft, Apple,
and Alphabet in the $3 Trillion Club Before 2028 - The Motley Fool
(sentiment=0.6)
- [other] Tech megacaps lose $770 billion in value as Nasdaq suffers steepest
drop since April - CNBC (sentiment=-0.6)
- [other] Nvidia CEO Huang Jen Hsun sells $42.8 million in NVDA stock -
Investing.com (sentiment=0.0)
- [other] Nvidia-backed AI stock's monster run gets CoreWeave jolt - Yahoo
Finance (sentiment=0.0)

Evaluator Conclusion: Neutral (score=50)

Evaluator Notes:
- News sentiment neutral
- No confirmed short-term uptrend

Portfolio Recommendation (base):
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 10,
  "price": 183.16000366210938,
  "rationale": "Trend neutral/down, volatility 0.496"
}

Optimized Allocation:
{
  "old_allocation_pct": 2.0,
  "new_allocation_pct": 1.4,
  "new_notional": 1400.0,
  "new_shares": 7,
  "reason": "Trim: evaluator indicates neutral/uncertain outlook",
```

```
  "evaluator_notes": [
    "News sentiment neutral",
    "No confirmed short-term uptrend"
  ]
}

Memory entries for NVDA: 4

================================================================================
Running full analysis for INTC …
```




```
df['SMA20'] >>> Date
2024-10-11        NaN
2024-10-14        NaN
2024-10-15        NaN
2024-10-16        NaN
2024-10-17        NaN
                  …
2025-10-06    30.3245
2025-10-07    30.9610
2025-10-08    31.5940
2025-10-09    32.2535
2025-10-10    32.8680
Name: SMA20, Length: 250, dtype: float64

/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  "latest_price": float(data['Close'].iloc[-1]),
```






```
--- FINAL REPORT ---

Investment Research Draft for INTC
Generated: 2025-10-11T05:59:50.730753+00:00

Latest price: 36.37

Key Financials:
- marketCap: 173020004352
- trailingPE: None
- forwardPE: 37.494843
- dividendYield: None
- beta: 1.33

Technical indicators:
```

- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.6305276798853632
- max_drawdown: -0.3380796069817177
- var95: -0.05278583669672623

Recent News Highlights:
- [other] Intel Stock Falls After Downgrade. Why This Analyst Is Worried After Nvidia, SoftBank Deals - Barron's (sentiment=0.0)
- [other] AMD, Nvidia, and Intel Stock Forecast: Microchip Companies Poised for Growth Amid Rising AI Demand - TECHi (sentiment=0.0)
- [other] Intel Stock Price Target Raised to $30 Amid Strategic Moves and Foundry Expansion - TECHi (sentiment=0.0)
- [other] Intel's Stock Price Has Doubled Since Hitting Its 2025 Low-Watch These Key Levels - Investopedia (sentiment=0.0)
- [other] Intel stock is up 50% over the last month, putting U.S. stake at $16 billion - CNBC (sentiment=0.0)

Evaluator Conclusion: Neutral (score=40)

Evaluator Notes:
- News sentiment neutral
- No confirmed short-term uptrend
- High volatility (0.63)

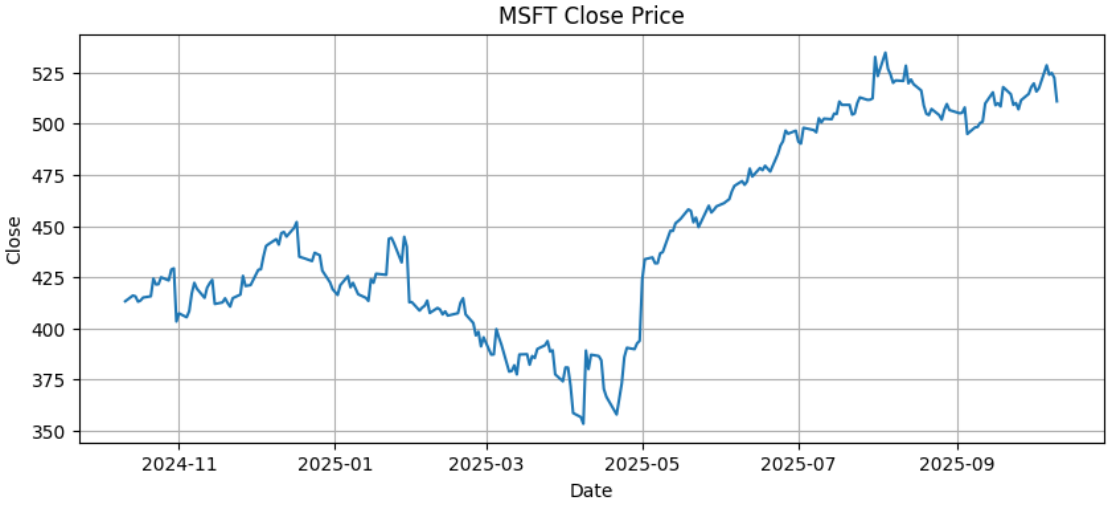
Portfolio Recommendation (base):

```
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 54,
  "price": 36.369998931884766,
  "rationale": "Trend neutral/down, volatility 0.631"
}
```

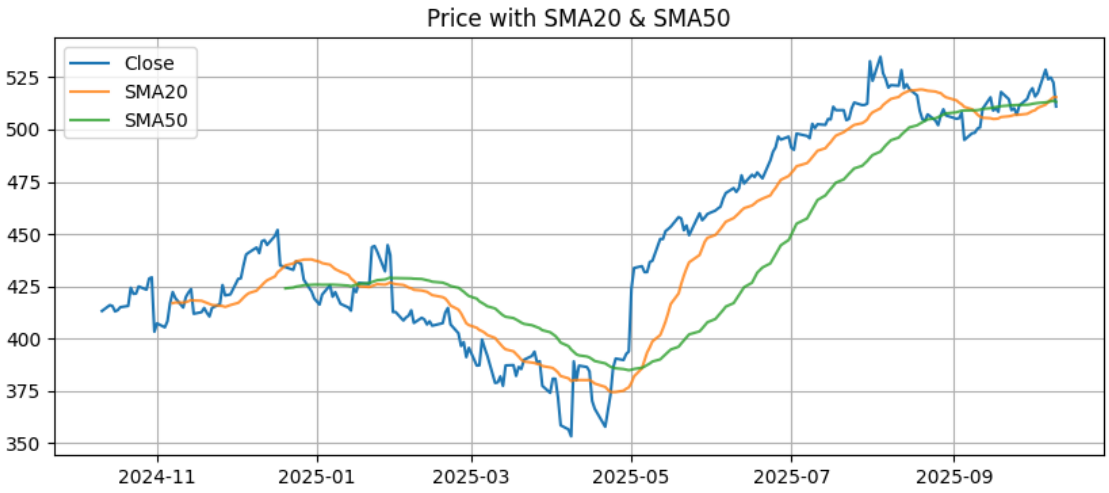

Optimized Allocation:

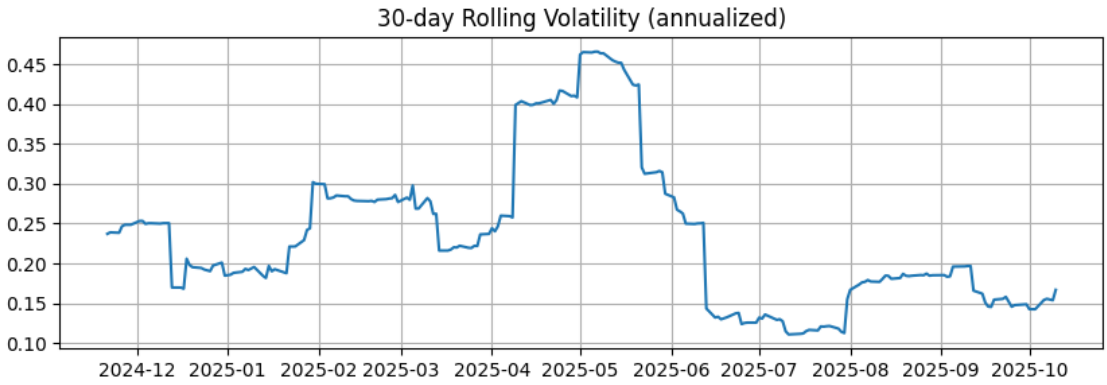
```
{
  "old_allocation_pct": 2.0,
  "new_allocation_pct": 1.4,
  "new_notional": 1400.0,
  "new_shares": 38,
  "reason": "Trim: evaluator indicates neutral/uncertain outlook",
  "evaluator_notes": [
    "News sentiment neutral",
```

```
    "No confirmed short-term uptrend",
    "High volatility (0.63)"
  ]
}

Memory entries for INTC: 5

================================================================================
Running full analysis for MSFT …
```




```
df['SMA20'] >>> Date
2024-10-11           NaN
2024-10-14           NaN
2024-10-15           NaN
2024-10-16           NaN
2024-10-17           NaN
                    …
2025-10-06    511.836499
2025-10-07    513.114998
2025-10-08    514.338997
2025-10-09    515.408498
2025-10-10    515.461497
Name: SMA20, Length: 250, dtype: float64

/tmp/ipython-input-4079960587.py:52: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  "latest_price": float(data['Close'].iloc[-1]),
```






```
--- FINAL REPORT ---

Investment Research Draft for MSFT
Generated: 2025-10-11T05:59:52.226768+00:00

Latest price: 510.96

Key Financials:
- marketCap: 3798050537472
- trailingPE: 37.460407
- forwardPE: 34.177925
- dividendYield: 0.71
- beta: 1.023

Technical indicators:
```

```
- SMA20: None
- SMA50: None
- EMA20: None
- RSI14: None
- close: None

Risk metrics:
- vol_annual: 0.24765858531911517
- max_drawdown: -0.21825690565450617
- var95: -0.021679437448987777

Recent News Highlights:
- [other] Jim Cramer About Microsoft (MSFT) Bull Comments: ‘He’s Been Dead
Right’ - Yahoo Finance (sentiment=0.0429)
- [other] Is There Still Room For Microsoft Stock To Grow? - Trefis
(sentiment=0.0)
- [other] Price-Driven Insight from (MSFT) for Rule-Based Strategy -
news.stocktradersdaily.com (sentiment=0.0)
- [other] Why Microsoft (MSFT) is a Top Momentum Stock for the Long-Term -
Nasdaq (sentiment=0.5)
- [other] Microsoft (NASDAQ: MSFT) Stock Price Prediction for 2025: Where Will
It Be in 1 Year - 24/7 Wall St. (sentiment=0.0)

Evaluator Conclusion: Positive (score=70)

Evaluator Notes:
- News sentiment positive
- No confirmed short-term uptrend

Portfolio Recommendation (base):
{
  "allocation_pct": 2.0,
  "notional": 2000.0,
  "shares": 3,
  "price": 510.9599914550781,
  "rationale": "Trend neutral/down, volatility 0.248"
}

Optimized Allocation:
{
  "old_allocation_pct": 2.0,
  "new_allocation_pct": 2.0,
  "new_notional": 2000.0,
  "new_shares": 3,
  "reason": "Keep allocation",
  "evaluator_notes": [
    "News sentiment positive",
    "No confirmed short-term uptrend"
```

```
  ]
}
```

```
Memory entries for MSFT: 6

=== Summary of All Analyses ===
```

	latest_price	eval_score	conclusion	suggested_alloc_pct
AAPL	245.270004	50	Neutral	2.0
TSLA	413.48999	20	Negative	2.0
GOOG	237.490005	35	Negative	2.0
NVDA	183.160004	50	Neutral	2.0
INTC	36.369999	40	Neutral	2.0
MSFT	510.959991	70	Positive	2.0

[13]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
!apt-get update --quiet
!pip install nbconvert --quiet
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc␣
  ↪texlive-fonts-recommended texlive-plain-generic --quiet
!ls /usr/share/texmf/tex/latex/
!which xelatex
!echo $PATH
!export PATH=/Library/TeX/texbin:$PATH
!which xelatex

from IPython.display import clear_output
clear_output(wait=False)
```

```
Hit:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:2 https://cli.github.com/packages stable InRelease
Hit:3 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Hit:10 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:11 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Reading package lists…
```

[12]:
```
!jupyter nbconvert --to pdf "/content/drive/My Drive/Colab Notebooks/
↪Stock_Analysis_RJ-v2-TechRiskPort_Opt.ipynb"
```

```
[NbConvertApp] Converting notebook /content/drive/My Drive/Colab
Notebooks/Stock_Analysis_RJ-v2-TechRiskPort_Opt.ipynb to pdf
[NbConvertApp] Support files will be in
Stock_Analysis_RJ-v2-TechRiskPort_Opt_files/
[NbConvertApp] Making directory ./Stock_Analysis_RJ-v2-TechRiskPort_Opt_files
[NbConvertApp] Writing 103629 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 815815 bytes to /content/drive/My Drive/Colab
Notebooks/Stock_Analysis_RJ-v2-TechRiskPort_Opt.pdf
```