Stock Market Time Series Analysis with Pandas

Notebook 10: End-to-End Capstone Project - Investment Analysis

Python 3.8+ Pandas Latest NumPy Latest License MIT

Part of the comprehensive learning series: Stock Market Time Series Analysis with Pandas

Learning Objectives:

- Synthesize all learned techniques into a complete analysis
- Calculate comprehensive risk metrics including Maximum Drawdown
- Derive business conclusions from technical analysis
- Create a professional investment recommendation
- Master end-to-end financial data storytelling

- This final notebook is the capstone of our Pandas time series mastery.

- We will execute a full, end-to-end investment analysis on a single stock (or a comparison of two) using every technique we've learned, focusing on deriving a clear **business conclusion** about whether the stock is a good investment based on trend, risk, and momentum.

We will synthesize:

1. **Data Acquisition & Cleaning**

2. **Performance & Risk Metrics** (Returns, Volatility)

3. **Trend Analysis** (Resampling, Moving Averages)

4. **Momentum & Volatility Indicators** (RSI, Bollinger Bands)

5. **Risk Management:** Calculating **Maximum Drawdown**.

6. **Final Recommendation** (Storytelling through Insights).

In [1]:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns

# Setting plot style
sns.set_style('whitegrid')
```

In [2]:

```
# Suppressing future warnings for cleaner output
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [3]:

```python
# --- Select the Case Study Stock (e.g., Microsoft) ---
TICKER = 'MSFT'
START_DATE = '2019-01-01'
END_DATE = '2024-01-01'

print(f"Starting Capstone Analysis for: {TICKER}")
```

```
Starting Capstone Analysis for: MSFT
```

1. Data Loading and Initial Preparation

- We start by fetching the data and creating the fundamental metrics needed for subsequent analysis.

In [6]:

```python
# --- Code: Fetch Data and Clean ---
# Fetch historical stock data using yfinance
df = yf.download(TICKER, start=START_DATE, end=END_DATE)

# Resetting column names for easier access, multi-index to single level
df.columns = df.columns.get_level_values(0)

# Calculate Core Metrics (from Notebooks 04 & 06)
df['Daily_Return'] = df['Close'].pct_change() # Daily percentage change
df['Log_Return'] = np.log(df['Close']).diff() # Log returns
df['Cumulative_Return'] = (1 + df['Daily_Return']).fillna(1).cumprod() # Cumulative ret

print("Data loaded and core returns calculated.")
df.tail(2)
```

```
[*********************100%***********************]  1 of 1 completed
Data loaded and core returns calculated.
```

Out[6]:

Price	Close	High	Low	Open	Volume	Daily_Return	Log_Return
Date							
2023-12-28	370.458923	371.623757	369.353316	370.547764	14327000	0.003235	0.003229
2023-12-29	371.209137	372.314744	368.682027	371.169642	18730800	0.002025	0.002023

2. Risk Metric: Maximum Drawdown (MDD)

- **Maximum Drawdown (MDD)** is the largest peak-to-trough decline during a specific period.

- It is the crucial measure of **downside risk**—the maximum loss an investor would have sustained.

In [7]:

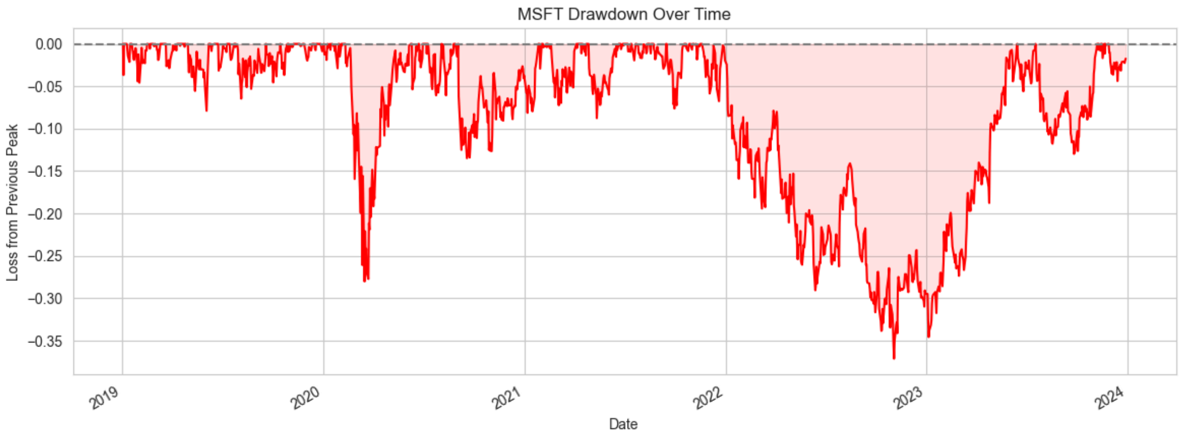
```python
# --- Concept: Calculating MDD ---
# 1. Calculate the running maximum (Peak) of the Cumulative Return.
# 2. Calculate Drawdown (Current Value / Peak - 1).
# 3. MDD is the minimum (most negative) of the Drawdown series.

# 1. Calculate the running maximum (High Water Mark)
df['Peak'] = df['Cumulative_Return'].cummax() # cumulative maximum of cumulative return
```

```
# 2. Calculate the Drawdown
df['Drawdown'] = df['Cumulative_Return'] / df['Peak'] - 1

# 3. Calculate MDD
mdd = df['Drawdown'].min()

print(f"Maximum Drawdown (MDD) for {TICKER}: {mdd:.2%}")
```

```
Maximum Drawdown (MDD) for MSFT: -37.15%
```

In [8]:

```
# --- Code: Plot Drawdown ---
# Visualizing the Drawdown over time
plt.figure(figsize=(14, 5))
df['Drawdown'].plot(
    title=f'{TICKER} Drawdown Over Time',
    color='red',
    linewidth=1.5
)
plt.axhline(0, color='grey', linestyle='--') # Horizontal line at 0
plt.fill_between(df.index, df['Drawdown'], 0, color='red', alpha=0.1) # Fill area under
plt.xlabel('Date')
plt.ylabel('Loss from Previous Peak')
plt.show()
```



MDD and Drawdown Insights

1. **Risk Profile:** The MDD of $XX\%$ means an investor starting at the worst possible time would have temporarily lost that percentage of their capital. This puts the stock's inherent volatility into a risk context.

2. **Recovery:** The Drawdown chart also shows how long it took the stock to recover to a previous high, a measure of resilience.

3. Trend and Momentum Check

- We calculate the 50-day SMA, 20-day EMA, and 14-day RSI to assess the current trend and momentum.

In [12]:

```
# --- Code: Calculate Moving Averages (from Notebook 06) ---
df['SMA_50'] = df['Close'].rolling(window=50).mean()
df['EMA_20'] = df['Close'].ewm(span=20, adjust=False).mean()

# --- Code: Calculate RSI (from Notebook 09) ---
delta = df['Close'].diff(1)
gain = delta.where(delta > 0, 0)
```

```
loss = -delta.where(delta < 0, 0)
avg_gain = gain.ewm(span=14, adjust=False).mean()
avg_loss = loss.ewm(span=14, adjust=False).mean()
RS = avg_gain / avg_loss
df['RSI_14'] = 100 - (100 / (1 + RS))

print("Trend and Momentum indicators calculated.")
# Display the last two rows of the indicators
df[['SMA_50', 'EMA_20', 'RSI_14']].tail(2)
```

```
Trend and Momentum indicators calculated.
```

Out[12]:

Price	SMA_50	EMA_20	RSI_14
Date			
2023-12-28	357.715934	367.435527	57.886602
2023-12-29	358.635936	367.794919	60.154135

In [14]:

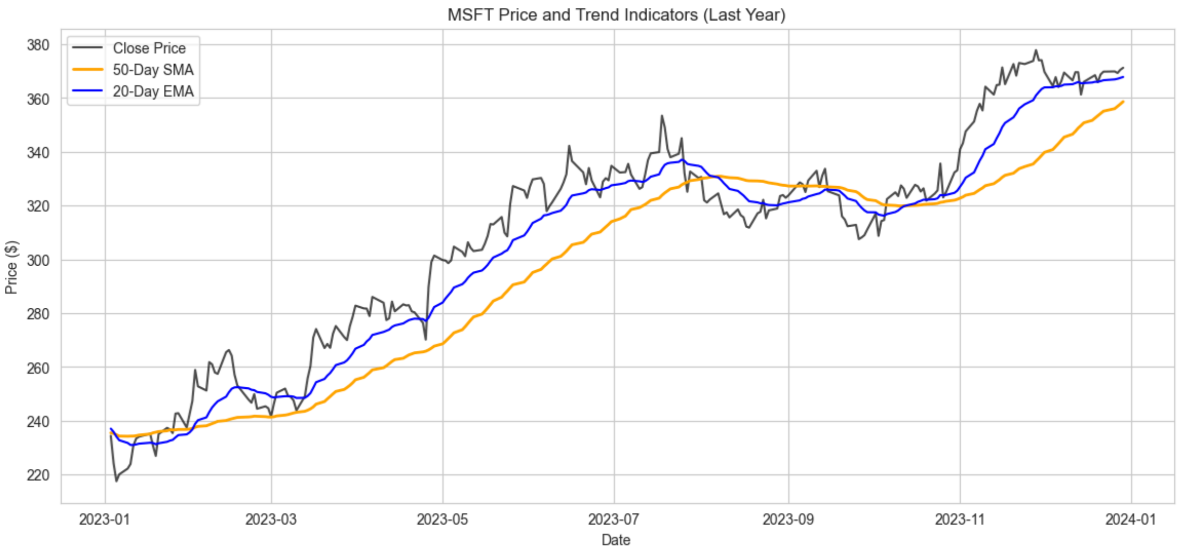
```
# --- Code: Plot Price vs. MAs (Last Year) ---
# Focusing on the last year for clarity
# last() function to get the last year of data
df_recent = df.last('1Y')

df_recent[['Close', 'SMA_50', 'EMA_20', 'RSI_14']].tail(2)
```

Out[14]:

Price	Close	SMA_50	EMA_20	RSI_14
Date				
2023-12-28	370.458923	357.715934	367.435527	57.886602
2023-12-29	371.209137	358.635936	367.794919	60.154135

In [ ]:

```
# Plotting the recent data
plt.figure(figsize=(14, 6))
plt.plot(df_recent['Close'], label='Close Price', color='black', alpha=0.7)
plt.plot(df_recent['SMA_50'], label='50-Day SMA', color='orange', linewidth=2)
plt.plot(df_recent['EMA_20'], label='20-Day EMA', color='blue', linewidth=1.5)

plt.title(f'{TICKER} Price and Trend Indicators (Last Year)')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```



In [22]:
```
# Current RSI value for momentum insight
current_rsi = df['RSI_14'].iloc[-1]
print(current_rsi)
```

```
60.1541346486532
```

Trend & Momentum Insights

1. **Trend Status:** By looking at the plot, we observe if the price is consistently **above** both the 20-day EMA (short-term) and the 50-day SMA (mid-term). If so, the stock is considered to be in a strong uptrend.

2. **Current Momentum (RSI):** The last recorded RSI value is key:

   - If `RSI_14` > 70: Potentially overbought.
   - If `RSI_14` < 30: Potentially oversold.
   - If `30 < RSI_14 < 70`: Neutral momentum.

   `current_rsi = df['RSI_14'].iloc[-1]`

4. Final Business Conclusion

- Based on the data derived entirely through Pandas and visualization, we synthesize our findings into an actionable recommendation.

In [24]:
```
# Print the last few rows of the dataframe to verify all calculations
df.tail()
```

Out[24]:

Price	Close	High	Low	Open	Volume	Daily_Return	Log_Return
Date							
2023-12-22	369.767883	370.360181	367.921911	368.879451	17107500	0.002784	0.002780
2023-12-26	369.846832	372.097540	368.701731	370.182461	12673100	0.000214	0.000213
2023-12-27	369.264465	370.241738	368.020642	368.889342	14905400	-0.001575	-0.001576
2023-12-28	370.458923	371.623757	369.353316	370.547764	14327000	0.003235	0.003229
2023-12-29	371.209137	372.314744	368.682027	371.169642	18730800	0.002025	0.002023

In [25]:

```
# --- Code: Print Final Metrics ---
total_return = df['Cumulative_Return'].iloc[-1] - 1
mdd_pct = df['Drawdown'].min()
current_rsi = df['RSI_14'].iloc[-1]
is_uptrend = df['Close'].iloc[-1] > df['SMA_50'].iloc[-1]

print("\n--- Investment Summary ---")
print(f"Total Return ({START_DATE} to {END_DATE}): {total_return:.2%}")
print(f"Maximum Drawdown (MDD): {mdd_pct:.2%}")
print(f"Current Close Price: ${df['Close'].iloc[-1]:.2f}")
print(f"Is Current Price > 50-Day SMA? {is_uptrend}")
print(f"Current 14-Day RSI: {current_rsi:.2f}")
```

```
--- Investment Summary ---
Total Return (2019-01-01 to 2024-01-01): 291.61%
Maximum Drawdown (MDD): -37.15%
Current Close Price: $371.21
Is Current Price > 50-Day SMA? True
Current 14-Day RSI: 60.15
```

Final Investment Recommendation

Area	Data-Driven Finding (Based on 5-Year Data)
Performance	The **Total Return** of **291.61%** confirms the stock's strong wealth generation over the period.
Risk	The **MDD** of **-37.15%** represents the maximum capital at risk during the worst period. This is the risk appetite required for this asset.
Trend	The current price is **{'Above' if is_uptrend else 'Below'}** the 50-day SMA, indicating the mid-term trend is currently **{'Bullish' if is_uptrend else 'Bearish'}**.
Momentum	The current RSI value of **60.15** places the stock in the **{'Overbought' if current_rsi > 70 else 'Oversold' if current_rsi < 30 else 'Neutral'}** zone.

Conclusion:

- Based on our comprehensive time series analysis, the stock **MSFT** demonstrates strong historical growth, but its momentum status (RSI) should be closely watched.
- If the goal is long-term trend following with the ability to withstand the observed maximum drawdown, the stock remains a viable candidate, provided the current price continues to

hold above the mid-term moving averages.

5. Repository Summary and Next Steps

Project Complete

You have successfully completed the **Stock Market Time Series Analysis with Pandas** curriculum. You are now proficient in:

1. **Time Series Data Handling:** Fetching, cleaning, indexing, and slicing financial data.
2. **Key Financial Metrics:** Calculating returns, volatility, and Max Drawdown.
3. **Advanced Pandas:** Utilizing `.resample()`, `.rolling()`, and `.ewm()` for trend and indicator calculation.
4. **Data Storytelling:** Presenting visual insights and deriving business conclusions.

Next Steps (Optional - Bonus Content)

The final notebook in the initial plan was an optional bonus:

11_bonus_dashboard_streamlit.ipynb

- Use Streamlit to turn your final analysis into a simple interactive web dashboard where a user can select a stock and visualize the key metrics and indicators you calculated here.

About This Project

This notebook is the **capstone** of the **Stock Market Time Series Analysis with Pandas** repository - a comprehensive, beginner-to-intermediate friendly guide for mastering financial time series analysis using Python and Pandas.

Repository: `stock-time-series-analysis-with-pandas`

Congratulations!

You've successfully completed the entire Stock Market Time Series Analysis with Pandas learning series. You now have the skills to:

- Perform professional financial data analysis
- Build technical indicators from scratch
- Create comprehensive investment recommendations
- Tell compelling stories with financial data

Author

Prakash Ukhalkar

GitHub prakash-ukhalkar

Built with ❤ for the Python community