Stock Market Time Series Analysis with Pandas

Notebook 11: BONUS - Interactive Dashboard with Streamlit

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

Learning Objectives:

- Transform Pandas analysis into interactive web applications
- Master Streamlit framework for financial dashboards
- Create user-friendly interfaces for stock analysis
- Deploy technical indicators in real-time applications
- Build professional data science web applications

- We have mastered complex time series analysis using Pandas.
- The final step is to transition this analysis into an **interactive and accessible format** using **Streamlit**, a framework that turns Python scripts into web apps.
    - **Note:** Streamlit applications are best run as a separate Python file (`.py`) from your command line.
    - This notebook provides the core code and instructions needed to create and run the dashboard.

Goals of this Guide:

1. **Define a Helper Function:** Consolidate the data fetching and indicator calculation logic.
2. **Create the App Script:** Write the Streamlit UI and plotting logic (`app.py`).
3. **Provide Instructions:** Guide the user on running the dashboard.

1. Required Setup and Helper Function

- First, ensure Streamlit is installed (add `streamlit` to your `requirements.txt` if not already there).

- We will define a function to perform all the data preparation from Notebooks 01-09 in one step.

In [1]:
```
# --- Concept: Consolidating Logic ---
# This function packages all the logic (fetch, clean, calculate returns,
# calculate MAs, calculate RSI)
# into a single reusable block.

# You would typically put this function in a utility file (e.g., src/utils.p

import yfinance as yf
import pandas as pd

def get_analyzed_data(ticker, start_date, end_date):
    """Fetches data and calculates all necessary technical indicators."""
    df = yf.download(ticker, start=start_date, end=end_date)
    if df.empty:
        return None
    df.columns = df.columns.get_level_values(0) # Clean MultiIndex

    # 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()

    # Calculate RSI (simplified 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))

    return df

print("Helper function defined. We can now use this in the Streamlit app.")
```

```
Helper function defined. We can now use this in the Streamlit app.
```

2. Streamlit App Template (`app.py`)

- Create a new file named `app.py` in your main repository directory and paste the following code into it.
- This script uses Streamlit to create an interactive interface with user controls and dynamic plots.

In [ ]:
```
import streamlit as st
import yfinance as yf
import pandas as pd
import numpy as np
from datetime import date

# --- 1. ANALYSIS HELPER FUNCTION (Copy/Paste from Section 1) ---
```

```python
def get_analyzed_data(ticker, start_date, end_date):
    # ... (Paste the function code here in the actual app.py file)
    df = yf.download(ticker, start=start_date, end=end_date)
    if df.empty:
        return None
    df.columns = df.columns.get_level_values(0)
    df['SMA_50'] = df['Close'].rolling(window=50).mean()
    df['EMA_20'] = df['Close'].ewm(span=20, adjust=False).mean()
    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))
    return df


# --- 2. STREAMLIT UI LAYOUT ---

st.set_page_config(
    page_title="Pandas Stock Analysis Dashboard",
    layout="wide"
)

st.title("Interactive Stock Time Series Analysis")
st.markdown("Powered by Pandas, yfinance, and Streamlit.")

# --- Sidebar for User Input ---
st.sidebar.header("Stock and Date Selection")
ticker_symbol = st.sidebar.text_input("Ticker Symbol (e.g., AAPL)", value='
start = st.sidebar.date_input("Start Date", value=pd.to_datetime('2021-01-0
end = st.sidebar.date_input("End Date", value=date.today())
show_sma = st.sidebar.checkbox("Show 50-Day SMA", value=True)
show_ema = st.sidebar.checkbox("Show 20-Day EMA", value=True)

# --- Main App Logic ---

if ticker_symbol:
    # Fetch and Analyze Data
    data = get_analyzed_data(ticker_symbol, start, end)

    if data is not None:
        st.subheader(f"Analysis for {ticker_symbol.upper()}")

        # --- A. Price and Trend Chart ---
        st.markdown("### Price and Trend")
        plot_cols = ['Close']
        if show_sma: plot_cols.append('SMA_50')
        if show_ema: plot_cols.append('EMA_20')

        st.line_chart(data[plot_cols])

        # --- B. Key Metrics (from Notebook 10) ---
        col1, col2, col3 = st.columns(3)

        with col1:
```

```
            # Calculate Total Return (Simplified from Notebook 04)
            total_return = (data['Close'].iloc[-1] / data['Close'].iloc[0])
            st.metric("Total Return", f"{total_return:.2%}")

        with col2:
            # Get Last RSI Value
            last_rsi = data['RSI_14'].iloc[-1]
            st.metric("Current 14-Day RSI", f"{last_rsi:.2f}")

        with col3:
            # Calculate Volatility (Annualized from Daily Log Returns)
            data['Log_Return'] = np.log(data['Close']).diff()
            annualized_vol = data['Log_Return'].std() * np.sqrt(252)
            st.metric("Annualized Volatility", f"{annualized_vol:.2%}")

        # --- C. Momentum Chart (RSI) ---
        st.markdown("### Momentum (RSI)")
        st.line_chart(data['RSI_14'])
        st.markdown("\*Note: RSI above 70 is overbought, below 30 is overso

    else:
        st.error("Could not retrieve data for this ticker/date range.")


# End of app.py script
```

3. How to Run the Dashboard

- Assuming you have saved the code above into a file named `app.py` in the root of your repository, running the application is simple:

1. **Ensure all requirements are installed** (including `streamlit`):

   ```
   pip install -r requirements.txt
   pip install streamlit
   ```
2. **Run the Streamlit command** from your terminal in the repository root directory:

   ```
   streamlit run app.py
   ```
3. **View the Dashboard:** Streamlit will automatically open the application in your web browser, typically at `http://localhost:8501`.

- You can now interactively change the stock ticker and dates to instantly update the plots and key metrics!

- This concludes the hands-on section of the repository. Congratulations!

About This Project

This **bonus notebook** is part 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`

Bonus Achievement Unlocked!

You've completed the bonus section and learned how to:

- Transform analytical notebooks into interactive web applications
- Create professional financial dashboards with Streamlit
- Build user-friendly interfaces for complex data analysis
- Deploy real-time technical analysis tools

Author

Prakash Ukhalkar

GitHub prakash-ukhalkar

Built with ❤️ for the Python community