Volatility checker

MOAKAL

August 3, 2023

Abstract

During an internship at an asset management firm, I developed an interest in market volatility and its correlation with global events. To enhance my Python skills, I have chosen to create a program that scrapes historical market data and corresponding news headlines as a learning tool. During the internship, I was introduced to the Streamlit Python module, which I will utilise as the framework for the tool.

Contents

1 Scope **1**
- 1.1 Preface . . . 1
- 1.2 Features & functionality . . . 2
- 1.3 Data sources . . . 2
- 1.4 Technology stack . . . 2
- 1.5 Deliverables . . . 2
- 1.6 Constraints . . . 2
- 1.7 Future enhancements & maintenance . . . 2

2 Design **3**
- 2.1 Inputs . . . 3
- 2.2 Outputs . . . 3

3 Code **3**

4 Known issues **5**

1 Scope

1.1 Preface

This project develops a Python program using Streamlit framework to create a user-friendly tool for searching and analysing stock volatility. Leveraging stock market data, users can search and assess the volatility of different stocks.

1.2 Features & functionality

The program will include the following features and functionality

- Date range selection
- Historical news headlines
- Stock search
- Stock volatility analysis using standard deviation and pandas
- Visualisation using Matplotlib

1.3 Data sources

The program will use reliable financial data sources to fetch historical stock price data. Within the date range it will also scrape for news headlines.

1.4 Technology stack

The following frameworks will be utilised in the development of the program

- Python
- Streamlit
- Matplotlib
- Yahoo Finance
- Feedparser

1.5 Deliverables

The final deliverable will be a Python program with dependencies and instructions for smooth execution. It will offer an interactive web interface to search, analyse stock volatility, and retrieve news headlines within the specified time frames.

1.6 Constraints

The project scope is limited by data availability and reliability. The program handles reasonable search requests and a specific range of stock volatility analysis but does not provide real-time or intraday data. Historical data accuracy depends on the selected source.

1.7 Future enhancements & maintenance

The project scope allows for future enhancements, including advanced volatility indicators, machine learning algorithms for volatility prediction, additional data sources/financial APIs, and expanded customisation options for visualisations and analysis.

2 Design

2.1 Inputs

- Dates
- Stock code
- Additional search query

2.2 Outputs

- Graph of stock (date, closing price)
- Headlines from the time frame
- Streamlit interface
- Metrics

3 Code

```
import datetime
import feedparser
import matplotlib.pyplot as plt
import pandas as pd
import plotly.express as px
import pytz
import streamlit as st
import streamlit_pandas as sp
import streamlit.components.v1 as components
import yfinance as yf


# Set page configuration and style
st.set_page_config(
    page_title="Volatility checker",
    initial_sidebar_state="expanded",
    menu_items={
        'Report a bug': 'https://www.github.com/moakal',
        'About': 'https://www.github.com/moakal',
    }
)

with open('style.css') as f:
    st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=
    True)

# Sidebar inputs for stock, start date, end date, and further news
    queries
with st.sidebar:
    stock = st.text_input('Enter stock', 'BLK')
    startDate = st.date_input('Start date', datetime.date(2013, 1,
    1))
    endDate = st.date_input('End date', datetime.date(2023, 1, 1))
    query = st.text_input('News query', 'BlackRock')

# Fetch historical stock data using yfinance
data = yf.download(stock, startDate, endDate)
```

```
35
36 data['Returns'] = data['Close'].pct_change() * 100
37 volatility = data['Returns'].std()
38
39 highest = max(data['Close'])
40 lowest = min(data['Close'])
41
42 # Prettify table by formatting date
43 data.reset_index(inplace=True)
44 data['Date'] = data['Date'].dt.strftime('%Y-%m-%d')
45
46 # Create and display the stock price graph using Plotly
47 st.title(stock)
48 data['Adj Close'].plot()
49 graph = px.line(data, x=data['Date'], y=data['Close'])
50 st.plotly_chart(graph)
51
52 # Display metrics in columns
53 col1, col2, col3 = st.columns(3)
54 col1.metric(label='Volatility*', value=round(volatility, 3))
55 col2.metric(label='Highest', value=round(highest, 3))
56 col3.metric(label='Lowest', value=round(lowest, 3))
57
58 # Table of historical stock data
59 with st.expander('View table'):
60     st.table(data)
61
62 # Function to scrape news related to the stock
63 def scrapeNews(stock, startDate, endDate):
64     url = f"https://news.google.com/rss/search?q={stock}&hl=en-US&
       gl=US&ceid=US:en"
65
66     feed = feedparser.parse(url)
67     entries = feed.entries
68
69     headlines = []
70     dates = []
71     links = []
72     timezone = pytz.timezone('UTC')  # Set the timezone to match
       the timestamps from the news articles
73
74     for entry in entries:
75         timestamp = pd.to_datetime(entry.published).tz_convert(
       timezone)
76         if startDate <= timestamp.tz_localize(None) <= endDate:
77             headlines.append(entry.title)
78             dates.append(timestamp.date())
79             links.append(entry.link)
80
81     data = {'Date': dates, 'Headline': headlines, 'Link': links}
82     df = pd.DataFrame(data)
83     if not df.empty:
84         df = df.sort_values(by='Date', ascending=False).reset_index
       (drop=True)
85
86     return df
87
88 # Load and display news on button click
89 if st.button('Load news'):
90     news = scrapeNews(stock, startDate, endDate)
91     newsq = scrapeNews(query, startDate, endDate)
92
```

```
93      news['Headline'] = news.apply(lambda row: f'<a href="{row["Link
        "]}" target="_blank">&#128279;</a> {row["Headline"]}', axis=1)
94      newsq['Headline'] = newsq.apply(lambda row: f'<a href="{row["
        Link"]}" target="_blank">&#128279;</a> {row["Headline"]}', axis
        =1)
95      st.subheader(stock + ' query')
96      st.write(news.drop(columns=['Link']).to_html(escape=False,
        index=False), unsafe_allow_html=True)
97      st.write('')
98      st.subheader('News query')
99      st.write(newsq.drop(columns=['Link']).to_html(escape=False,
        index=False), unsafe_allow_html=True)
100
101 st.write('\* Volatility calculated using standard deviation of
        returns')
```

4 Known issues

- News functionality does not work with stock code with longer length than 3 characters

Links

- github.com/moakal