Trend indicator MAs

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

What are technical indicators?

- Mathematical calculations based on historical market data
- Assume the market is efficient and the price has incorporated all public information
- Help traders to gain insight into past price patterns

Types of indicators

- **Trend indicators:** measure the direction or strength of a trend
  - Example: Moving averages (MA), Average Directional Movement Index (ADX)
- **Momentum indicators**: measure the velocity of price movement
  - Example: Relative Strength Index (RSI)
- **Volatility indicators**: measure the magnitude of price deviations
  - Example: Bollinger Bands

The TA-Lib package

TA-Lib : Technical Analysis Library

- Includes 150+ technical indicator implementations

```
import talib
```

Moving average indicators

- SMA: Simple Moving average
- EMA: Exponential Moving Average

- Characteristics:
  - Move with the price
  - Smooth out the data to better indicate the price direction

Simple Moving Average (SMA)

$$SMA = (P_1 + P_2 + ... + P_n)/n$$

```
# Calculate two SMAs
stock_data['SMA_short'] = talib.SMA(stock_data['Close'], timeperiod=10)
stock_data['SMA_long'] = talib.SMA(stock_data['Close'], timeperiod=50)
# Print the last five rows
print(stock_data.tail())
```

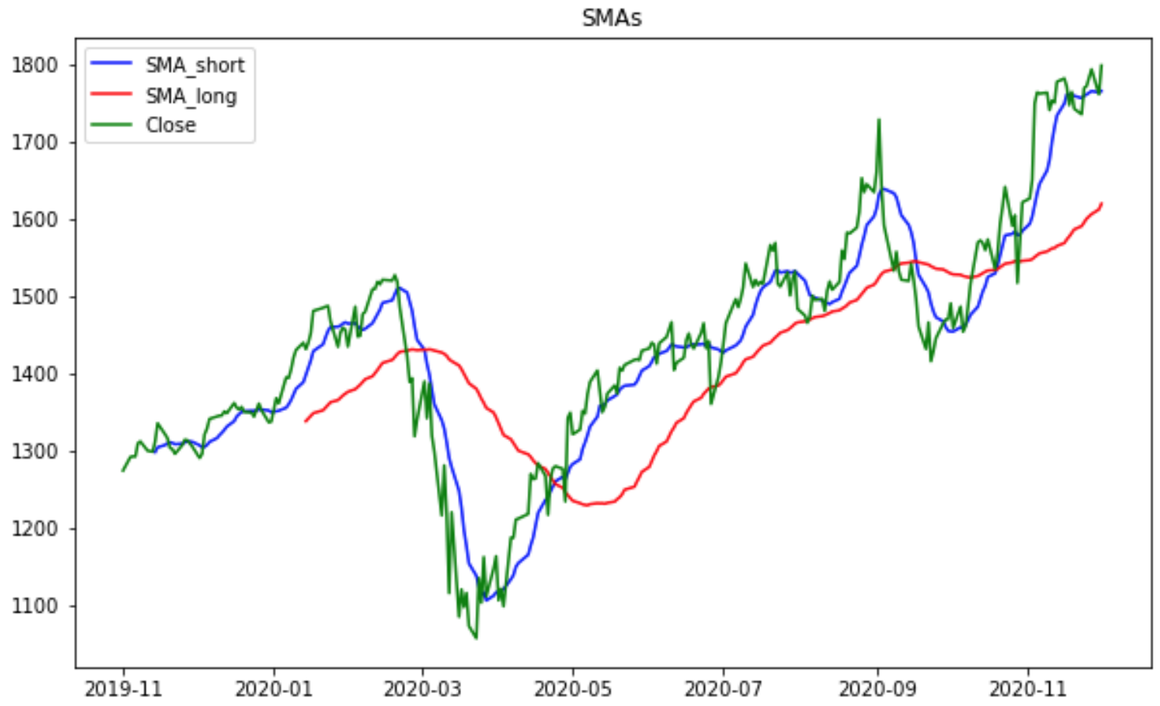
```
              Close  SMA_short  SMA_long
Date
2020-11-24 1768.88    1758.77   1594.74
2020-11-25 1771.43    1760.65   1599.75
2020-11-27 1793.19    1764.98   1605.70
2020-11-30 1760.74    1763.35   1611.71
2020-12-01 1798.10    1765.02   1619.05
```

Plotting the SMA

```
import matplotlib.pyplot as plt

# Plot SMA with the price
plt.plot(stock_data['SMA_short'],
         label='SMA_short')
plt.plot(stock_data['SMA_long'],
         label='SMA_long')
plt.plot(stock_data['Close'],
         label='Close')

# Customize and show the plot
plt.legend()
plt.title('SMAs')
plt.show()
```



Exponential Moving Average (EMA)

$$EMA_n = P_n \times multiplier + \text{previous EMA} \times (1 - multiplier)$$

$$multiplier = 2/(n+1)$$

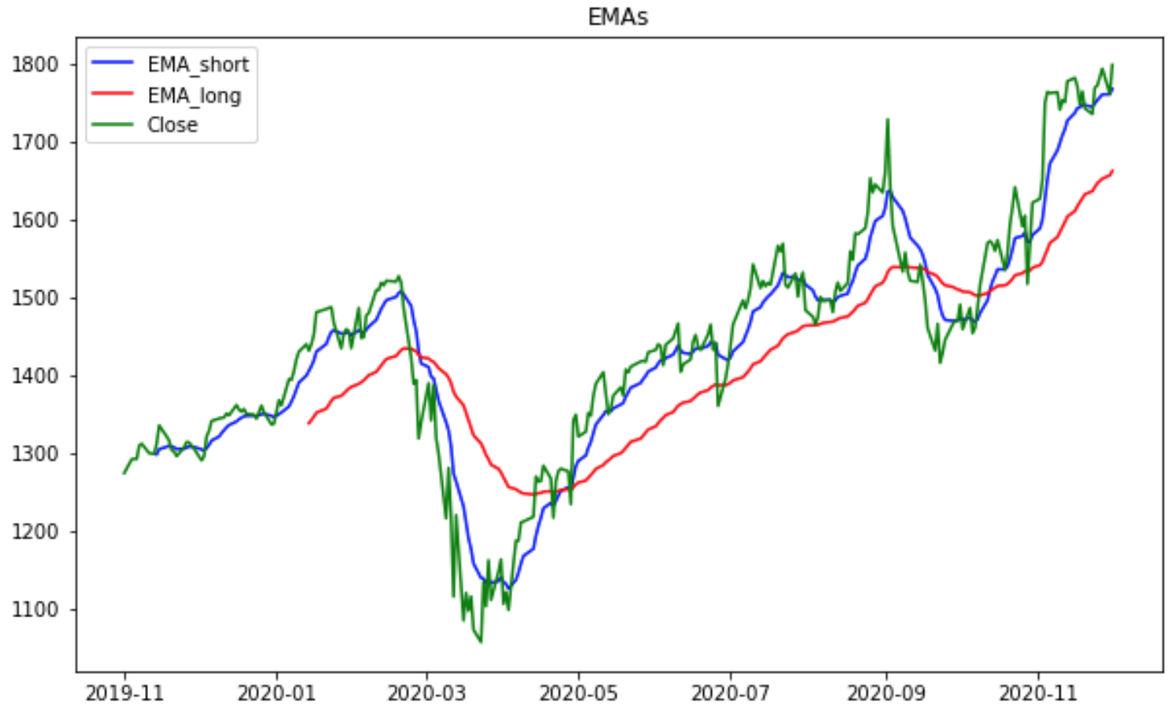
```
# Calculate two EMAs
stock_data['EMA_short'] = talib.EMA(stock_data['Close'], timeperiod=10)
stock_data['EMA_long'] = talib.EMA(stock_data['Close'], timeperiod=50)
# Print the last five rows
print(stock_data.tail())
```

```
             Close  EMA_short  EMA_long
Date
2020-11-24 1768.88    1748.65   1640.98
2020-11-25 1771.43    1752.79   1646.09
2020-11-27 1793.19    1760.13   1651.86
2020-11-30 1760.74    1760.24   1656.13
2020-12-01 1798.10    1767.13   1661.70
```

Plotting the EMA

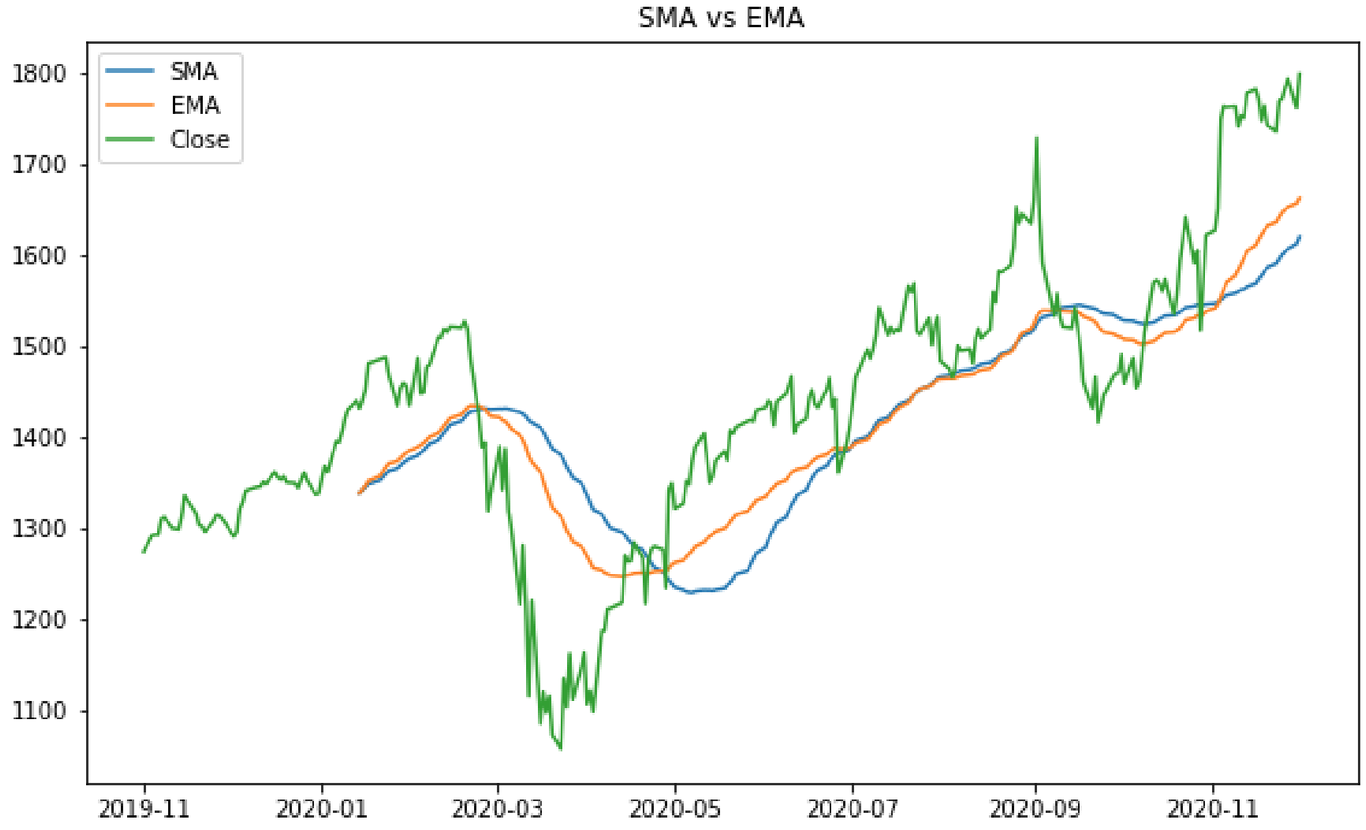
```
import matplotlib.pyplot as plt
# Plot EMA with the price
plt.plot(stock_data['EMA_short'],
         label='EMA_short')
plt.plot(stock_data['EMA_long'],
         label='EMA_long')
plt.plot(stock_data['Close'],
         label='Close')

# Customize and show the plot
plt.legend()
plt.title('EMAs')
plt.show()
```



SMA vs. EMA

EMA is more sensitive to the most recent price movement



Let's practice!

FINANCIAL TRADING IN PYTHON

Strength indicator: ADX

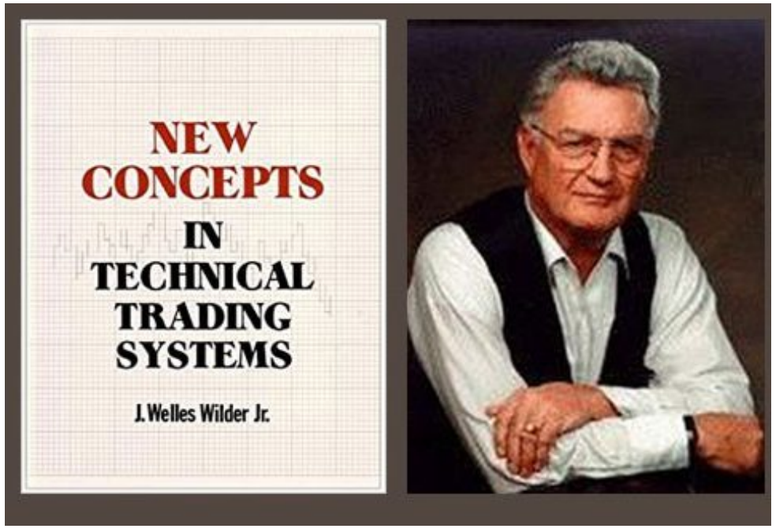
FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

What is ADX?

- Stands for "Average Directional Movement Index"
- Developed by J. Welles Wilder
  - "New Concepts in Technical Systems" (1987)



- Measures the strength of a trend
  - Oscillates between 0 and 100
  - ADX <= 25: no trend
  - ADX > 25: trending market
  - ADX > 50: strong trending market

How is ADX calculated?

- Derived from the smoothed averages of the difference between +DI and -DI
  - +DI (Plus Directional Index): quantify the presence of an uptrend
  - -DI (Minus Directional Index): quantify the presence of a downtrend

- Calculation input:
  - high, low, and close prices of each period

Implementing ADX in Python

```
# Calculate ADX
stock_data['ADX'] = talib.ADX(stock_data['High'], stock_data['Low'], stock_data['Close'],
                              timeperiod=14)
# Print the last five rows
print(stock_data.tail())
```

```
              Open   High    Low  Close   ADX
Date
2020-11-24 540.40 559.99 526.20 555.38 21.16
2020-11-25 550.06 574.00 545.37 574.00 23.92
2020-11-27 581.16 598.78 578.45 585.76 26.82
2020-11-30 602.21 607.80 554.51 567.60 28.25
2020-12-01 597.59 597.85 572.05 584.76 29.58
```

Plotting ADX

```
import matplotlib.pyplot as plt

# Create subplots
fig, (ax1, ax2) = plt.subplots(2)

# Plot ADX with the price
ax1.set_ylabel('Price')
ax1.plot(stock_data['Close'])
ax2.set_ylabel('ADX')
ax2.plot(stock_data['ADX'])

ax1.set_title('Price and ADX')
plt.show()
```

Let's practice!

FINANCIAL TRADING IN PYTHON

Momentum indicator: RSI

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

What is RSI?

- Stands for "Relative Strength Index"
- Developed by J. Welles Wilder
  - "New Concepts in Technical Systems" (1987)
- Measures the momentum of a trend
  - Oscillates between 0 and 100
  - RSI > 70: Overbought
  - RSI < 30: Oversold

How is RSI calculated?

$$RSI = 100 - 100/(1 + RS)$$

Where:

- RS: relative strength
- RS = average of upward price changes / average of downward price changes

Implementing RSI in Python

```
# Calculate RSI
stock_data['RSI'] = talib.RSI(stock_data['Close'], timeperiod=14)
# Print the last five rows
print(stock_data.tail())
```

```
              Open    High     Low   Close   RSI
Date
2020-11-24 1730.50 1771.60 1727.69 1768.88 62.78
2020-11-25 1772.89 1778.54 1756.54 1771.43 63.10
2020-11-27 1773.09 1804.00 1772.44 1793.19 65.81
2020-11-30 1781.18 1788.06 1755.00 1760.74 58.87
2020-12-01 1774.37 1824.83 1769.37 1798.10 63.63
```

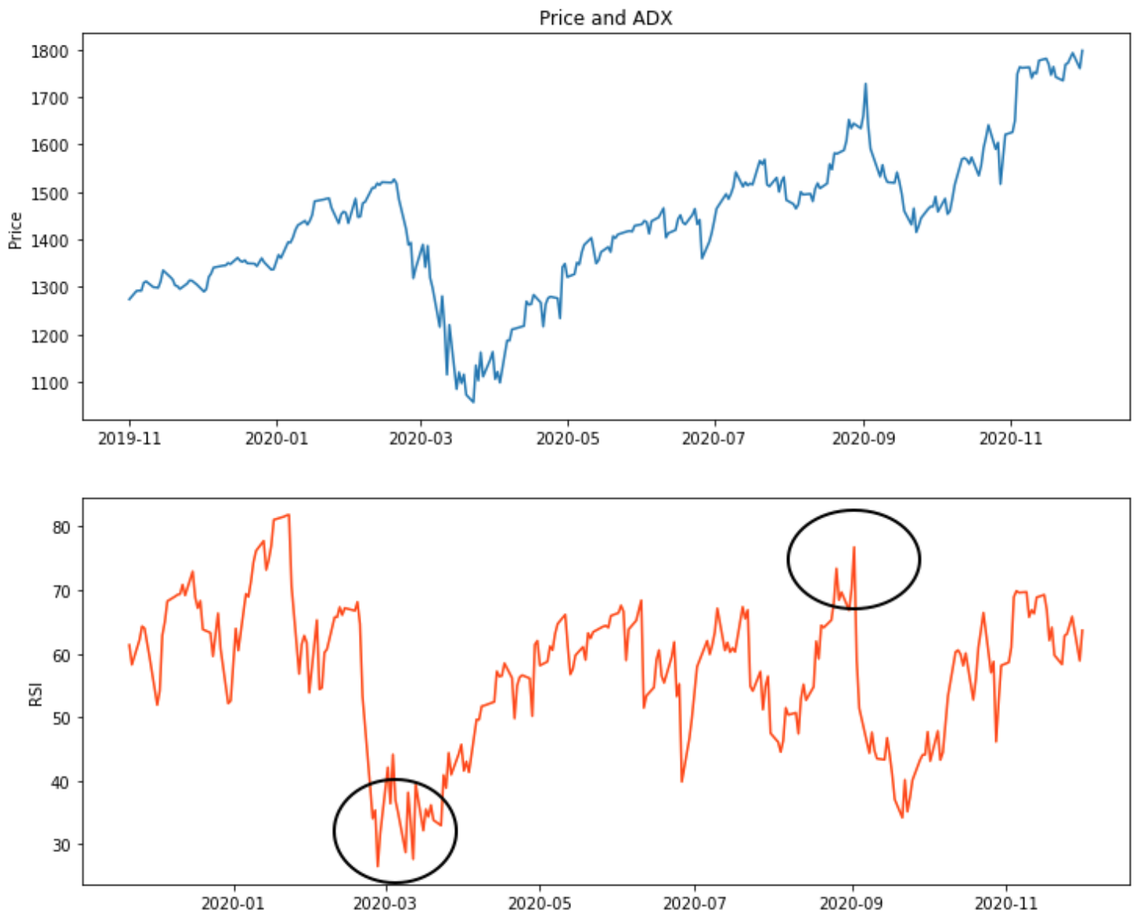
Plotting RSI

```
import matplotlib.pyplot as plt

# Create subplots
fig, (ax1, ax2) = plt.subplots(2)

# Plot RSI with the price
ax1.set_ylabel('Price')
ax1.plot(stock_data['Close'])
ax2.set_ylabel('RSI')
ax2.plot(stock_data['RSI'])

ax1.set_title('Price and RSI')
plt.show()
```



Let's practice!

FINANCIAL TRADING IN PYTHON

Volatility indicator: Bollinger Bands

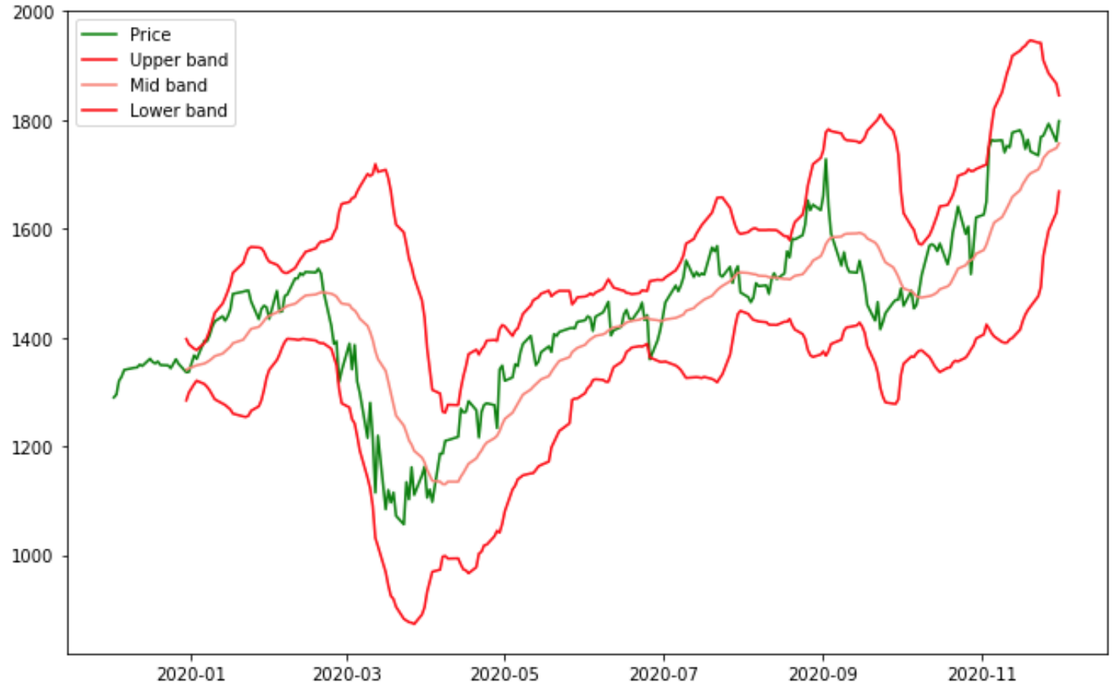
FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

What are Bollinger Bands?

- Developed by John Bollinger
  - "Bollinger on Bollinger Bands"



- Measure price volatility
- Composed of three lines:
  - Middle band: n-period simple moving average
  - Upper band: k-standard deviations above the middle band
  - Lower band: k-standard deviations below the middle band

Bollinger Bands implications

- The wider the bands, the more volatile the asset prices.

- Measure whether a price is too high or too low on a relative basis:
  - Relatively high: price close to the upper band
  - Relatively low: price close to the lower band

Implementing Bollinger Bands in Python

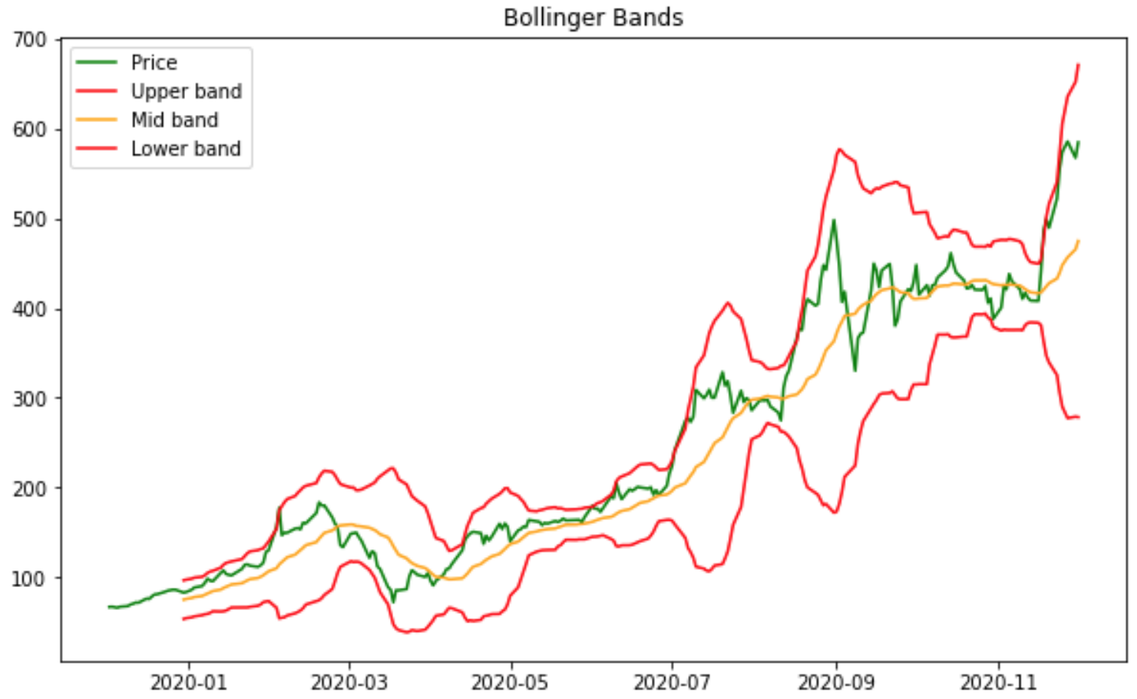
```
# Define the Bollinger Bands
upper, mid, lower = talib.BBANDS(stock_data['Close'],
                                 nbdevup=2,
                                 nbdevdn=2,
                                 timeperiod=20)
```

Plotting Bollinger Bands

```
import matplotlib.pyplot as plt

# Plot the Bollinger Bands
plt.plot(stock_data['Close'], label='Price')
plt.plot(upper, label="Upper band")
plt.plot(mid, label='Middle band')
plt.plot(lower, label='Lower band')

# Customize and show the plot
plt.title('Bollinger Bands')
plt.legend()
plt.show()
```



Let's practice!

FINANCIAL TRADING IN PYTHON