Published in Analytics Vidhya

Rohan Kumar Follow

May 22, 2021 · 7 min read · Listen

Save

How to Build Stock Technical Indicators with Python

Hello again! I am back with another important article on stock analysis, I know ya'll are enjoying my stock analysis series will be posting more next week so be sure to follow me! Now here we'll be analyzing MSFT stock in Python, calculating some Trading Indicators.

For a more complex analysis check this article out

Python for Stock Market :

Python for stock analysis

In this project, we'll analyse data from the stock market.

medium.com

Stock technical indicators are indispensable in stock analysis. They are calculated by a different mathematical formula based on the historical stock prices. In **algorithmic trading**, technical indicators are also essential to form a trading signal that can trigger the opening and closing of a trade by a trading robot.

In this article, I am going to show how we can use a Python library, **TA-Lib**, to build some popular technical indicators with few lines of codes. There will be three main groups of technical indicators presented here:

1. **Trend indicators —** Simple Moving Average(SMA), Exponential Moving Average (EMA) and Average Directional Movement Index (ADX)
2. **Momentum indicators —** Moving Average Convergence Divergence (MACD) and Relative Strength Index (RSI)
3. **Volatility indicators —** Bollinger Bands

install it with pip check out my article: -

How to install TA-LIB in Python

Here, I will help to install TA-LIB on your PC/Laptop. It should be easy going for y'all!

rohan09.medium.com

Here we'll be taking the data from the year 2020 and lets see how our indicators perform. I'll be using YFINANCE for downloading the data, you can install YFINANCE by: —

```
pip install yfinance
```

Let's download the data now! I'll be use Microsoft's data for 2020 you guys can play around with the other companies and indices.

```
import yfinance as yf

msft = yf.download("MSFT", start="2020-01-01", end="2020-12-31",
interval="1d")
```

```
msft
```

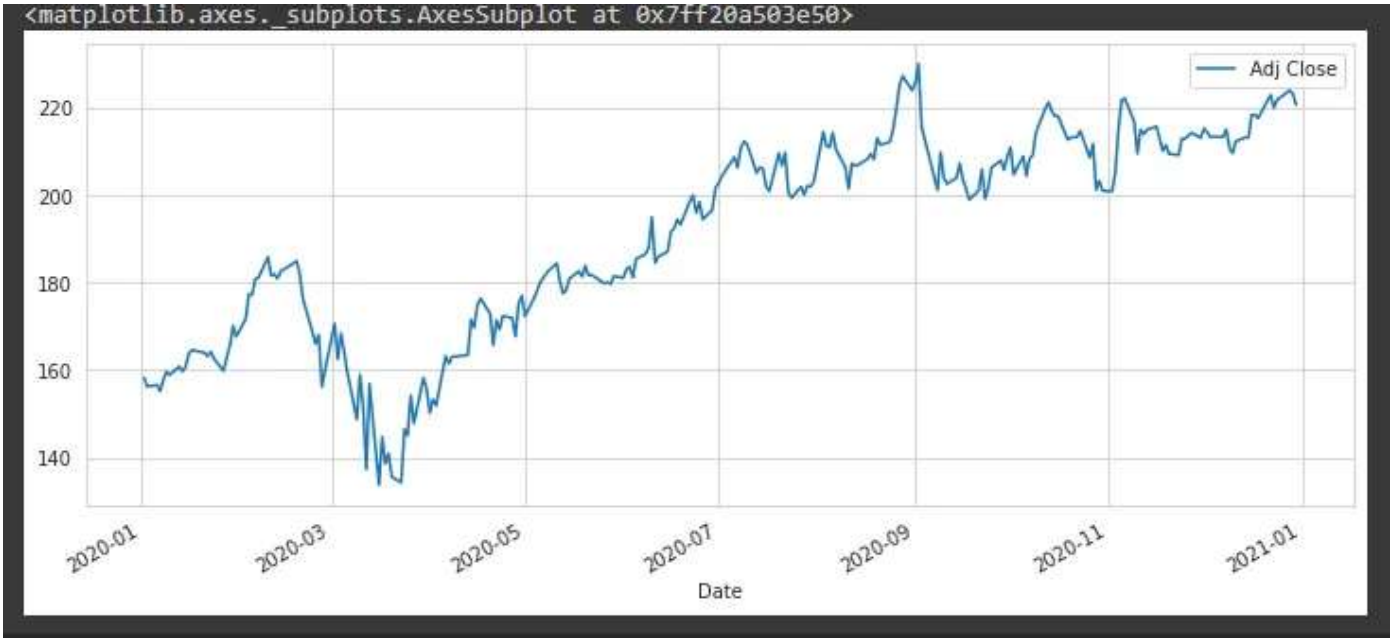
Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	158.779999	160.729996	158.330002	160.619995	158.205765	22622100
2020-01-03	158.320007	159.949997	158.059998	158.619995	156.235825	21116200
2020-01-06	157.080002	159.100006	156.509995	159.029999	156.639694	20813700
2020-01-07	159.320007	159.669998	157.320007	157.580002	155.211456	21634100
2020-01-08	158.929993	160.800003	157.949997	160.089996	157.683731	27746500
...	...	...	...	...	...	...
2020-12-23	223.110001	223.559998	220.000000	221.020004	220.004105	18699600
2020-12-24	221.419998	223.610001	221.199997	222.750000	221.726166	10550600
2020-12-28	224.449997	226.029999	223.020004	224.960007	223.925995	17933500
2020-12-29	226.309998	227.179993	223.580002	224.149994	223.119720	17403200
2020-12-30	225.229996	225.630005	221.470001	221.679993	220.661072	20272300

252 rows × 6 columns

MSFT historical data 2020

Come on! let's plot this!!

```
msft['Adj Close'].plot(legend=True,figsize=(12,5))
```



MSFT price plot

Trending indicators

Trend indicators are basically a measurement of the direction of a stock price trend (upwards, downwards or sideways). Let's started with the most basic one, simple moving average (SMA).

Simple Moving Average (SMA)

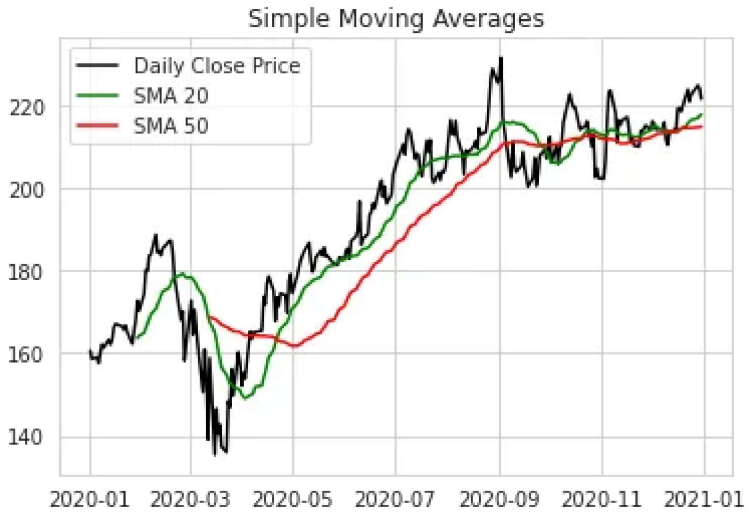
A simple moving average (SMA) is a rolling mean of the recent prices over a specific number of time periods (e.g. 10 days, 20 days, 50 days etc). SMA moves with the price and it can smooth out the daily price to show the price direction.

Let us use *talib SMA* command to build SMA indicators for 20 days and 50 days time frames.

```
msft['SMA20'] = talib.SMA(msft['Close'], timeperiod=20)
msft['SMA50'] = talib.SMA(msft['Close'], timeperiod=50)

plt.plot(msft['Close'], color='black', label='Daily Close Price')
plt.plot(msft['SMA20'], color='green', label='SMA 20')
plt.plot(msft['SMA50'], color='red', label='SMA 50')
plt.legend()
plt.title('Simple Moving Averages')
plt.show()
```

msft_sma.py hosted with ❤ by GitHub view raw



MSFT SMA PLOT

The shorter timeframe will result in an SMA that is more sensitive to a change in the price of the underlying stock and the longer timeframe will be relatively less sensitive.

You can play around with big timeperiods like 50 & 100 or 50 & 200

Exponential Moving Average (EMA)

Exponential moving average (EMA) is similar to SMA except that EMA gives a higher weight to the more recent prices while SMA assigns equal weight to all values. EMA is calculated based on the formula below:

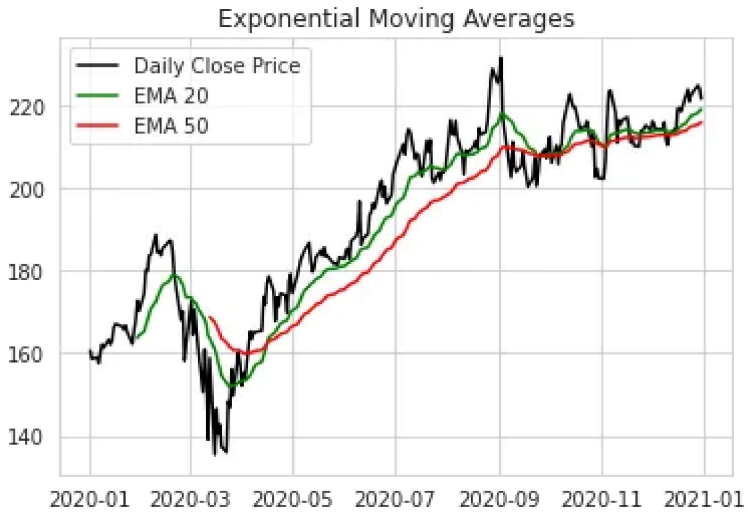
```
EMA = Closing Price * multiplier + EMA_previous_day * (1-multiplier)
```

Fortunately, the Python TA-Lib library offers us a one-liner command to perform the complex calculation.

```
msft['EMA20'] = talib.EMA(msft['Close'], timeperiod=20)
msft['EMA50'] = talib.EMA(msft['Close'], timeperiod=50)

plt.plot(msft['Close'], color='black', label='Daily Close Price')
plt.plot(msft['EMA20'], color='green', label='EMA 20')
plt.plot(msft['EMA50'], color='red', label='EMA 50')
plt.legend()
plt.title('Exponential Moving Averages')
plt.show()
```

msft_ema.py hosted with ❤ by GitHub view raw



Now if you compare EMA and SMA plots you'll see that EMA is more sensitive to the price movement due to the different weighting on the more recent prices.

MSFT EMA and MSFT SMA PLOTS

Average Directional Movement (ADX)

Average directional movement (ADX) is an indicator developed by J. Welles Wilder to measure the strength of a trend rather than just the price movement. The higher the magnitude of the ADX, the stronger the trend. The measurement of the ADX oscillates between 0 and 100 and it can be categorized into three range groups:

ADX ≤ 25: No trend

25 < ADX ≤ 50: Trending

ADX > 50: Strong Trending

We will use the TA-Lib *ADX* command to obtain the ADX values over the time series.

```
msft['ADX'] = talib.ADX(msft['High'], msft['Low'], msft['Close'], timeperiod=14)
fig, (ax1, ax2) = plt.subplots(2, sharex=True)
ax1.set_ylabel('Price')
ax1.plot(msft['Close'], color='black')
ax2.set_ylabel('ADX')
ax2.plot(msft['ADX'], color='blue')
ax1.set_title('Daily Close Price and ADX')
ax2.axhline(y = 50, color = 'r', linestyle = '-')
ax2.axhline(y = 25, color = 'r', linestyle = '-')
plt.show()
```

msft_adx.py hosted with ❤ by GitHub view raw

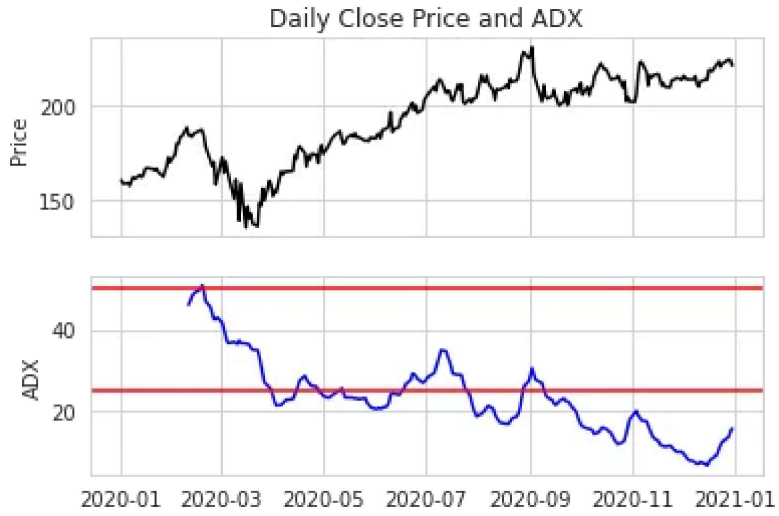
Let me explain some of the lines here:-

Line 1: TA-Lib *ADX* command requires high, low and close prices as its input. We set the time period to 14 days by following the industry standard.

Line 2: Here, we split the chart into two separate subplots to facilitate us to visualize the relationship between price movement and the ADX values over time.

Line 3–7: We plot the price movement on the first subplot and the ADX value on the second subplot.

Line 8–9: Use Matplotlib *axhline* command to create two horizontal lines overlayed on the ADX line plot. These two lines will help us to identify the high or low trend period from the plot.



From the ADX plot above, we can see an upward trend is formed when the ADX value stands between 25–50. When the ADX falls under 25, the sideways price movement is observed.

Now let's start building some momentum indicators!

Momentum Indicators

Momentum indicators are to measure the velocity of the price movement instead of a trend. This type of indicators offers a signal if the price movement is strengthening or weakening.

Moving Average Convergence Divergence (MACD)

Moving average convergence divergence (MACD) shows the relationship between two exponential moving averages (EMA) of a stock price. It is calculated by subtracting the EMA-26 from the EMA-12. This will result in a MACD line. Another EMA-9 of the MACD will also be plotted on top of the MACD line as the signal line to trigger a buy and sell action.

We can use *TA-Lib MACD* command to generate all the mentioned MACD and the signal line.

```python
macd, macdsig = talib.MACD(msft['Close'], fastperiod=12, slowperiod=26, signalperiod=9)
fig, (ax1, ax2) = plt.subplots(2, sharex=True)
ax1.set_ylabel('Price')
ax1.plot(msft['Close'], color='black')
ax2.set_ylabel('MACD')
ax2.plot(macdsig, color='green', label='Signal Line')
ax2.plot(macd, color='red', label='MACD')
ax1.set_title('Daily Close Price and MACD')
plt.legend()
plt.show()
```

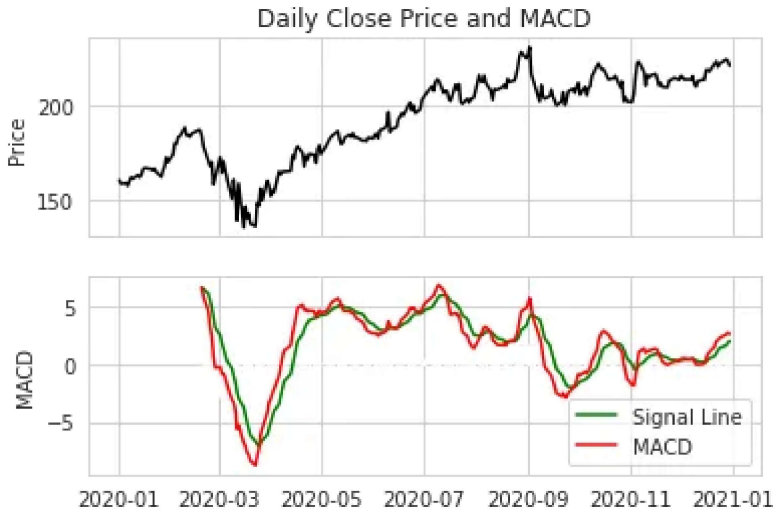
msft_macd.py hosted with ❤️ by GitHub view raw

TA-Lib MACD accepts four inputs which are the closing price, shorter EMA time frame (*fastperiod*), longer EMA time frame (*slowperiod*)and another EMA timeframe which is set to the *signalperiod*. Usually, we will adhere to the industry standard to use the following parameter setting:

fastperiod=12

showperiod=26

signalperiod=9



MSFT MACD PLOT

Relative Strength Index (RSI)

The Relative Strength Index (RSI) is also developed by J.Welles Wilder but it is used to measure the momentum of a trend by determining if a stock is overbought or oversold. There are two important benchmarks here:

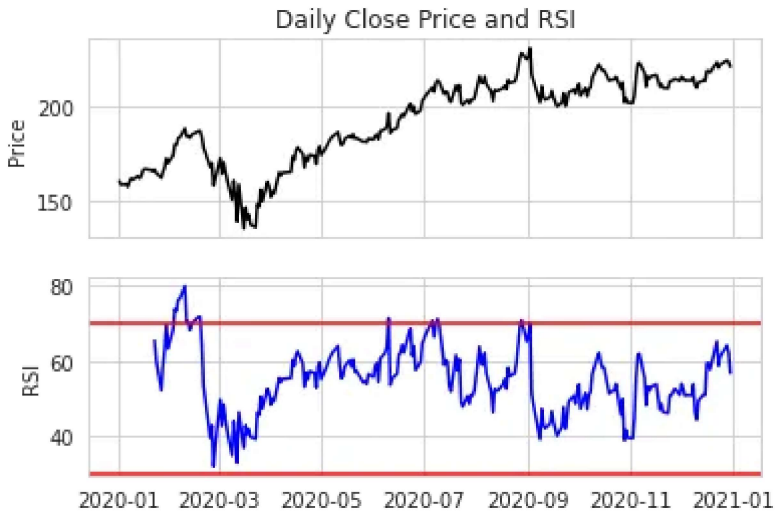
RSI > 70: Overbought

RSI < 30: Oversold

We are going to use *TA-Lib RSI* command to obtain the RSI values.

```
msft['RSI'] = talib.RSI(msft['Close'], timeperiod=14)
fig, (ax1, ax2) = plt.subplots(2, sharex=True)
ax1.set_ylabel('Price')
ax1.plot(msft['Close'],color = 'black')
ax2.set_ylabel('RSI')
ax2.plot(msft['RSI'], color='blue')
ax2.axhline(y = 70, color = 'r', linestyle = '-')
ax2.axhline(y = 30, color = 'r', linestyle = '-')
ax1.set_title('Daily Close Price and RSI')
plt.show()
```

msft_rsi.py hosted with ❤ by GitHub view raw



MSFT RSI PLOT

From the RSI plot, we can see the MSFT stock is generally overbought at the first half of the time period. An overbought condition (RSI > 70) shows that the stocks are being overvalued and may lead to a trend reversal in a bearish direction. On the opposite,

when there is an oversold condition happens (RSI < 30), the stock price is undervalued and may move up to form a bullish trend.

Let's build one Volatility Indicator last one in this article.

Volatility Indicator

The volatility indicator shows us the fluctuation of the price movement. It will look at the changes in market prices over a specified period of time. Here we will only look at one example, Bollinger Bands.

Bollinger Bands

A Bollinger Band is composed of three trendlines that form a middle band, upper band and lower band, respectively.

Middle band — A simple moving average (20 days by standard)

Upper band — 2 standard deviations above the middle band

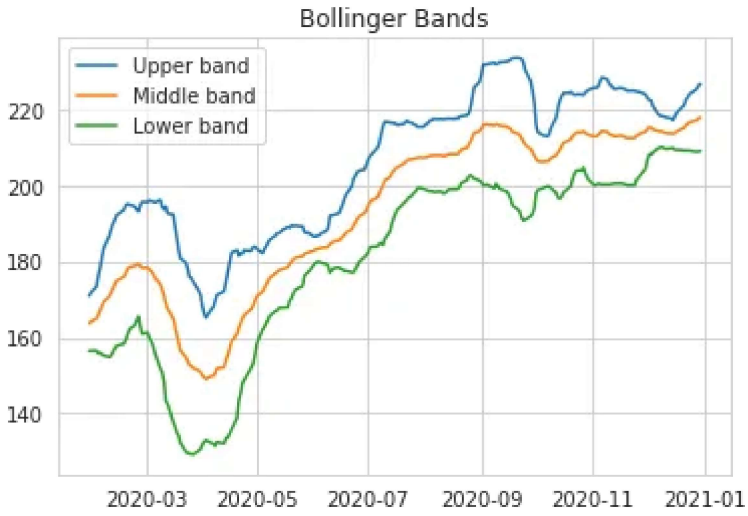
Lower band — 2 standard deviations below the middle band

The wider the distance between the upper and lower bands, the more volatile the stock prices. A price close to the upper band is considered relatively high. On another hand, when the price close to the lower band, it is viewed as relatively low.

We can use the TA-Lib *BBANDS* command to obtain the middle, upper and lower band values and plot them onto a graph.

```
upper, mid, lower = talib.BBANDS(msft['Close'], nbdevup=2, nbdevdn=2, timeperiod=20)
plt.plot(upper, label="Upper band")
plt.plot(mid, label='Middle band')
plt.plot(lower, label='Lower band')
plt.title('Bollinger Bands')
plt.legend()
plt.show()
```

msft_bollinger_bands.py hosted with ❤ by GitHub view raw



MSFT BOLLINGER BANDS PLOT

The Bollinger Bands above show us a clear picture of a more volatile period in the wider band area and a less volatile period in the narrower area.

Disclaimer There have been attempts to predict stock prices using time series analysis algorithms, though they still cannot be used to place bets in the real market. This is just a tutorial article that does not intent in any way to "direct" people into buying stocks.

How can we attempt to predict future stock behaviour?

Here I am providing you with a list of different models that can be used to predict stock prices.

Predicting Stock Prices Using Facebook's Prophet Model

Time-Series Forecasting: Predicting Microsoft (MSFT) Stock Prices Using ARIMA Model

Time-Series Forecasting: Predicting Apple Stock Price Using An LSTM Model

I recommend you to read through these articles, the models described are able to predict the prices with very good precision.

Now its your turn to clap and follow me. Thank you for reading!

Give me a FOLLOW if you liked this, for more tech blogs!

> "If at first you don't succeed, then skydiving isn't for you." — Mel Helitzer

Adios!

Stock Market Analytics Python Analysis Stocks