# 003 - Ploting data with Cufflinks

November 11, 2023

1 #003 Ploting data with Cufflinks

for future: - develope some about techinical analysis with that data

In this code will generate some graphical data as Candlestick plots, MACD, Bollinger Bands and other to improve analysis about one asset.

Implemented the same code from study #001, to obtain data from Yfinance, except for utilizing Cufflings to connect Pandas and Plotly for generate more useful info.

[1]:

```
#   !pip install pandas
#   !pip install pandas-datareader
#   !pip install yfinance
#   !pip install datetime
#   !pip install plotly_express
#   !pip install cufflinks
```

[2]:

```
#import libraries

import pandas as pd
from pandas_datareader import data as pdr
import numpy as np
import yfinance as yf
import datetime as dt
import plotly.express as px
import plotly.graph_objects as go
import cufflinks as cf

yf.pdr_override()
cf.go_offline() # Enabling offline mode for interactive data visualization␣
 ↪locally
```

1.1 3.1 Import, ajust and analyse DataFrame

[3]:

```
# Define the start and end dates, last 2 years
end = dt.datetime.now()
start = end - dt.timedelta(days = 365*2)
```

[4]:

```
# define Tickers
tk = input('Enter the ticker code: ')
```

```
Enter the ticker code: AAPL
```

[5]:

```
#obtain data from Yahoo Finance
df = pdr.get_data_yahoo(tk, start = start, end = end)
df
```

```
[*********************100%%**********************]  1 of 1 completed
```

[5]:

```
                  Open        High         Low       Close   Adj Close  \
Date
2021-11-11  148.960007  149.429993  147.679993  147.869995  146.199554
2021-11-12  148.429993  150.399994  147.479996  149.990005  148.295624
2021-11-15  150.369995  151.880005  149.429993  150.000000  148.305496
2021-11-16  149.940002  151.490005  149.339996  151.000000  149.294189
2021-11-17  151.000000  155.000000  150.990005  153.490005  151.756104
…                  …           …           …           …           …
2023-11-06  176.380005  179.429993  176.210007  179.229996  178.994186
2023-11-07  179.179993  182.440002  178.970001  181.820007  181.580780
2023-11-08  182.350006  183.449997  181.589996  182.889999  182.649368
2023-11-09  182.960007  184.119995  181.809998  182.410004  182.169998
2023-11-10  183.970001  186.570007  183.529999  186.399994  186.399994

              Volume
Date
2021-11-11  41000000
2021-11-12  63804000
2021-11-15  59222800
2021-11-16  59256200
2021-11-17  88807000
…                …
2023-11-06  63841300
2023-11-07  70530000
2023-11-08  49340300
2023-11-09  53763500
2023-11-10  66133400

[503 rows x 6 columns]
```

[6]:

```
#test for Null values on DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 503 entries, 2021-11-11 to 2023-11-10
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
```

```
 0   Open       503 non-null    float64
 1   High       503 non-null    float64
 2   Low        503 non-null    float64
 3   Close      503 non-null    float64
 4   Adj Close  503 non-null    float64
 5   Volume     503 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 27.5 KB
```

[7]:

```
#calc a daily percentage change for adjusted close price
df['Daily Return'] = df['Adj Close'].pct_change(1) * 100
df['Daily Return'].replace(np.nan, 0, inplace = True) #replace the first row,␣
 ↪changes null for 0

df
```

[7]:

```
                  Open        High         Low       Close   Adj Close  \
Date
2021-11-11  148.960007  149.429993  147.679993  147.869995  146.199554
2021-11-12  148.429993  150.399994  147.479996  149.990005  148.295624
2021-11-15  150.369995  151.880005  149.429993  150.000000  148.305496
2021-11-16  149.940002  151.490005  149.339996  151.000000  149.294189
2021-11-17  151.000000  155.000000  150.990005  153.490005  151.756104
…                  …           …           …          …           …
2023-11-06  176.380005  179.429993  176.210007  179.229996  178.994186
2023-11-07  179.179993  182.440002  178.970001  181.820007  181.580780
2023-11-08  182.350006  183.449997  181.589996  182.889999  182.649368
2023-11-09  182.960007  184.119995  181.809998  182.410004  182.169998
2023-11-10  183.970001  186.570007  183.529999  186.399994  186.399994

              Volume  Daily Return
Date
2021-11-11  41000000      0.000000
2021-11-12  63804000      1.433704
2021-11-15  59222800      0.006657
2021-11-16  59256200      0.666660
2021-11-17  88807000      1.649035
…                …            …
2023-11-06  63841300      1.460520
2023-11-07  70530000      1.445071
2023-11-08  49340300      0.588492
2023-11-09  53763500     -0.262454
2023-11-10  66133400      2.322005

[503 rows x 7 columns]
```

[8]:

```
df.describe().round(2)
```

```
[8]:          Open    High     Low   Close  Adj Close        Volume  Daily Return
     count  503.00  503.00  503.00  503.00     503.00  5.030000e+02        503.00
     mean   161.83  163.79  160.12  162.04     161.04  7.725599e+07          0.07
     std     16.22   16.00   16.39   16.19      16.29  2.694275e+07          1.88
     min    126.01  127.77  124.17  125.02     124.33  3.145820e+07         -5.87
     25%    148.86  150.76  147.26  149.30     148.28  5.695555e+07         -0.98
     50%    163.06  165.39  161.00  163.64     162.25  7.159840e+07          0.08
     75%    174.02  175.87  172.58  174.61     173.37  9.007030e+07          1.22
     max    196.24  198.23  195.28  196.45     195.93  1.954327e+08          8.90
```

1.2 3.2 Ploting results with Cufflinks

```
[21]: cf.set_config_file(theme='pearl', sharing='public', offline=True)
```

```
[22]: # Plot Candlestick figure using Cufflinks QuantFig module,
      figure = cf.QuantFig(df, title = tk + ' - Candlestick, RSI Chart', name =␣
       ↪tk,legend='top', rangeslider=False)
      figure.add_sma(periods =[14, 21], column = 'Close', color = ['magenta',␣
       ↪'green'])    # plot 14 and 21, Simple Moving Average
      figure.add_volume()
      figure.add_rsi(periods=20, color='java') #plot RSI, with close price and 20␣
       ↪periods
      figure.iplot(up_color = 'green', down_color = 'red')
```

```
[23]: figure = cf.QuantFig(df, title = tk + ' - Bollinger Bands, MACD Chart ', name =␣
       ↪tk, legend='top', rangeslider=False)
      figure.add_bollinger_bands(periods=20, boll_std=2, colors=['magenta','grey'],␣
       ↪fill=True)

      figure.add_volume()
      figure.add_macd()

      figure.iplot(up_color = 'green', down_color = 'red')
```

```
[31]: figure = cf.QuantFig(df, title= tk + ' - Another Quant Figures', legend='top',␣
       ↪name = tk)

      figure.add_adx(color = 'purple')     #Plot Average Directional Index    (ADX)
      #figure.add_cci(color = 'java')      #Plot Commodity Channel Indicator  (CCI)
      figure.add_dmi()                     #Plot Directional Movement Index   (DMI)
      figure.add_ema(color = 'magenta')    #Plot Exponencial Moving Average   (EMA)
      figure.add_sma(color = 'java')       #Plot Simple Moving Average        (SMA)

      figure.iplot(up_color = 'green', down_color = 'red')
```

```
[ ]:
```