003 - Ploting data with Cufflinks

November 28, 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

from pandas_datareader import data as pdr
import numpy as np
import yfinance as yf
import datetime as dt
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 10 years
end = dt.datetime.now()
start = end - dt.timedelta(days = 365*10)
```

[4]:
```
# define Tickers
tk = input('Enter the ticker code: ')
stock = yf.Ticker(tk)
```

```
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]:

Date	Open	High	Low	Close	Adj Close	Volume
2013-12-02	19.928572	20.154642	19.672144	19.686787	17.259239	472544800
2013-12-03	19.939285	20.227858	19.917143	20.225714	17.731710	450968000
2013-12-04	20.196428	20.328215	20.029285	20.178572	17.690390	377809600
2013-12-05	20.451786	20.540714	20.228930	20.282143	17.781179	447580000
2013-12-06	20.206785	20.241072	19.984644	20.000713	17.534451	344352400
…	…	…	…	…	…	…
2023-11-20	189.889999	191.910004	189.880005	191.449997	191.449997	46505100
2023-11-21	191.410004	191.520004	189.740005	190.639999	190.639999	38134500
2023-11-22	191.490005	192.929993	190.830002	191.309998	191.309998	39617700
2023-11-24	190.869995	190.899994	189.250000	189.970001	189.970001	24048300
2023-11-27	189.919998	190.669998	188.899994	189.789993	189.789993	40500500

```
[2514 rows x 6 columns]
```

[6]:
```
#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
```

[6]:

Date	Open	High	Low	Close	Adj Close
2013-12-02	19.928572	20.154642	19.672144	19.686787	17.259239

```
2013-12-03   19.939285   20.227858   19.917143   20.225714   17.731710
2013-12-04   20.196428   20.328215   20.029285   20.178572   17.690390
2013-12-05   20.451786   20.540714   20.228930   20.282143   17.781179
2013-12-06   20.206785   20.241072   19.984644   20.000713   17.534451
…                  …           …           …           …           …
2023-11-20  189.889999  191.910004  189.880005  191.449997  191.449997
2023-11-21  191.410004  191.520004  189.740005  190.639999  190.639999
2023-11-22  191.490005  192.929993  190.830002  191.309998  191.309998
2023-11-24  190.869995  190.899994  189.250000  189.970001  189.970001
2023-11-27  189.919998  190.669998  188.899994  189.789993  189.789993

                Volume  Daily Return
Date
2013-12-02  472544800      0.000000
2013-12-03  450968000      2.737497
2013-12-04  377809600     -0.233033
2013-12-05  447580000      0.513215
2013-12-06  344352400     -1.387585
…                 …            …
2023-11-20   46505100      0.927827
2023-11-21   38134500     -0.423086
2023-11-22   39617700      0.351447
2023-11-24   24048300     -0.700432
2023-11-27   40500500     -0.094756

[2514 rows x 7 columns]
```

[7]:
```
#test for Null values on DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2514 entries, 2013-12-02 to 2023-11-27
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          2514 non-null   float64
 1   High          2514 non-null   float64
 2   Low           2514 non-null   float64
 3   Close         2514 non-null   float64
 4   Adj Close     2514 non-null   float64
 5   Volume        2514 non-null   int64
 6   Daily Return  2514 non-null   float64
dtypes: float64(6), int64(1)
memory usage: 157.1 KB
```

[8]:
```
df.describe().round(2)
```

[8]:
```
              Open         High          Low        Close    Adj Close  \
count  2514.000000  2514.000000  2514.000000  2514.000000  2514.000000
mean     76.310000    77.150000    75.530000    76.380000    74.420000
std      55.320000    55.970000    54.730000    55.380000    55.860000
min      17.680000    17.910000    17.630000    17.850000    15.650000
25%      29.430000    29.720000    29.190000    29.460000    27.030000
50%      47.500000    47.950000    47.170000    47.580000    45.490000
75%     132.950000   134.390000   131.390000   132.740000   131.200000
max     196.240000   198.230000   195.280000   196.450000   195.930000

              Volume  
count  2.514000e+03  
mean   1.391076e+08  
std    8.613531e+07  
min    2.404830e+07  
25%    8.247592e+07  
50%    1.135670e+08  
75%    1.692130e+08  
max    1.065523e+09  

       Daily Return
count       2514.00
mean           0.11
std            1.79
min          -12.86
25%           -0.72
50%            0.09
75%            1.02
max           11.98
```

1.2 3.2 Ploting results with Cufflinks

[9]:
```
cf.set_config_file(theme='pearl', sharing='public', offline=True)
```

[10]:
```
# 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')
```

[11]:
```
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')
```

[12]:
```
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')
```