Bitcoin Trading Strategy

January 29, 2021

1 Bitcoin Trading Strategy

1.1 Import modules and data

[1]:

```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter, MaxNLocator
```

[2]:

```
#Yahoo Ticker of Bitcoin
ticker = ['BTC-USD']
```

[3]:

```
#We have the data for two months
ohlcv = pd.DataFrame()
ohlcv = yf.download(ticker, start='2020-12-1', interval='15m')
ohlcv
```

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

[3]:

```
                                   Open          High           Low  \
Datetime
2020-11-30 23:00:00+00:00  19461.968750  19461.968750  19428.796875
2020-11-30 23:15:00+00:00  19449.361328  19542.371094  19448.765625
2020-11-30 23:30:00+00:00  19581.863281  19633.156250  19581.863281
2020-11-30 23:45:00+00:00  19644.591797  19696.324219  19627.376953
2020-12-01 00:00:00+00:00  19633.769531  19652.005859  19508.767578
…                                   …           …           …
2021-01-29 14:00:00+00:00  38347.859375  38347.859375  37905.632812
2021-01-29 14:15:00+00:00  37989.000000  38063.734375  37912.335938
2021-01-29 14:30:00+00:00  38027.222656  38068.984375  37094.570312
2021-01-29 14:45:00+00:00  37017.703125  37017.703125  36779.867188
2021-01-29 14:55:02+00:00  37012.226562  37012.226562  37012.226562

                                  Close     Adj Close      Volume
Datetime
2020-11-30 23:00:00+00:00  19435.390625  19435.390625    18898944
2020-11-30 23:15:00+00:00  19542.371094  19542.371094   347623424
```

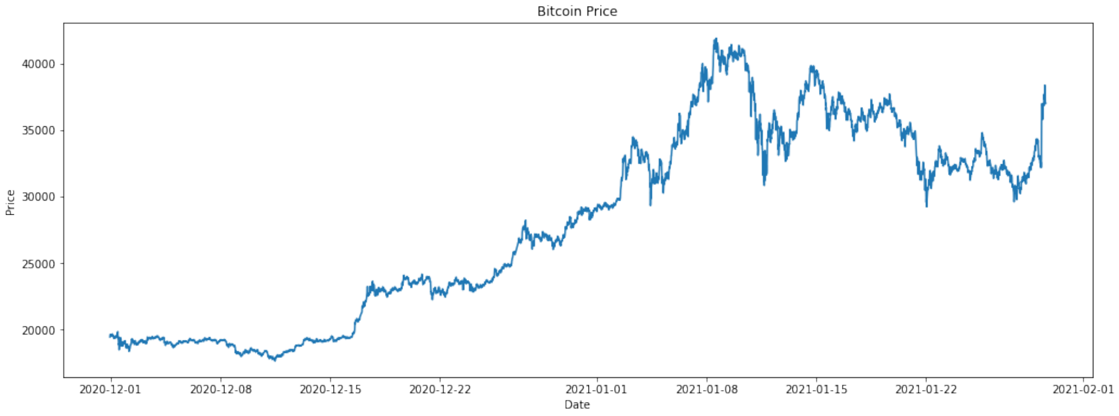
```
2020-11-30 23:30:00+00:00  19633.156250  19633.156250   437739520
2020-11-30 23:45:00+00:00  19627.376953  19627.376953   386248704
2020-12-01 00:00:00+00:00  19548.341797  19548.341797    94183424
…                                     …             …           …
2021-01-29 14:00:00+00:00  37983.972656  37983.972656   952721408
2021-01-29 14:15:00+00:00  38022.714844  38022.714844  1100570624
2021-01-29 14:30:00+00:00  37094.570312  37094.570312  3122028544
2021-01-29 14:45:00+00:00  36970.410156  36970.410156  1523105792
2021-01-29 14:55:02+00:00  37012.226562  37012.226562           0

[5674 rows x 6 columns]
```

[4]:

```python
#This function allows to display only a specific number of labels
def format_fn(tick_val, tick_pos):
    if int(tick_val) in xs:
        return labels[int(tick_val)]
    else:
        return ''
```

[5]:

```python
#We display the price of the underlying for two months
#I use a temporary dataframe in order to use the Dates as xlabels

temp_dir = ohlcv.reset_index()

fig, ax = plt.subplots(figsize=(17, 6))

xs= temp_dir["Datetime"]
xy= temp_dir["Close"]

ax.plot(xs, xy)

plt.title("Bitcoin Price")
plt.ylabel("Price")
plt.xlabel("Date")

plt.show()
```



2 Strategy

2.1 Creation of the technical indicators

[6]:
```python
def ATR(DF,n):
    #function to calculate True Range and Average True Range
    df = DF.copy()
    df['H-L']=abs(df['High']-df['Low'])
    df['H-PC']=abs(df['High']-df['Close'].shift(1))
    df['L-PC']=abs(df['Low']-df['Close'].shift(1))
    df['TR']=df[['H-L','H-PC','L-PC']].max(axis=1,skipna=False)
    df['ATR'] = df['TR'].rolling(n).mean()
    df2 = df.drop(['H-L','H-PC','L-PC'],axis=1)
    return df2['ATR']
```

[7]:
```python
#We add four columns to the dataframe, they will be the technical indicators␣
 ↪used for the backtesting
ohlcv_dict = ohlcv.copy()
tickers_signal = {}
tickers_ret = {}

ohlcv_dict["ATR"] = ATR(ohlcv_dict,5)
ohlcv_dict["roll_max_cp"] = ohlcv_dict["High"].rolling(20).max()
ohlcv_dict["roll_min_cp"] = ohlcv_dict["Low"].rolling(14).min()
ohlcv_dict["roll_max_vol"] = ohlcv_dict["Volume"].rolling(1).max()
tickers_signal = ""
tickers_ret = [0]
```

2.2 Backtesting

[8]:

```
#There is a Buy Signal if :
#   -The Max Price of the candle is the new max price since 20 periods
#   -The Volume is at least equal to the volume of the previous period

#There is a Sell Signal if :
#   -The Lower Price of the candle is the new lower price since 14 periods
#   -The Volume is at least equal to the volume of the previous period




for i in range(1,len(ohlcv_dict)):
    if tickers_signal == "":
            tickers_ret.append(0)
            if ohlcv_dict["High"][i]>=ohlcv_dict["roll_max_cp"][i] and \
               ohlcv_dict["Volume"][i]>1*ohlcv_dict["roll_max_vol"][i-1]:
                tickers_signal = "Buy"
            elif ohlcv_dict["Low"][i]<=ohlcv_dict["roll_min_cp"][i] and \
               ohlcv_dict["Volume"][i]>1*ohlcv_dict["roll_max_vol"][i-1]:
                tickers_signal = "Sell"

    elif tickers_signal == "Buy":
            if ohlcv_dict["Low"][i]<ohlcv_dict["Close"][i-1] -␣
 ↪ohlcv_dict["ATR"][i-1]:
                tickers_signal = ""
                tickers_ret.append(((ohlcv_dict["Close"][i-1] -␣
 ↪ohlcv_dict["ATR"][i-1])/ohlcv_dict["Close"][i-1])-1)
            elif ohlcv_dict["Low"][i]<=ohlcv_dict["roll_min_cp"][i] and \
               ohlcv_dict["Volume"][i]>1*ohlcv_dict["roll_max_vol"][i-1]:
                tickers_signal = "Sell"
                tickers_ret.append((ohlcv_dict["Close"][i]/
 ↪ohlcv_dict["Close"][i-1])-1)
            else:
                tickers_ret.append((ohlcv_dict["Close"][i]/
 ↪ohlcv_dict["Close"][i-1])-1)

    elif tickers_signal == "Sell":
            if ohlcv_dict["High"][i]>ohlcv_dict["Close"][i-1] +␣
 ↪ohlcv_dict["ATR"][i-1]:
                tickers_signal = ""
                tickers_ret.append((ohlcv_dict["Close"][i-1]/
 ↪(ohlcv_dict["Close"][i-1] + ohlcv_dict["ATR"][i-1]))-1)
            elif ohlcv_dict["High"][i]>=ohlcv_dict["roll_max_cp"][i] and \
               ohlcv_dict["Volume"][i]>1*ohlcv_dict["roll_max_vol"][i-1]:
                tickers_signal = "Buy"
                tickers_ret.append((ohlcv_dict["Close"][i-1]/
 ↪ohlcv_dict["Close"][i])-1)
            else:
                tickers_ret.append((ohlcv_dict["Close"][i-1]/
 ↪ohlcv_dict["Close"][i])-1)
```

[9]:

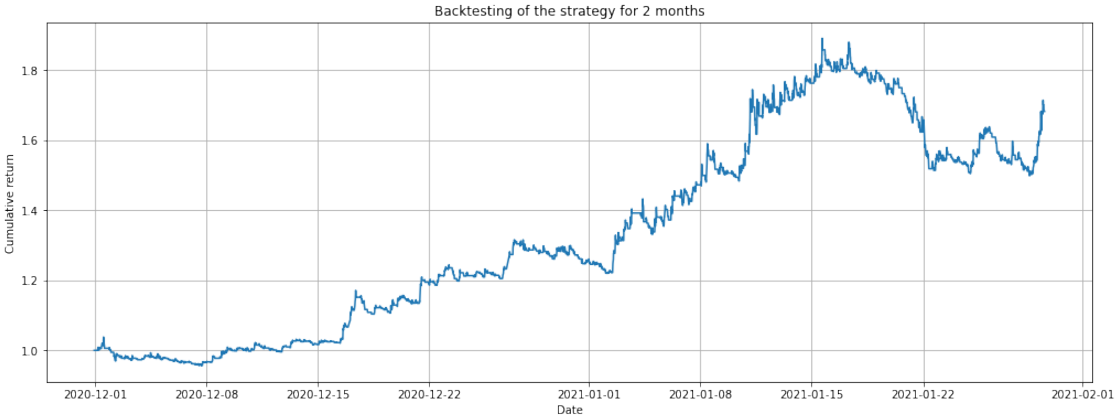
```
#We add a column with the returns and compute the cumulative returns
ohlcv_dict["ret"] = np.array(tickers_ret)
ohlcv_dict["cum_prod"]=(1+ohlcv_dict["ret"]).cumprod()
ohlcv_dict.tail(3)
```

[9]:

```
                                   Open          High           Low  \
Datetime
2021-01-29 14:30:00+00:00  38027.222656  38068.984375  37094.570312
2021-01-29 14:45:00+00:00  37017.703125  37017.703125  36779.867188
2021-01-29 14:55:02+00:00  37012.226562  37012.226562  37012.226562

                                  Close     Adj Close      Volume         ATR  \
Datetime
2021-01-29 14:30:00+00:00  37094.570312  37094.570312  3122028544  526.991406
2021-01-29 14:45:00+00:00  36970.410156  36970.410156  1523105792  534.317188
2021-01-29 14:55:02+00:00  37012.226562  37012.226562           0  389.746094

                            roll_max_cp   roll_min_cp  roll_max_vol  ret  \
Datetime
2021-01-29 14:30:00+00:00  38406.261719  36527.839844  3.122029e+09  0.0
2021-01-29 14:45:00+00:00  38406.261719  36668.519531  1.523106e+09  0.0
2021-01-29 14:55:02+00:00  38406.261719  36668.519531  0.000000e+00  0.0

                           cum_prod
Datetime
2021-01-29 14:30:00+00:00  1.682222
2021-01-29 14:45:00+00:00  1.682222
2021-01-29 14:55:02+00:00  1.682222
```

[10]:

```
#We display the cumulative return of the strategy for 2 months

fig, ax = plt.subplots(figsize=(17, 6))
xs= temp_dir["Datetime"]
xy= (ohlcv_dict["cum_prod"])

ax.plot(xs, xy)

plt.title("Backtesting of the strategy for 2 months")
plt.ylabel("Cumulative return")
plt.xlabel("Date")
plt.grid(b = True)

plt.show()
```



2.3 Annualised return and volatility

2.3.1 Compound Annual Growth Rate

[11]:

```
def CAGR(DF):
    # function to calculate the Cumulative Annual Growth Rate of a trading␣
 ↪strategy
    df = DF.copy()
    df["cum_return"] = (1 + df["ret"]).cumprod()
    n = len(df)/(365 * 96) #There is 96 subperiods in each day
    CAGR = (df["cum_return"].tolist()[-1])**(1/n) - 1
    return CAGR
```

[12]:

```
print ('CAGR = ' + str(round(CAGR(ohlcv_dict)*100, 2)) + '%' )
```

```
CAGR = 2382.85%
```

2.3.2 Annualised Volatility

[13]:

```
def volatility(DF):
    # function to calculate annualized volatility of a trading strategy
    df = DF.copy()
    vol = df["ret"].std() * np.sqrt(365*96)
    return vol
```

[14]:

```
print ('Volatility = ' + str(round(volatility(ohlcv_dict)*100, 2)) + '%' )
```

```
Volatility = 64.5%
```

2.3.3 Annualised Sharpe Ratio

[15]:

```
def sharpe(DF,rf):
    # function to calculate sharpe ratio ; rf is the risk free rate
    df = DF.copy()
    sr = (CAGR(df) - rf)/volatility(df)
    return sr
```

[16]:

```
print ('Sharpe Ratio = ' + str(round(sharpe(ohlcv_dict, 0.02), 2)))
```

```
Sharpe Ratio = 36.91
```

2.3.4 Max Drawdown over the period

[17]:

```
def max_dd(DF):
    # function to calculate max drawdown
    df = DF.copy()
    df["cum_return"] = (1 + df["ret"]).cumprod()
    df["cum_roll_max"] = df["cum_return"].cummax()
    df["drawdown"] = df["cum_roll_max"] - df["cum_return"]
    df["drawdown_pct"] = df["drawdown"]/df["cum_roll_max"]
    max_dd = df["drawdown_pct"].max()
    return max_dd
```

[18]:

```
print ('Max Drawdown = ' + str(round(max_dd(ohlcv_dict)*100, 2)) + '%')
```

```
Max Drawdown = 20.74%
```

2.3.5 Created by Pierre Marchand-Lentz and Emmanuel Zheng