TITLE : Performance Measurement of Strategies - KPI

Author : BRYAN LIM YUQIANG

Contact me :

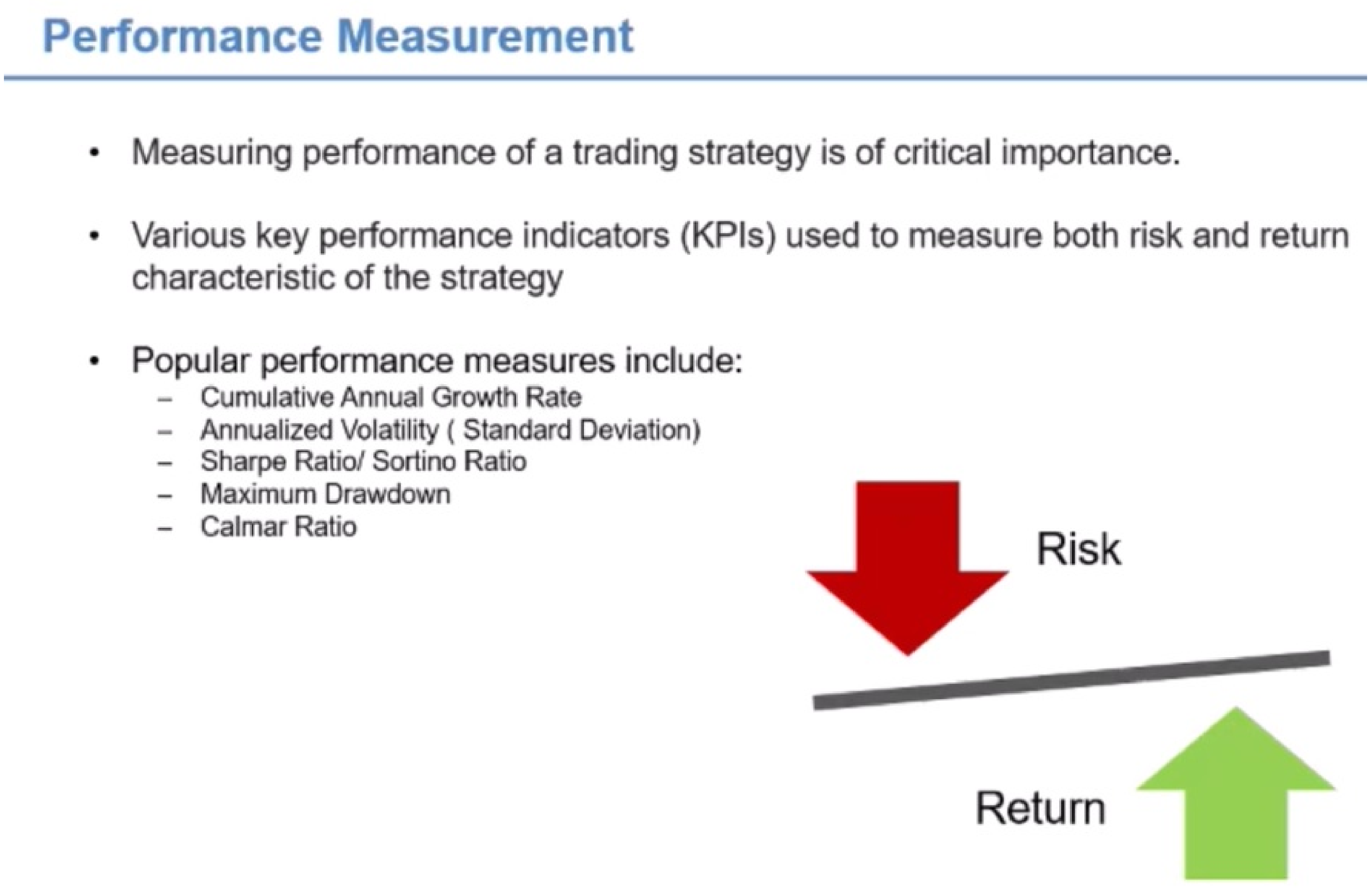
- Bryanlimyuqiang@gmail.com
- Linkedin (https://www.linkedin.com/in/bryanlimyuqiang/)
- Follow my trading journey (https://www.etoro.com/people/bryanlimyuqiang)

Covers:

1. Compounded Annual Growth Rate (CAGR)
2. Volatility
3. Sharpe and Sortino
4. Maximum Drawdown and Calmar Ratio

In [15]:

```
import pandas_datareader.data as pdr
import numpy as np
import datetime
```



1. COMPOUNDED ANNUAL GROWTH RATE (CAGR)

CAGR

- Compounded Annual Growth Rate is the annual rate of return realized by an asset/portfolio to reach its current market value from its initial value.
- CAGR calculation assumes profits are continuously reinvested.

$$CAGR = \left[\frac{\text{End value}}{\text{Beginning value}}\right]^{[1/\text{years}]} - 1$$

- Provides ease of comparison between different trading strategies.
- Does not reflect investment risk and therefore should always be used in conjunction with a volatility measure.

In [16]:

```
# We try to use the S&P500
ticker = '^GSPC'

# get_data_yahoo(ticker, start, end) , We use 5 years of Data
SnP = pdr.get_data_yahoo(ticker, datetime.date.today()- datetime.timedelta(1825), datetime.
```

In [17]:

```
def CAGR(DF):
    df = DF.copy()
    df['daily_ret'] = DF['Adj Close'].pct_change()

    # cum prod of daily_ret, use excel to visualise if unsure
    df['cum_ret'] = (1 + df['daily_ret']).cumprod()

    # get len of rows of data divide by 252 to get number of trading years!!
    n = len(df)/252

    # We only want the last value as it represents the end net ret of our strategy over the
    CAGR = df['cum_ret'][-1]**(1/n) - 1


    return CAGR
```

In [18]:

```
CAGR(SnP)
```

Out[18]:

```
0.06838149223270218
```

2. Annualised Volatility

Annualized Volatility

- Volatility of a strategy is represented by the standard deviation of the returns. This captures the variability of returns from the mean return.
- Annualization is achieved by multiplying volatility with square root of the annualization factor. For example :
  - To annualize daily volatility multiply with √252 – 252 trading days in a year
  - To annualize weekly volatility multiply with √52 – 52 trading weeks in a year
  - To annualize monthly volatility multiply with √12 – 12 trading month in a year
- Widely used measure of risk. However, this approach assumes normal distribution of returns which is not true.
- Does not capture tail risk.

In [19]:

```
def volatility(DF):
    df = DF.copy()
    df['daily_ret'] = df['Adj Close'].pct_change()

    #annualised volatility based of daily volatility
    vol = df['daily_ret'].std() * np.sqrt(252)

    return vol
```

In [21]:

```
volatility(SnP)
```

Out[21]:

```
0.18940117733974982
```

3. Sharpe And Sortino Ratios

$$\text{Sortino Ratio} = \frac{R_p - R_f}{\sigma_p}$$

R_p = Expected Return

R_f = Risk Free rate of return

σ_p = Standard Deviation of Negative Asset Returns

$$Sharpe\ ratio = \frac{\bar{r}_p - r_f}{\sigma_p}$$

$\bar{r}_p = expected\ return\ of\ the\ portfolio\ or\ investment$

$r_f = risk\ free\ rate$

$\sigma_p = standard\ deviation\ of\ portfolio\ returns$

In [25]:

```
# rf = risk-free rate, you may choose to use 10-year treasury yields etc
rf = 0.022

def sharpe(DF, rf):
    df = DF.copy()
    sr = (CAGR(df) - rf)/volatility(df)
    return sr
```

In [26]:

```
sharpe(SnP, rf)
```

Out[26]:

```
0.24488492037989062
```

In [28]:

```
def sortino(DF, rf):
    df = DF.copy()
    df['daily_ret'] = df['Adj Close'].pct_change()

    #filter only negative vol
    neg_vol = df[df['daily_ret'] < 0]['daily_ret'].std() * np.sqrt(252)
    sr = (CAGR(df) - rf) / neg_vol

    return sr
```

In [29]:

```
sortino(SnP, rf)
```

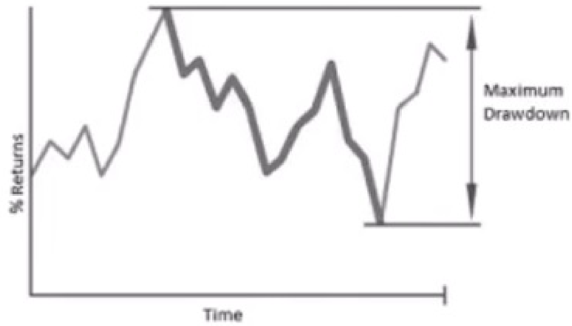
Out[29]:

```
0.2746330291554383
```

4. Maximum Drawdown & Calmar Ratio

Maximum Drawdown & Calmar Ratio

- Largest percentage drop in asset price over a specified time period (distance between peak and trough in the line curve of the asset)
- Investments with longer backtesting period will likely have larger max drawdown and therefore caution must be applied in comparing across strategies



- Calmar Ratio is the ratio of CAGR and Max drawdown and it's a measure of risk adjusted return

$$\boldsymbol{Calmar\ Ratio = \frac{Annualized(R_p)}{MaxDD_p}}$$

$$Annualized(R_p) - Annualized\ Portfolio\ Return$$

$$MaxDD_p - Portfolio\ Maximum\ Drawdown$$

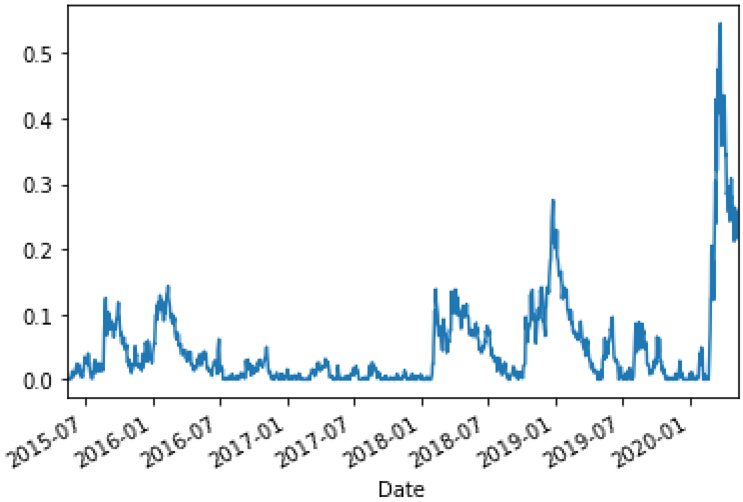
In [40]:

```
def max_drawdown(DF):
    df = DF.copy()
    df['daily_ret'] = df['Adj Close'].pct_change()
    df['cum_return'] = (1+ df['daily_ret']).cumprod()

    # Find highest peak in DF
    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 df
```

In [53]:

```
print("The max drawdown in the dataset was {:2f}".format(max_drawdown(SnP)['drawdown_pct'].
max_drawdown(SnP)['drawdown'].plot();
```

The max drawdown in the dataset was 0.339250



In [60]:

```
def calmar(DF):
    df = DF.copy()
    clmr  = CAGR(df)/(max_drawdown(df)['drawdown_pct'].max())
    return clmr
```

In [61]:

```
calmar(SnP)
```

Out[61]:

0.2015669059811831