Red White and Blue strategy used to leverage ETFs.

You got 6 short term EMAs in red and 6 long term EMAs in blue Identify the short term trend in an equity.

The entry points are when the red first crosses above the blue and when it breaks down, it's the sell signal.

Need to inumerate these entries and exists into a language that Python interprets and backtest.

check for a red-white-blue pattern or a blue-white-red pattern

based on that and a position availabe at the time, it will simulate entering and exiting our positions

at the end, print out summary statistics that identify how effective that strategy was.

In [5]:
```
import pandas as pd
import numpy as np
import yfinance as yf
import datetime as dt
from pandas_datareader import data as pdr
```

comment out the moving average and create a lot of exponential moving averages

In [ ]:
```
yf.pdr_override()
stock = input("Enter a stock ticker symbol: ")
# I used the AAPL as my stock ticker symbol.
print(stock)

startyear = 2019
startmonth = 1
startday = 1

# create a datetime object
start = dt.datetime(startyear, startmonth, startday)
# two date time objects, the start and the now
now = dt.datetime.now()

df = pdr.get_data_yahoo(stock, start, now)
print(df)

ma = 50
# str converts the ma which is an integer into a str and concatenates with the string "Sma_"
smaString = "Sma_"+str(ma)

# creates a new column to our data frame, but when cut out the first 4 rows
df[smaString] = df.iloc[:,4].rolling(window=ma).mean()
print(df)
```

In [8]:
```
# list full of expontial moving averages
# 3, 5, 8, 10, 12, 15 for short term EMAs
# 30, 35, 40, 45, 50, 60 for long term EMAs
emasUsed = [3, 5, 8, 10, 12, 15, 30, 35, 40, 45, 50, 60]

# create a for loop that goes through each period and create a different column in our data frame corresponding
for x in emasUsed:
    ema=x
    # making a column each time corresponding to each value
    # ewm making it exponential
    df["Ema_" + str(ema)]=round(df.iloc[:,4].ewm(span=ema, adjust=False).mean(), 2)
print(df.tail())
```

```
                 Open        High         Low       Close   Adj Close  \
Date
2023-11-29  190.899994  192.089996  188.970001  189.369995  189.369995
2023-11-30  189.839996  190.320007  188.190002  189.949997  189.949997
2023-12-01  190.330002  191.559998  189.229996  191.240005  191.240005
2023-12-04  189.979996  190.050003  187.449997  189.429993  189.429993
2023-12-05  190.210007  194.399994  190.210007  193.360001  193.360001

              Volume      Sma_50   Ema_3   Ema_5   Ema_8  Ema_10  Ema_12  \
Date
2023-11-29  43014200  178.645636  189.81  189.88  189.46  188.95  188.35
2023-11-30  48794400  178.939453  189.88  189.90  189.57  189.13  188.60
2023-12-01  45679300  179.290230  190.56  190.35  189.94  189.52  189.01
2023-12-04  43389500  179.587630  190.00  190.04  189.83  189.50  189.07
2023-12-05  49011979  179.937863  191.68  191.15  190.61  190.20  189.73

            Ema_15  Ema_30  Ema_35  Ema_40  Ema_45  Ema_50  Ema_60
Date
2023-11-29  187.42  183.87  183.16  182.61  182.18  181.83  181.30
2023-11-30  187.74  184.26  183.53  182.96  182.51  182.15  181.58
2023-12-01  188.17  184.71  183.96  183.37  182.89  182.51  181.90
2023-12-04  188.33  185.02  184.27  183.66  183.18  182.78  182.14
2023-12-05  188.96  185.56  184.77  184.14  183.62  183.19  182.51
```

In [ ]:
```
# Create a for loop and iterate each date and check if our entry set has been satisfied
# for i in df.index:
#    print(i)

for i in df.index:
# cmin corresponds to the minimum of the short term EMAs
# cmax corresponds to the maximum of the long term EMAs
# important b/c those are the critical values to compare to determine if we are in a red-white-blue pattern
    cmin = min(df["Ema_3"][i], df["Ema_5"][i], df["Ema_8"][i], df["Ema_10"][i], df["Ema_12"][i], df["Ema_15"][i
    cmax = min(df["Ema_30"][i], df["Ema_35"][i], df["Ema_40"][i], df["Ema_45"][i], df["Ema_50"][i], df["Ema_60"

# close is the closing values at that point
    close = df["Adj Close"][i]

    if (cmin>cmax):
        print("Red White Blue")
    elif(cmin<cmax):
        print("Blue White Red")

# will print if each date is either former or the latter
```

In [ ]:
```
# create some variables

# variable to determine if we are entering a position, 1 = enter, 0 = not enter
pos = 0

# variable helps us keep track of what row we are in
num = 0

# variable empty list to add the results of our trade in
percentchange=[]

# same code from the prior block
for i in df.index:
    cmin = min(df["Ema_3"][i], df["Ema_5"][i], df["Ema_8"][i], df["Ema_10"][i], df["Ema_12"][i], df["Ema_15"][i
    cmax = min(df["Ema_30"][i], df["Ema_35"][i], df["Ema_40"][i], df["Ema_45"][i], df["Ema_50"][i], df["Ema_60"
    close = df["Adj Close"][i]
    if (cmin>cmax):
        print("Red White Blue")
        if(pos==0):
            # bp = buy price
            # close is our adjusting close at this point
            bp=close
            # turning on our position
            pos=1
            # tells user we are buying now at this price
            print("Buying now at "+str(bp))

    elif(cmin<cmax):
        print("Blue White Red")
        if(pos==1):
            # sp = sell price
            sp = close
            # turning off our position
            pos = 0
            # tell user we are selling now at this price
            print("Selling now at "+str(sp))
            # pc = percent change
            pc = (sp/bp - 1) * 100
            # store pc to our list to analyze our trade
            percentchange.append(pc)

    # if we are at the end of our pandas data frame and we have a position open
    # keep track of where we are at in the pandas data frame
    if(num==df["Adj Close"].count()-1 and pos==1):
        # same code for elif(cmin<cmax)
        # simulate closing position
        sp = close
        pos = 0
        print("Selling now at "+str(sp))
        pc = (sp/bp - 1) * 100
        percentchange.append(pc)
    num+=1
print(percentchange)
# TIP : scroll all the way to the bottom
# calculate all these different trades
```

In [12]:
```
# part 1 is completed with the goal of calculating all these different trades
# part 2 is analyze the results and output a bunch of summary statistics to compare different strategies and se

# VARIABLES
gains=0
# ng = number of gains
ng=0
losses=0
# nl = number of losses
nl=0
totalR=1
```

```
# calculate the number of gains and losses, total gains and total losses, and total returns
for i in percentchange:
    # winning trade
    if(i>0):
        gains+=i
        ng+=1
    else:
        losses+=i
        nl+=1
    totalR = totalR*((i/100) + 1)

# multiples all the different percentages together and calculate the total return would be if you were going 10
totalR = round((totalR - 1)*100, 2)

# calcualte our average gains/losses, average wins-loss ratios
# calculate the number of gains
if(ng>0):
    avgGain = gains/ng
    maxR = str(max(percentchange))
else:
    avgGain=0
    maxR = "undefined"

# calculate the number of losses
if(ng>0):
    avgLoss = losses/nl
    maxL = str(max(percentchange))
    ratio = str(-avgGain/avgLoss)
else:
    avgLoss=0
    maxL = "undefined"
    ratio = "inf"

# calculate our batting average meaning that percentage of time a trade ends up with a gain
if(ng>0 or nl>0):
    battingAvg = ng/(ng+nl)
else:
    battingAvg = 0

# now we print it all nice for users to read and interpret
print()
print("Results for "+ stock +" going back to "+str(df.index[0])+", Sample size: "+str(ng+nl)+" trades")
print("EMAs used: "+str(emasUsed))
print("Batting Avg: "+ str(battingAvg))
print("Gain/loss ratio: "+ ratio)
print("Average Gain: "+ str(avgGain))
print("Average Loss: "+ str(avgLoss))
print("Max Return: "+ maxR)
print("Max Loss: "+ maxL)
print("Total return over "+str(ng+nl)+ " trades: "+ str(totalR)+"%" )
#print("Example return Simulating "+str(n)+ " trades: "+ str(nReturn)+"%" )
print()
```

```
Results for AAPL going back to 2019-01-02 00:00:00, Sample size: 18 trades
EMAs used: [3, 5, 8, 10, 12, 15, 30, 35, 40, 45, 50, 60]
Batting Avg: 0.5555555555555556
Gain/loss ratio: 3.575342819793644
Average Gain: 17.670283763306998
Average Loss: -4.942262785398258
Max Return: 54.669707446832014
Max Loss: 54.669707446832014
Total return over 18 trades: 208.0%
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js