Simple Moving Average Crossover Trading Strategy

Introduction:

The program aims to produce buy and sell signal by using Simple Moving Average Crossover Strategy on AAPL and JPM stock separately.

The Simple Moving Average Crossover involves two terms:

Golden Cross: The golden cross when a short-term moving average cross over a major long-term moving average to the upside and is interpreted by analysts and traders as signaling a definitive upward turn in a market.

Death Cross: The death cross occurs when the short-term average trends down and crosses the long-term average, basically going in the opposite direction of the golden cross

Data and Methods:

Importing packages and dataset for AAPL and JPM stocks

```
# Importing necessary packages
from datetime import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

```
#Importing the dataset
from pandas_datareader import data

#Specifying start date and end date
#I started form January 1st of 2020, to include the market crash due to covid
start_date='2018-1-1'
end_date='2020-12-1'

#Importing AAPL stock
AAPL_data=data.DataReader("AAPL","yahoo",start_date,end_date)

#Importing JPM stock
JPM_data=data.DataReader("JPM","yahoo",start_date,end_date)
```

Showing AAPL stock dataset

```
#Showing the AAPL dataset
AAPL_data
```

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-02	43.075001	42.314999	42.540001	43.064999	102223600.0	41.442081
2018-01-03	43.637501	42.990002	43.132500	43.057499	118071600.0	41.434864
2018-01-04	43.367500	43.020000	43.134998	43.257500	89738400.0	41.627323
2018-01-05	43.842499	43.262501	43.360001	43.750000	94640000.0	42.101261
2018-01-08	43.902500	43.482498	43.587502	43.587502	82271200.0	41.944889
2018-01-09	43.764999	43.352501	43.637501	43.582500	86336000.0	41.940075
2018-01-10	43.575001	43.250000	43.290001	43.572498	95839600.0	41.930458
2018-01-11	43.872501	43.622501	43.647499	43.820000	74670800.0	42.168621
2018-01-12	44.340000	43.912498	44.044998	44.272499	101672400.0	42.604076
2018-01-16	44.847500	44.035000	44.474998	44.047501	118263600.0	42.387554
2018-01-17	44.812500	43.767502	44.037498	44.775002	137547200.0	43.087643
2018-01-18	45.025002	44.562500	44.842499	44.814999	124773600.0	43.126129
2018-01-19	44.895000	44.352501	44.652500	44.615002	129700400.0	42.933674
2018-01-22	44.445000	44.150002	44.325001	44.250000	108434400.0	42.582420
2018-01-23	44.860001	44.205002	44.325001	44.259998	130756400.0	42.592049
2018-01-24	44.325001	43.299999	44.312500	43.555000	204120400.0	41.913613

Showing JMP stock dataset

```
#Showing the JPM dataset
JPM_data
```

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-02	108.019997	106.809998	107.629997	107.949997	13578800.0	98.750923
2018-01-03	108.489998	107.480003	107.860001	108.059998	11901000.0	98.851555
2018-01-04	110.029999	108.199997	108.360001	109.040001	12953700.0	100.267670
2018-01-05	109.550003	107.779999	109.260002	108.339996	14155000.0	99.623993
2018-01-08	108.680000	107.699997	108.150002	108.500000	12466500.0	99.771095
2018-01-09	109.629997	108.489998	108.720001	109.050003	13292300.0	100.276863
2018-01-10	110.699997	109.389999	109.470001	110.250000	15834500.0	101.380318
2018-01-11	110.930000	110.050003	110.669998	110.839996	13676800.0	101.922836
2018-01-12	112.849998	110.839996	111.650002	112.669998	18884200.0	103.605621
2018-01-16	113.430000	111.070000	111.510002	112.269997	22703300.0	103.237808
2018-01-17	113.300003	111.309998	111.889999	112.989998	14940300.0	103.899887
2018-01-18	113.720001	112.269997	112.760002	113.260002	14572900.0	104.148163
2018-01-19	114.339996	112.800003	113.940002	113.010002	18785500.0	103.918266
2018-01-22	114.389999	112.500000	112.660004	114.330002	12475700.0	105.132080
2018-01-23	114.639999	113.349998	113.669998	114.209999	12320800.0	105.021736
2018-01-24	116.000000	114.660004	114.860001	115.669998	15904500.0	106.364281
2018-01-25	116.169998	115.080002	116.040001	115.699997	13510000.0	106.391853

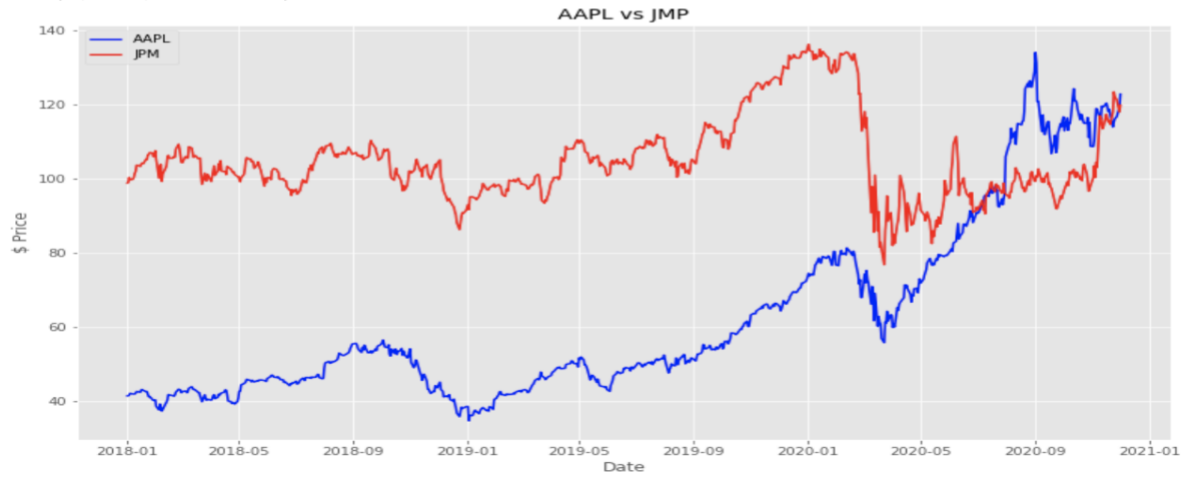
Plot of AAPL and JPM stock from Jan.2018- Dec.2020

```
#Plot the data
plt.figure(figsize=(13,7.5))
plt.plot(AAPL_data["Adj Close"],label="AAPL",color="blue")
plt.plot(JPM_data["Adj Close"],label="JPM",color="red")
plt.title("AAPL vs JMP")
plt.legend(loc="upper left")
plt.xlabel("Date")
plt.ylabel("$ Price")

##Analyzing the graph
#From the graph; we can see that APPL stock plummeted after Feb 20th till about
# about the mid  of March,  then it got into upward trend untill the beginning of the september

#From the graph; we can see that JPM stock severly dropped on Feb 2nd  till about
# about the end of March.
```

```
Text(0, 0.5, '$ Price')
```



Constructing simple moving average with 20-day period for AAPL and JPM stocks

```
#Constructing the simple moving average with 20 day period for AAPL
AAPL_SMA20=pd.DataFrame()
AAPL_SMA20["Adj Close"]=AAPL_data["Adj Close"].rolling(window=20).mean()

AAPL_SMA20

#Constructing the simple moving average with 20 day period for JPM
JPM_SMA20=pd.DataFrame()
JPM_SMA20["Adj Close"]=JPM_data["Adj Close"].rolling(window=20).mean()
#JPM_SMA20
```

Constructing simple moving average with 100-day period for AAPL and JPM stocks

```
#Constructing the simple moving average with 100 day period for AAPL
AAPL_SMA100=pd.DataFrame()
AAPL_SMA100["Adj Close"]=AAPL_data["Adj Close"].rolling(window=100).mean()
AAPL_SMA100

#Constructing the simple moving average with 100 day period for JPM
JPM_SMA100=pd.DataFrame()
JPM_SMA100["Adj Close"]=JPM_data["Adj Close"].rolling(window=100).mean()
JPM_SMA100
```

Plot of AAPL stock with moving Averages

```
#Plot the AAPL stock and moving averages
plt.figure(figsize=(16,10))
plt.plot(AAPL_SMA20,label="SMA20",color="yellow")
plt.plot(AAPL_SMA100,label="SMA100",color="green")
plt.plot(AAPL_data["Adj Close"],label="AAPL",color="blue")
plt.ylabel("$ Price")
plt.xlabel("Date")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7faeeed54550>
```

Plot of JPM stock with Simple Moving Averages

```
#Plot the JPM stock and moving averages
plt.figure(figsize=(16,10))
plt.plot(JPM_SMA20,label="SMA20",color="yellow")
plt.plot(JPM_SMA100,label="SMA100",color="green")
plt.plot(JPM_data["Adj Close"],label="JPM",color="purple")
plt.legend()
plt.ylabel("$ Price")
plt.xlabel("Date")
```

```
Text(0.5, 0, 'Date')
```

Constructing a new data frame (AAPL_df) including SMA 20, SMA100 and Adjusted Closing price

```
#Constructing a new AAPL data frame which includes Adj Close, SMA20, SMA100
AAPL_df=pd.DataFrame()
AAPL_df["SMA20"]=AAPL_SMA20["Adj Close"]
AAPL_df["SMA100"]=AAPL_SMA100["Adj Close"]
AAPL_df["AAPL"]=AAPL_data["Adj Close"]
AAPL_df
```

Date	SMA20	SMA100	AAPL
2018-01-02	NaN	NaN	41.442081
2018-01-03	NaN	NaN	41.434864
2018-01-04	NaN	NaN	41.627323
2018-01-05	NaN	NaN	42.101261
2018-01-08	NaN	NaN	41.944889
2018-01-09	NaN	NaN	41.940075
2018-01-10	NaN	NaN	41.930458
2018-01-11	NaN	NaN	42.168621
2018-01-12	NaN	NaN	42.604076
2018-01-16	NaN	NaN	42.387554
2018-01-17	NaN	NaN	43.087643
2018-01-18	NaN	NaN	43.126129
2018-01-19	NaN	NaN	42.933674
2018-01-22	NaN	NaN	42.582420
2018-01-23	NaN	NaN	42.592049
2018-01-24	NaN	NaN	41.913612

Constructing a new data frame (JPM_df) including SMA 20, SMA100 and Adjusted Closing price

```
#Constructing a new JPM data frame which includes Adj Close, SMA20, SMA100
JPM_df=pd.DataFrame()
JPM_df["SMA20"]=JPM_SMA20["Adj Close"]
JPM_df["SMA100"]=JPM_SMA100["Adj Close"]
JPM_df["JPM"]=JPM_data["Adj Close"]

JPM_df
```

Date	SMA20	SMA100	JPM
2018-01-02	NaN	NaN	98.750923
2018-01-03	NaN	NaN	98.851555
2018-01-04	NaN	NaN	100.267670
2018-01-05	NaN	NaN	99.623993
2018-01-08	NaN	NaN	99.771095
2018-01-09	NaN	NaN	100.276863
2018-01-10	NaN	NaN	101.380318
2018-01-11	NaN	NaN	101.922836
2018-01-12	NaN	NaN	103.605621
2018-01-16	NaN	NaN	103.237808
2018-01-17	NaN	NaN	103.899887
2018-01-18	NaN	NaN	104.148163
2018-01-19	NaN	NaN	103.918266
2018-01-22	NaN	NaN	105.132080
2018-01-23	NaN	NaN	105.021736

Creating a function to give a signal of where to buy and where to sell

```
##Creating a function to give a singal of where to buy and where to sell
def signal(array):

  sig=-1
  count=0
  lis=list()

  for i in range(len(array)):

    if array["SMA20"][count]>array["SMA100"][count] and sig is not 1 :

      lis.append("buy")

      sig=1


    elif array["SMA20"][count]<array["SMA100"][count] and sig is not 0:


      lis.append("sell")

      sig=0



    else:
      lis.append(np.nan)


    count=count+1

  return lis
```

Calculating daily return for AAPL stock and adding it into AAPL data frame with Signal

```
#Adding the Signal and Return to the AAPL_data frame
AAPL_df["Signal_AAPL"]=signal(AAPL_df)
AAPL_df["Signal_AAPL"]
AAPL_df["AAPL_Return"]=AAPL_df[["AAPL"]].pct_change()
AAPL_df
```

Date	SMA20	SMA100	AAPL	Signal_AAPL	AAPL_Return
2018-01-02	NaN	NaN	41.442081	NaN	NaN
2018-01-03	NaN	NaN	41.434864	NaN	-0.000174
2018-01-04	NaN	NaN	41.627323	NaN	0.004645
2018-01-05	NaN	NaN	42.101261	NaN	0.011385
2018-01-08	NaN	NaN	41.944889	NaN	-0.003714
2018-01-09	NaN	NaN	41.940075	NaN	-0.000115
2018-01-10	NaN	NaN	41.930458	NaN	-0.000229
2018-01-11	NaN	NaN	42.168621	NaN	0.005680
2018-01-12	NaN	NaN	42.604076	NaN	0.010327
2018-01-16	NaN	NaN	42.387554	NaN	-0.005082
2018-01-17	NaN	NaN	43.087643	NaN	0.016516
2018-01-18	NaN	NaN	43.126129	NaN	0.000893
2018-01-19	NaN	NaN	42.933674	NaN	-0.004463
2018-01-22	NaN	NaN	42.582420	NaN	-0.008181

Calculating daily return for JPM stock and adding it into JPM data frame with Signal

```
# Adding signal and Return in dataframe
JPM_df["Signal_JPM"]=signal(JPM_df)
#JPM_df["Signal_JPM"]
JPM_df["JPM_Return"]=JPM_df[["JPM"]].pct_change()
JPM_df
```

Date	SMA20	SMA100	JPM	Signal_JPM	JPM_Return
2018-01-02	NaN	NaN	98.750923	NaN	NaN
2018-01-03	NaN	NaN	98.851555	NaN	0.001019
2018-01-04	NaN	NaN	100.267670	NaN	0.014326
2018-01-05	NaN	NaN	99.623993	NaN	-0.006420
2018-01-08	NaN	NaN	99.771095	NaN	0.001477
2018-01-09	NaN	NaN	100.276863	NaN	0.005069
2018-01-10	NaN	NaN	101.380318	NaN	0.011004
2018-01-11	NaN	NaN	101.922836	NaN	0.005351
2018-01-12	NaN	NaN	103.605621	NaN	0.016510
2018-01-16	NaN	NaN	103.237808	NaN	-0.003550
2018-01-17	NaN	NaN	103.899887	NaN	0.006413
2018-01-18	NaN	NaN	104.148163	NaN	0.002390
2018-01-19	NaN	NaN	103.918266	NaN	-0.002207
2018-01-22	NaN	NaN	105.132080	NaN	0.011680
2018-01-23	NaN	NaN	105.021736	NaN	-0.001050
2018-01-24	NaN	NaN	106.364281	NaN	0.012783

In order to see both AAPL data frame and JPM data frame together, two data frames were joined.

```
joined_df=AAPL_df.join(JPM_df)
#JPM_df[["Signal","JPM"]])

joined_df
```

Date	SMA20	SMA100	AAPL	Signal_AAPL	Signal_JPM	JPM	SMA20_JPM	SMA100_JPM
2018-01-02	NaN	NaN	41.105534	NaN	NaN	98.750923	NaN	NaN
2018-01-03	NaN	NaN	41.098373	NaN	NaN	98.851555	NaN	NaN
2018-01-04	NaN	NaN	41.289268	NaN	NaN	100.267670	NaN	NaN
2018-01-05	NaN	NaN	41.759361	NaN	NaN	99.623993	NaN	NaN
2018-01-08	NaN	NaN	41.604263	NaN	NaN	99.771095	NaN	NaN
...	...	...	...	...	...	...	...	...
2020-11-24	115.943592	112.329833	115.169998	NaN	NaN	123.320000	109.9300	100.536680
2020-11-25	116.194669	112.561686	116.029999	NaN	NaN	122.029999	111.2045	100.842268
2020-11-27	116.268099	112.777516	116.589996	NaN	NaN	121.220001	112.4070	101.130046
2020-11-30	116.786974	113.013861	119.050003	NaN	NaN	117.879997	113.3990	101.404438
2020-12-01	117.493840	113.285236	122.720001	NaN	NaN	119.739998	114.3735	101.647989

Summary:

To sum up,

According to plots and data frames above, it could be said that simple moving average crossover strategy is useful but not the best strategy, because it's not able to capture all the buy and sell point during the life of the stock. However, for middle and long-term strategies it could be quite useful.

Muhammet Furkan Isik