In [1]:
```
import pandas as pd
import numpy as np
import util as ut
import matplotlib.pyplot as plt
import datetime as dt
import random

from indicators import *
from marketsimcode import *
import ManualStrategy as ms
import StrategyLearner as sl
from marketsimcode import compute_portvals
```

In [2]:
```
def createBenchmark(symbol="JPM", sd=dt.datetime(2008, 1, 1),  ed=dt.datetime(2009, 12, 31),  sv=100000):

    trades = pd.DataFrame(columns=['Date', 'Symbol', 'Order', 'Shares'])
    syms = [symbol]
    dates = pd.date_range(sd, ed)
    prices_all = ut.get_data(syms, dates)
    prices = prices_all[syms]
    numRows = prices.shape[0]-1

    trades.loc[0] = [prices.index[0].strftime('%Y-%m-%d'),symbol,'BUY', 1000]
    trades.loc[1] = [prices.index[numRows].strftime('%Y-%m-%d'),symbol,'SELL', -1000]

    trades.set_index('Date', inplace=True)
    trades.index = pd.to_datetime(trades.index, format='%Y-%m-%d')

    return trades
```

In [3]:
```
def computePerformance(port_vals):
    # Summary of Performance of Manual Strategy
    dailyRets = (port_vals / port_vals.shift(1)) - 1
    dailyRets = dailyRets[1:]
    dailyRiskFreeReturn = 0
    sharpeRatio = (np.mean(dailyRets - dailyRiskFreeReturn)) / (np.std(dailyRets))
    numberOfSamples = 252                                   # daily(252), weekly(52), monthly(12)
    k_factor = np.sqrt(numberOfSamples)
    sharpeRatioAnnualized = k_factor * sharpeRatio

    cr_in, adr_in, sddr_in, sr_in = [
        (port_vals[-1]/port_vals[0])-1,
        dailyRets.mean(),
        dailyRets.std(),
        sharpeRatioAnnualized,
    ]
```

In [4]:
```
def toOrders(dailyTrades, symbol="JPM"):

    trades = pd.DataFrame(columns=['Date', 'Symbol', 'Order', 'Shares'])
    index = 0

    for i in range(0, dailyTrades.shape[0]):
        if dailyTrades[symbol].iloc[i] == 2000:
            trades.loc[index] = [dailyTrades.index[i].strftime('%Y-%m-%d'),symbol,'BUY',2000]
            index += 1
        elif dailyTrades[symbol].iloc[i] == 1000:
            trades.loc[index] = [dailyTrades.index[i].strftime('%Y-%m-%d'),symbol,'BUY',1000]
            index += 1
        elif dailyTrades[symbol].iloc[i] == -2000:
            trades.loc[index] = [dailyTrades.index[i].strftime('%Y-%m-%d'),symbol,'SELL',2000]
            index += 1
        elif dailyTrades[symbol].iloc[i] == -1000:
            trades.loc[index] = [dailyTrades.index[i].strftime('%Y-%m-%d'),symbol,'SELL',1000]
            index += 1

    trades.set_index('Date', inplace=True)
    trades.index = pd.to_datetime(trades.index, format='%Y-%m-%d')
    return trades
```

In [5]:
```
def computePerformance(port_vals):
    dailyRets = (port_vals / port_vals.shift(1)) - 1
    dailyRets = dailyRets[1:]
    dailyRiskFreeReturn = 0
    sharpeRatio = (np.mean(dailyRets - dailyRiskFreeReturn)) / (np.std(dailyRets))
    numberOfSamples = 252                                   # daily(252), weekly(52), monthly(12)
    k_factor = np.sqrt(numberOfSamples)
    sharpeRatioAnnualized = k_factor * sharpeRatio

    cr, adr, sddr, sr = [
        (port_vals[-1]/port_vals[0])-1,
        dailyRets.mean(),
        dailyRets.std(),
        sharpeRatioAnnualized,
    ]

    return (cr, adr, sddr, sr)
```

In [58]:
```
symbol="SPY"
in_sd=dt.datetime(2018, 1, 1)
in_ed=dt.datetime(2019, 12, 31)
out_sd=dt.datetime(2020, 1, 1)
out_ed=dt.datetime(2020, 12, 31)
sv=100000
commission=0.0
impact=0.0
```

In [59]:
```
in_benchmark = createBenchmark(symbol=symbol, sd=in_sd, ed=in_ed, sv=sv)
in_ManualTrades, in_buys, in_sells, in_strongbuys, in_strongsells = ms.testPolicy(symbol=symbol, sd=in_sd,  ed=in_ed,  sv=sv)
out_benchmark = createBenchmark(symbol=symbol, sd=out_sd, ed=out_ed, sv=sv)
out_ManualTrades, out_buys, out_sells, out_strongbuys, out_strongsells = ms.testPolicy(symbol=symbol, sd=out_sd,  ed=out_ed,  sv=sv)

learner = sl.StrategyLearner(verbose = False, impact=impact)
learner.add_evidence(symbol=symbol, sd=in_sd, ed=in_ed, sv=sv)

in_MLTrades = learner.testPolicy(symbol=symbol, sd=in_sd, ed=in_ed, sv=sv)
out_MLTrades = learner.testPolicy(symbol=symbol, sd=out_sd, ed=out_ed, sv=sv)

in_MLTrades = toOrders(symbol=symbol, dailyTrades = in_MLTrades)
out_MLTrades = toOrders(symbol=symbol, dailyTrades = out_MLTrades)

in_df1 = compute_portvals(in_benchmark, start_date = in_sd, end_date = in_ed , start_val = sv, commission=commission, impact=impact)
in_df2 = compute_portvals(in_ManualTrades, start_date = in_sd, end_date = in_ed , start_val = sv, commission=commission, impact=impact)
in_ml = compute_portvals(in_MLTrades, start_date = in_sd, end_date = in_ed , start_val = sv, commission=commission, impact=impact)

out_df1 = compute_portvals(out_benchmark, start_date = out_sd, end_date = out_ed , start_val = sv, commission=commission, impact=impact)
out_df2 = compute_portvals(out_ManualTrades, start_date = out_sd, end_date = out_ed , start_val = sv, commission=commission, impact=impact)
out_ml = compute_portvals(out_MLTrades, start_date = out_sd, end_date = out_ed , start_val = sv, commission=commission, impact=impact)
```

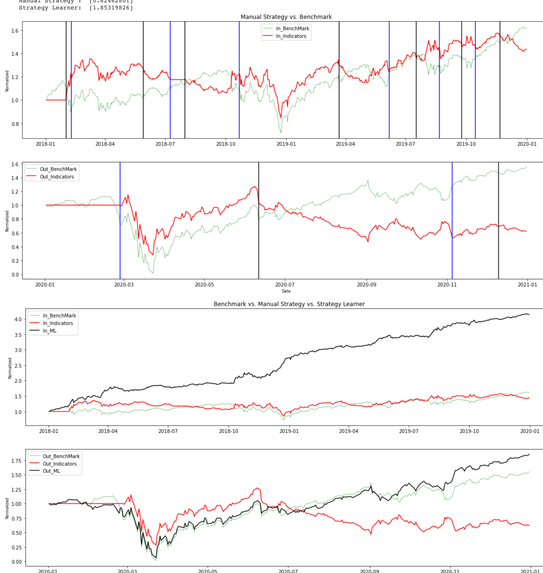
```
<Figure size 1440x1008 with 0 Axes>
<Figure size 1440x1008 with 0 Axes>
```

In [60]:
```
# Summary of Performance of Manual Strategy
cr_in, adr_in, sddr_in, sr_in = computePerformance(in_df2)
print('Performance stats of Manual Strategy')
print('   In-Sample Period   ')
print('    Cumulative Return   : ', cr_in)
print('    Mean of Daily Return : ', adr_in)
print('    Standard Deviation  : ', sddr_in)
print('    Sharpe Ratio        : ', sr_in)

cr_out, adr_out, sddr_out, sr_out = computePerformance(out_df2)
print('   Out-Sample Period   ')
print('    Cumulative Return   : ', cr_out)
print('    Mean of Daily Return : ', adr_out)
print('    Standard Deviation  : ', sddr_out)
print('    Sharpe Ratio        : ', sr_out)
```

```
Performance stats of Manual Strategy
   In-Sample Period
    Cumulative Return   :  0.4358141800000305
    Mean of Daily Return:  0.0009304201710181016
    Standard Deviation  :  0.020609034820742913
    Sharpe Ratio        :  0.7173890279497704
   Out-Sample Period
    Cumulative Return   :  -0.3757179900000254
    Mean of Daily Return:  0.002141080429169356
    Standard Deviation  :  0.09225675753041711
    Sharpe Ratio        :  0.36916426196693073
```

In [61]:
```
in_df1 = pd.DataFrame(in_df1)
in_df2 = pd.DataFrame(in_df2)
in_ml1 = pd.DataFrame(in_ml1)
out_df1 = pd.DataFrame(out_df1)
out_df2 = pd.DataFrame(out_df2)
out_ml1 = pd.DataFrame(out_ml1)

in_df1_norm = in_df1 / in_df1.iloc[0]
in_df2_norm = in_df2 / in_df2.iloc[0]
in_ml_norm = in_ml1 / in_ml1.iloc[0]
out_df1_norm = out_df1 / out_df1.iloc[0]
out_df2_norm = out_df2 / out_df2.iloc[0]
out_ml1_norm = out_ml1 / out_ml1.iloc[0]

perf_in_df1 = in_df1_norm.iloc[-1].values
perf_in_df2 = in_df2_norm.iloc[-1].values
perf_in_ml = in_ml_norm.iloc[-1].values
perf_out_df1 = out_df1_norm.iloc[-1].values
perf_out_df2 = out_df2_norm.iloc[-1].values
perf_out_ml1 = out_ml1_norm.iloc[-1].values

print('Performance of Manual Strategy')
print('   In-Sample Period   ')
print('    BenchMark       : ', perf_in_df1)
print('    Manual Strategy : ', perf_in_df2)
print('    Strategy Lerarer: ', perf_in_ml1)
print('   Out-Sample Period   ')
print('    BenchMark       : ', perf_out_df1)
print('    Manual Strategy : ', perf_out_df2)
print('    Strategy Learner: ', perf_out_ml1)

plt.figure(figsize=(20,10))
plt.subplot(211)
plt.plot(in_df1_norm.index, in_df1_norm['Total'], label='In_BenchMark', linewidth=0.5, color='green')
plt.plot(in_df2_norm.index, in_df2_norm['Total'], label='In_Indicators', color='red')
for i in range(len(in_buys)):
    plt.axvline(x=in_buys[i], color='blue')
for i in range(len(in_strongbuys)):
    plt.axvline(x=in_strongbuys[i], color='blue')
for i in range(len(in_sells)):
    plt.axvline(x=in_sells[i], color='black')
for i in range(len(in_strongsells)):
    plt.axvline(x=in_strongsells[i], color='black')
plt.title('Manual Strategy vs. Benchmark')
plt.ylabel('Normalized', fontsize=8)
plt.legend(loc='best')
plt.subplot(212)
plt.plot(out_df1_norm.index, out_df1_norm['Total'], label='Out_BenchMark', linewidth=0.5, color='green')
plt.plot(out_df2_norm.index, out_df2_norm['Total'], label='Out_Indicators', color='red')

for i in range(len(out_buys)):
    plt.axvline(x=out_buys[i], color='blue')
for i in range(len(out_strongbuys)):
    plt.axvline(x=out_strongbuys[i], color='blue')
for i in range(len(out_sells)):
    plt.axvline(x=out_sells[i], color='black')
for i in range(len(out_strongsells)):
    plt.axvline(x=out_strongsells[i], color='black')
plt.xlabel('Date', fontsize=8)
plt.ylabel('Normalized', fontsize=8)
plt.legend(loc='best')
plt.show()

plt.clf
plt.figure(figsize=(20,10))
plt.subplot(211)
plt.plot(in_df1_norm.index, in_df1_norm['Total'], label='In_BenchMark', linewidth=0.5, color='green')
plt.plot(in_df2_norm.index, in_df2_norm['Total'], label='In_Indicators', color='red')
plt.plot(in_ml_norm.index, in_ml_norm['Total'], label='In_ML', color = 'black')
plt.title('Benchmark vs. Manual Strategy vs. Strategy Learner')
plt.ylabel('Normalized', fontsize=8)
plt.legend(loc='best')
plt.subplot(212)
plt.plot(out_df1_norm.index, out_df1_norm['Total'], label='Out_BenchMark', linewidth=0.5, color='green')
plt.plot(out_df2_norm.index, out_df2_norm['Total'], label='Out_Indicators', color='red')
plt.plot(out_ml_norm.index, out_ml_norm['Total'], label='Out_ML', color = 'black')
plt.xlabel('Date', fontsize=8)
plt.ylabel('Normalized', fontsize=8)
plt.legend(loc='best')

plt.show()
```

```
Performance of Manual Strategy
   In-Sample Period
    BenchMark       : [1.61846786]
    Manual Strategy : [1.43581418]
    Strategy Lerarer: [4.13950638]
   Out-Sample Period
    BenchMark       : [1.54965698]
    Manual Strategy : [0.62462851]
    Strategy Learner: [1.85319826]
```



In [ ]:

In [ ]:

In [ ]: