Mark Trinquero

mtrinquero3

CS 7646 – MC3P2

Fall 2015

**Machine Learning for Trading**

**MC3-P2: Build a Trading Learner**

**Overview of Approach and Indicators:** For this project I choose to use a modified version of my KNN Learner from MC3-Project 1 to server as the ‘learner’ behind my trading strategy. The indicators I selected to utilize for this project were:

X1: Normalized Bollinger Band Feature

X2: Normalized Momentum

X3: Normalized Rolling Standard Deviation of SMA

Y: 5-Day future return (proxy for predicting future price movements)

Datasets from these indicators were inputted into my modified KNN Learner for training and testing purposes (in and out of sample). A brief outline of my approach to each indicator, and overall trading policy has been provided below.

**X1: Normalized Bollinger Band Feature** – for this I used my existing Bollinger Band code and normalized it per the code snippet provide below.



**X2: Normalized Momentum** – for this I adapted the pseudo code provided in the course wiki per the code snippet provided below.



**X3: Normalized Rolling Standard Deviation of SMA** – this was a basic indicator I added to track the movements of the rolling standard deviation of the assets simple moving average.

**Trading Strategy Overview:** I followed a similar trading strategy to the one suggested on the course wiki page, with a few minor modifications I found to improve results. My strategy was as follows:

* Price Forecast (Prediction) Threshold: 0.05% or -0.05%
* Share Amount (Buy/Sell): 100
* Holding Period = 5 Days
* Starting Portfolio Value = $10,000

Based on predicted data from my KNN leaner, my trading agent executed trades when forecasted data indicated movements above of below a particular threshold. I originally used a threshold of 1%, but after further testing decided I wanted to make my agent even more sensitive to predicated data.

* When my forecaster predicted a future price movement (5 day window) of negative .05%, a short sell of 100 shares was triggered. Once the specified holding period (5 trading days) was reached, an exit was triggered.
* When my forecaster predicted a future price movement (5 day window) or positive .05% growth, a buy order of 100 shares was triggered. Once the specified holding period (5 trading days) was reached, an exit was triggered.

**Discussion of Results:** A snapshot of my results using SINE &IBM data is provided below.

SINE DATA:

Data Range: 2007-12-31 to 2009-12-31

RMSE: 0.000489858626518

corr: 0.999994352754

Average Daily Return of Fund: 0.00143110790718

Average Daily Return of SPY: -0.000206479400499

Final Portfolio Value: 19114.8494

Data Range: 2009-12-31 to 2010-12-31

RMSE: 0.00184003058165

corr: 0.999921037189

Average Daily Return of Fund: 0.001943510885

Average Daily Return of SPY: 0.000604173932453

Final Portfolio Value: 15744.054

IBM DATA:

Data Range: 2007-12-31 to 2009-12-31

RMSE: 0.033015636321

corr: 0.647893616273

Average Daily Return of Fund: 0.00154943169043

Average Daily Return of SPY: -0.000206479400499

Final Portfolio Value: 21234.0

Data Range: 2009-12-31 to 2010-12-31

RMSE: 0.0317172844637

corr: 0.0160994792001

Average Daily Return of Fund: 0.00102531028247

Average Daily Return of SPY: 0.000604173932453

Final Portfolio Value: 12752.0

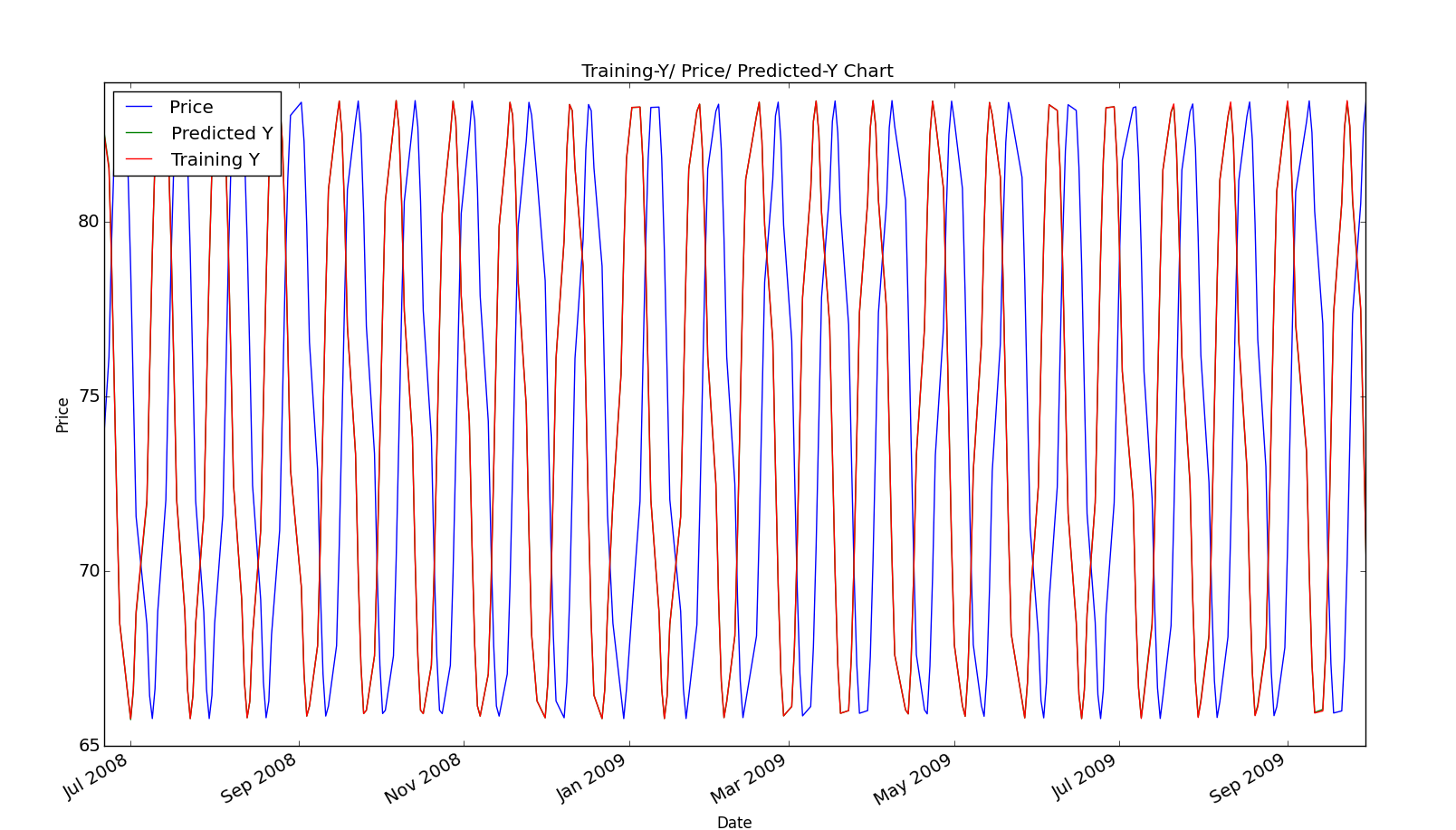
In order to improve my strategy, I would explore adding additional indicators as well as adding functionally to dynamically weight buy/sell amounts based on the strength of the associated indicator. Although this may add additional risk into the portfolio, if leveraged properly I think it could significantly improve expected returns.

**Resources Consulted**

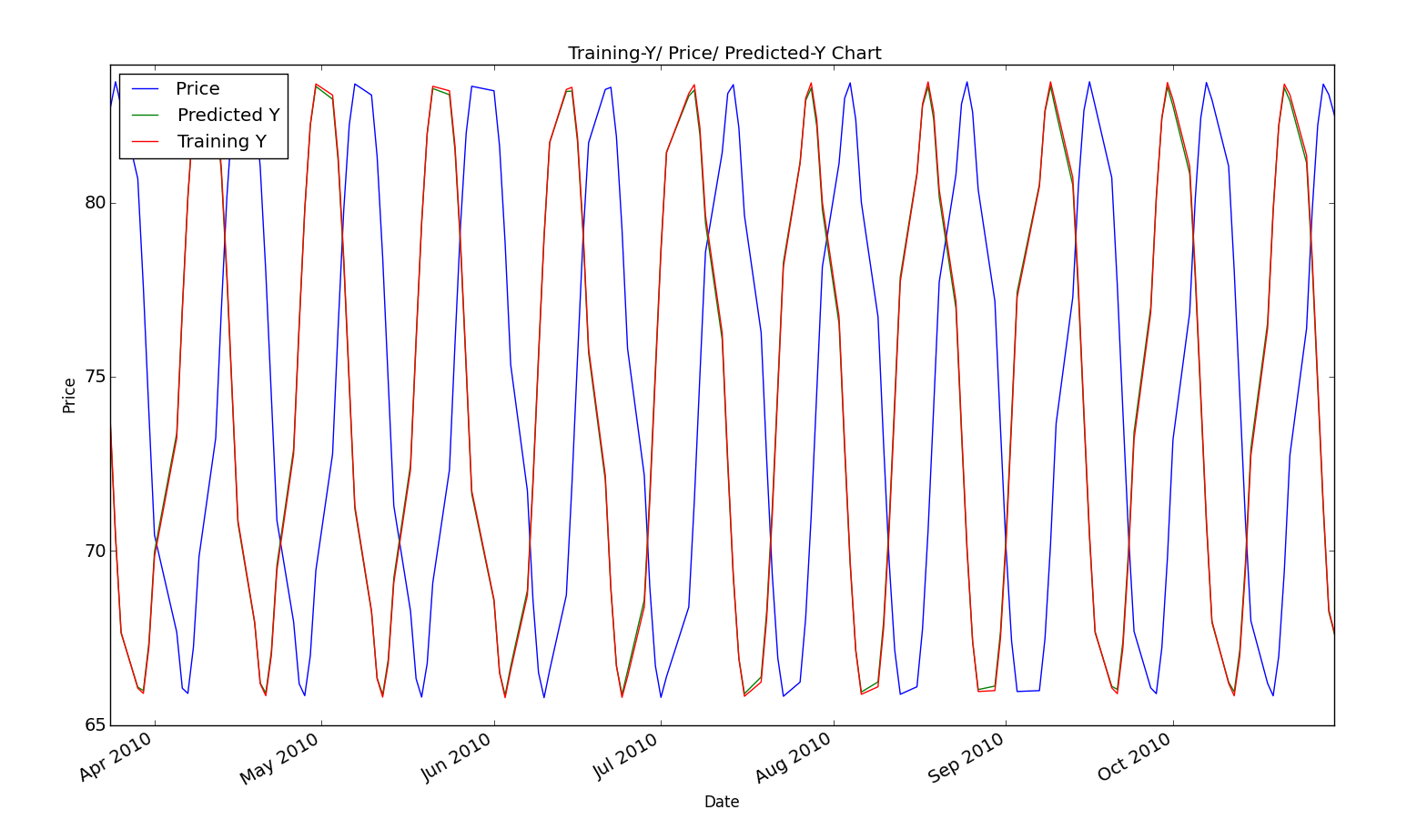
1. Course Piazza Page (all- @1383, @1370, @1332, @1326, @1366, @1219…)
2. <http://quantsoftware.gatech.edu/MC3-Project-1#Hints_.26_resources>
3. <http://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html>
4. <https://www.youtube.com/watch?v=0fEcg_ZsYNY>
5. <http://pandas.pydata.org/pandas-docs/stable/timeseries.html>
6. <http://stackoverflow.com/questions/28356492/how-to-create-all-zero-dataframe-in-python>
7. <http://pandas.pydata.org/pandas-docs/stable/indexing.html>
8. <http://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe>
9. <http://docs.scipy.org/doc/numpy/reference/generated/numpy.sum.html>

**Plot 1A – SINE Data:** Training Y, Predicted Y, and Price (In & Out of Sample)

In Sample Plot:



Out of Sample Plot:



**Price:** Blue Line

**Training Y:** Red Line

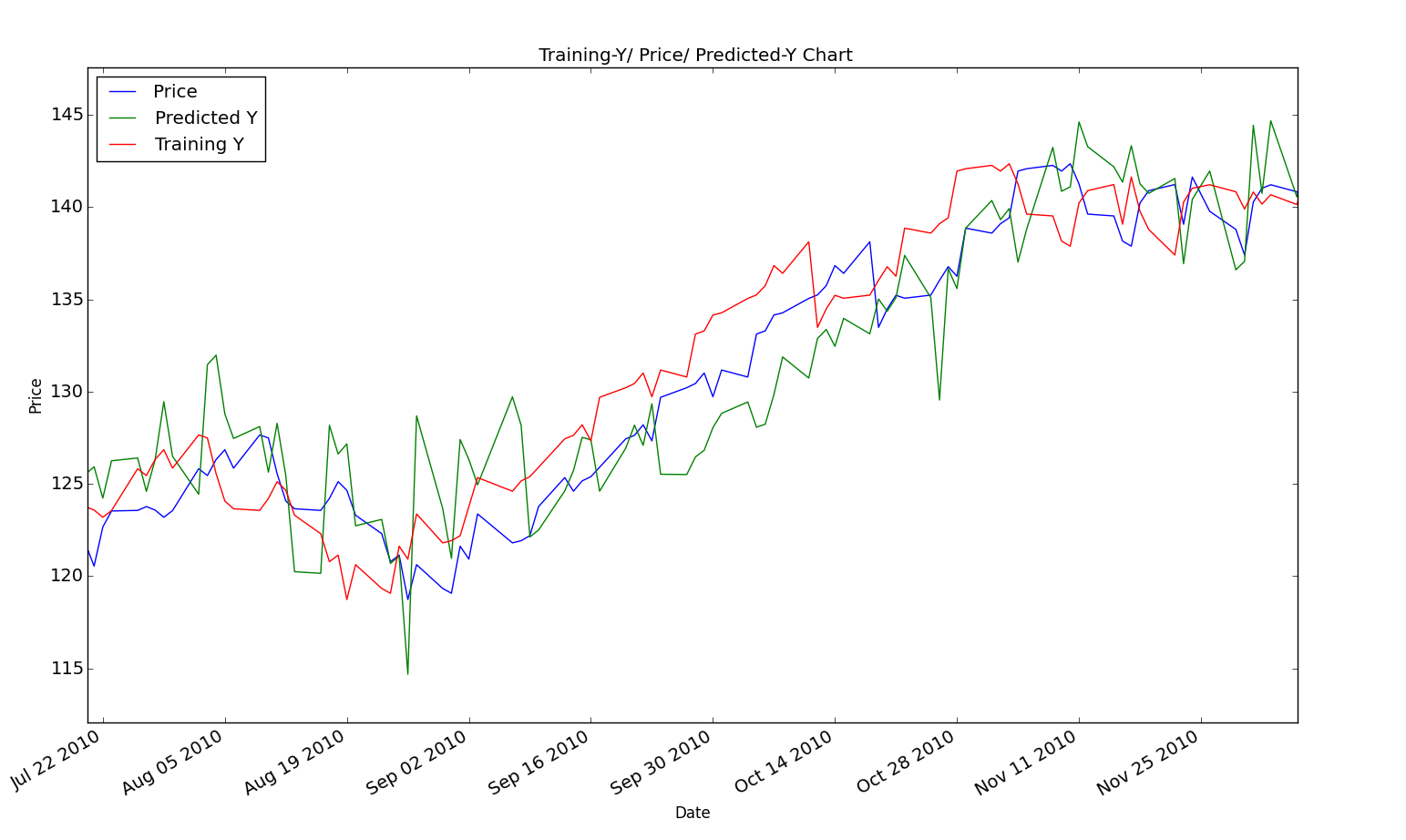
**Predicted Y:** Green Line *(Note: overlays very closely with Training Y)*

**Plot 1B – IBM Data:** Training Y, Predicted Y, and Price (In & Out of Sample)

In Sample Plot:



Out of Sample Plot:



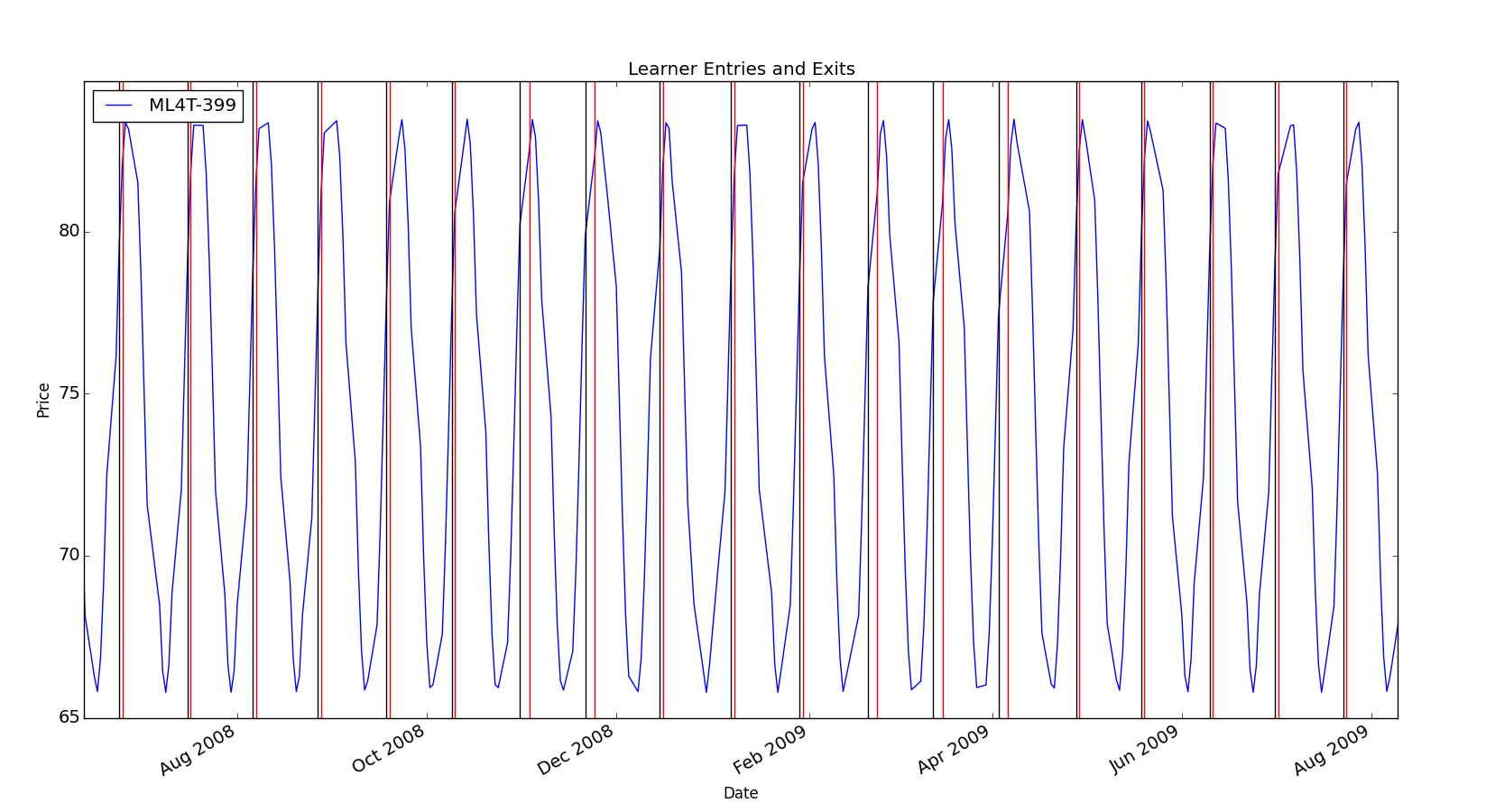
**Price:** Blue Line

**Training Y:** Red Line

**Predicted Y:** Green Line

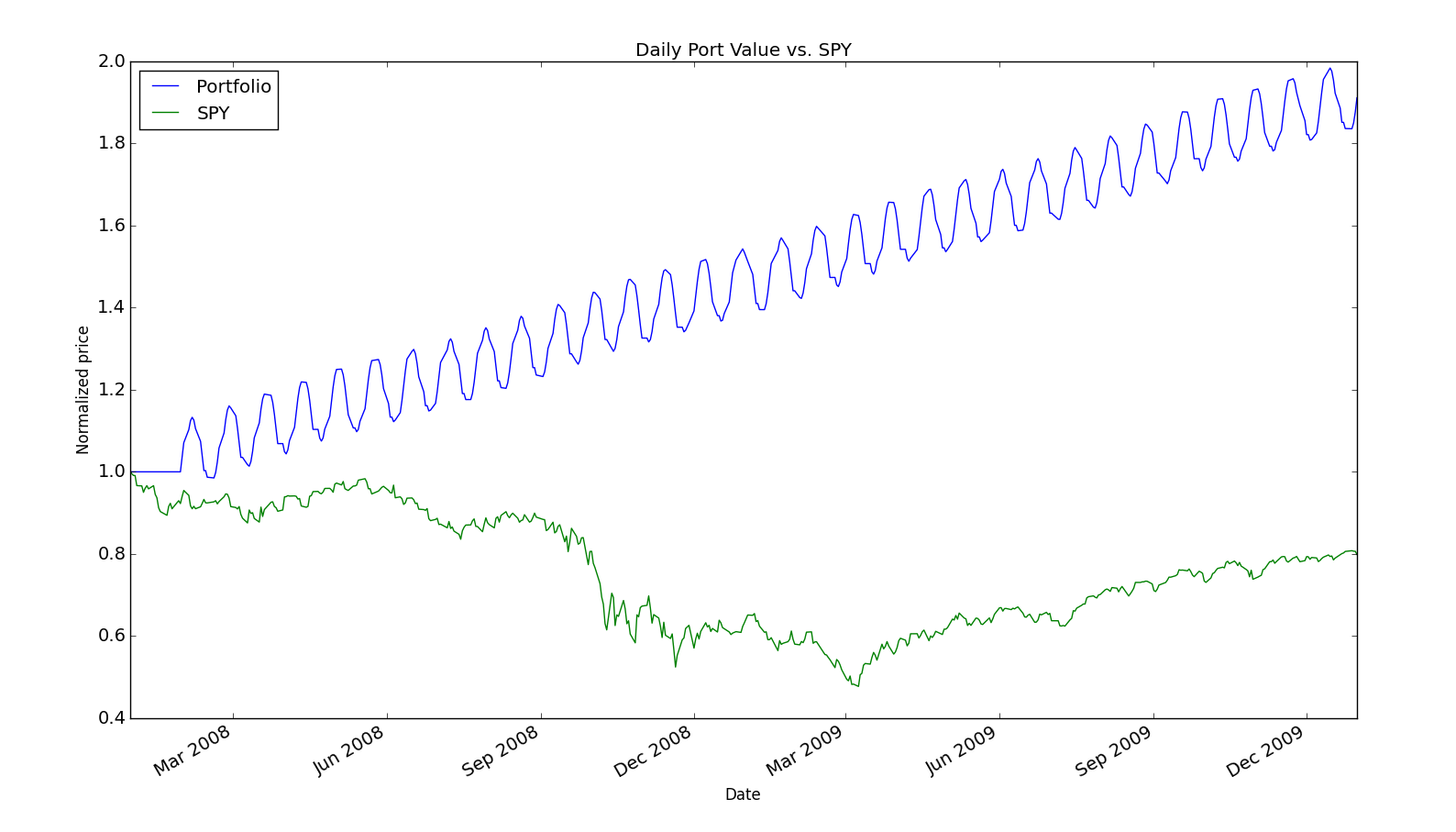
**Plot 2:** Sine Data Entries/Exits (In Sample)

Green Bar = long entry Red Bar = short entry Black Bar = short/long exit



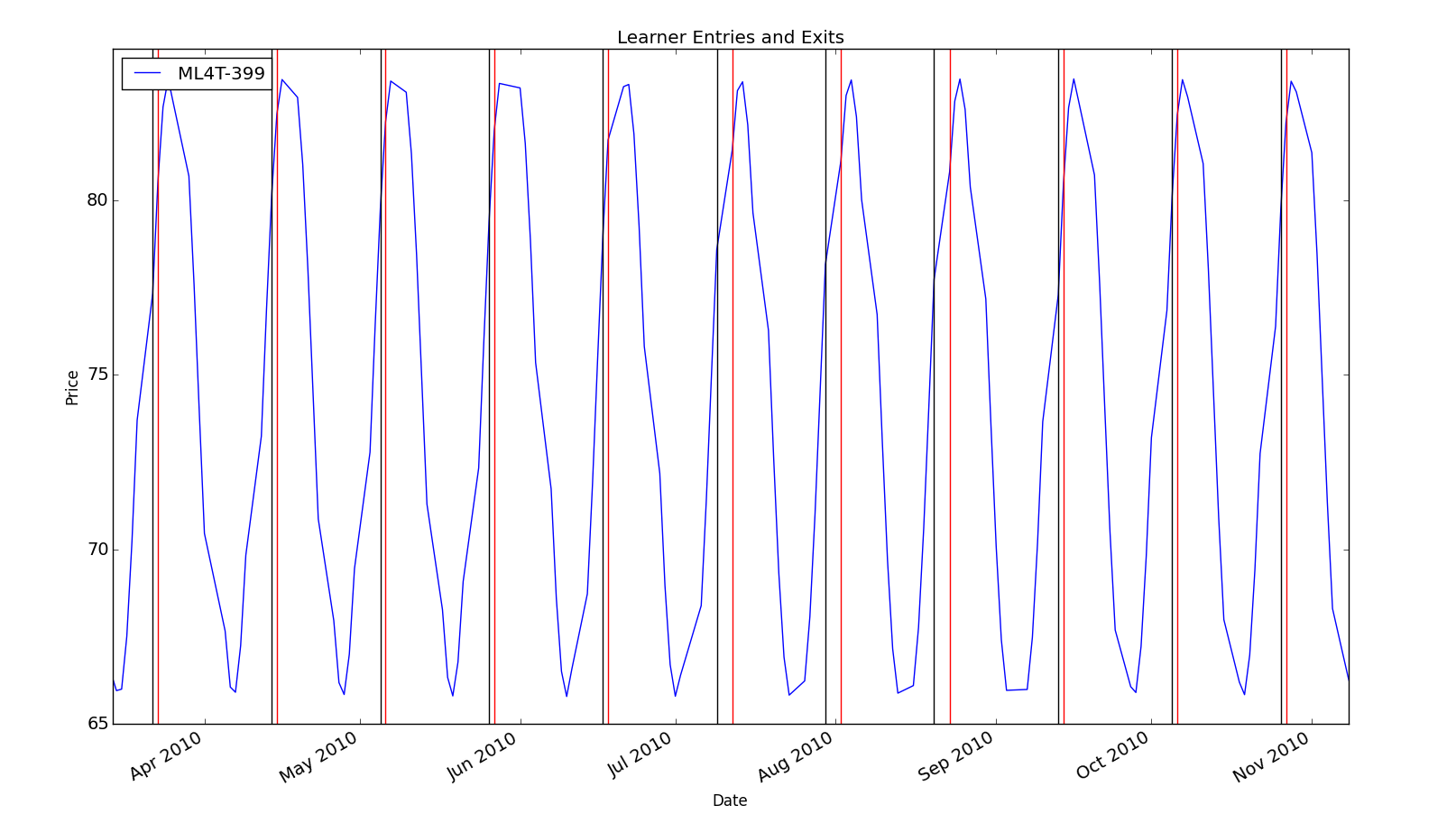
**Plot 3:** Sine Data Backtest (In Sample)

Green Line = SPY Blue Line = Portfolio



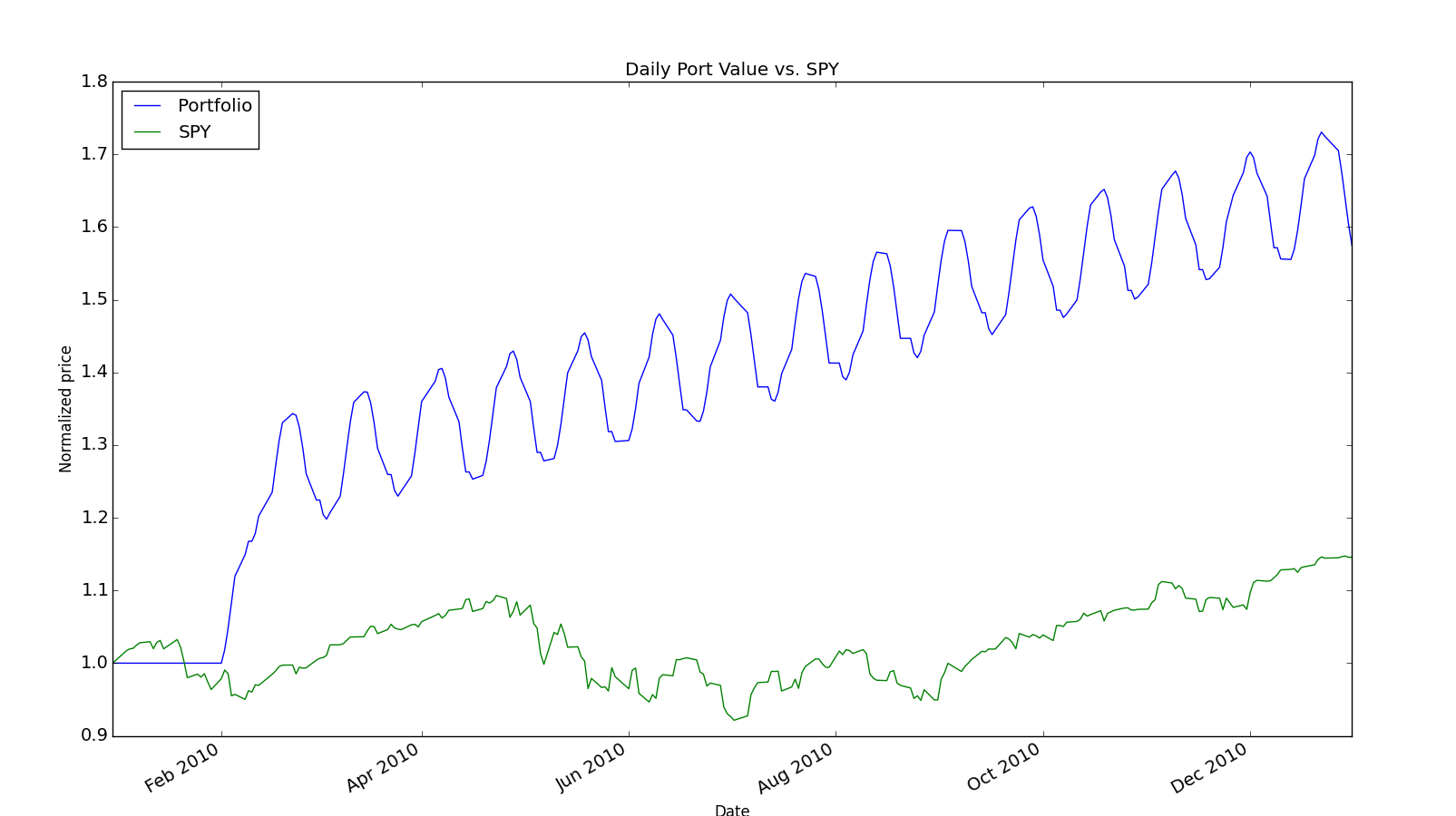
**Plot 4:** Sine Data Entries/Exits (Out of Sample)

Green Bar = long entry Red Bar = short entry Black Bar = short/long exit



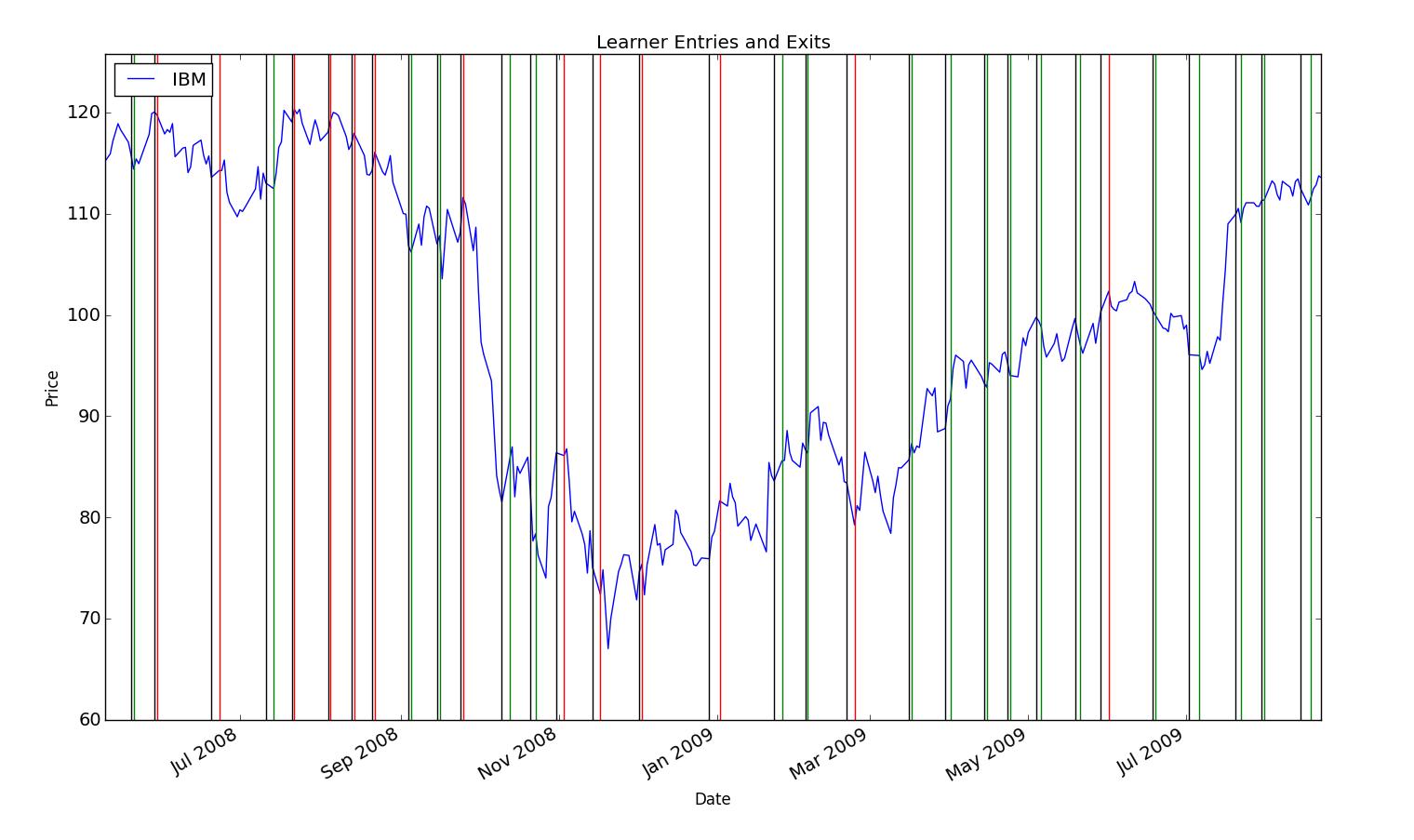
**Plot 5:** Sine Data Backtest (Out of Sample)

Green Line = SPY Blue Line = Portfolio



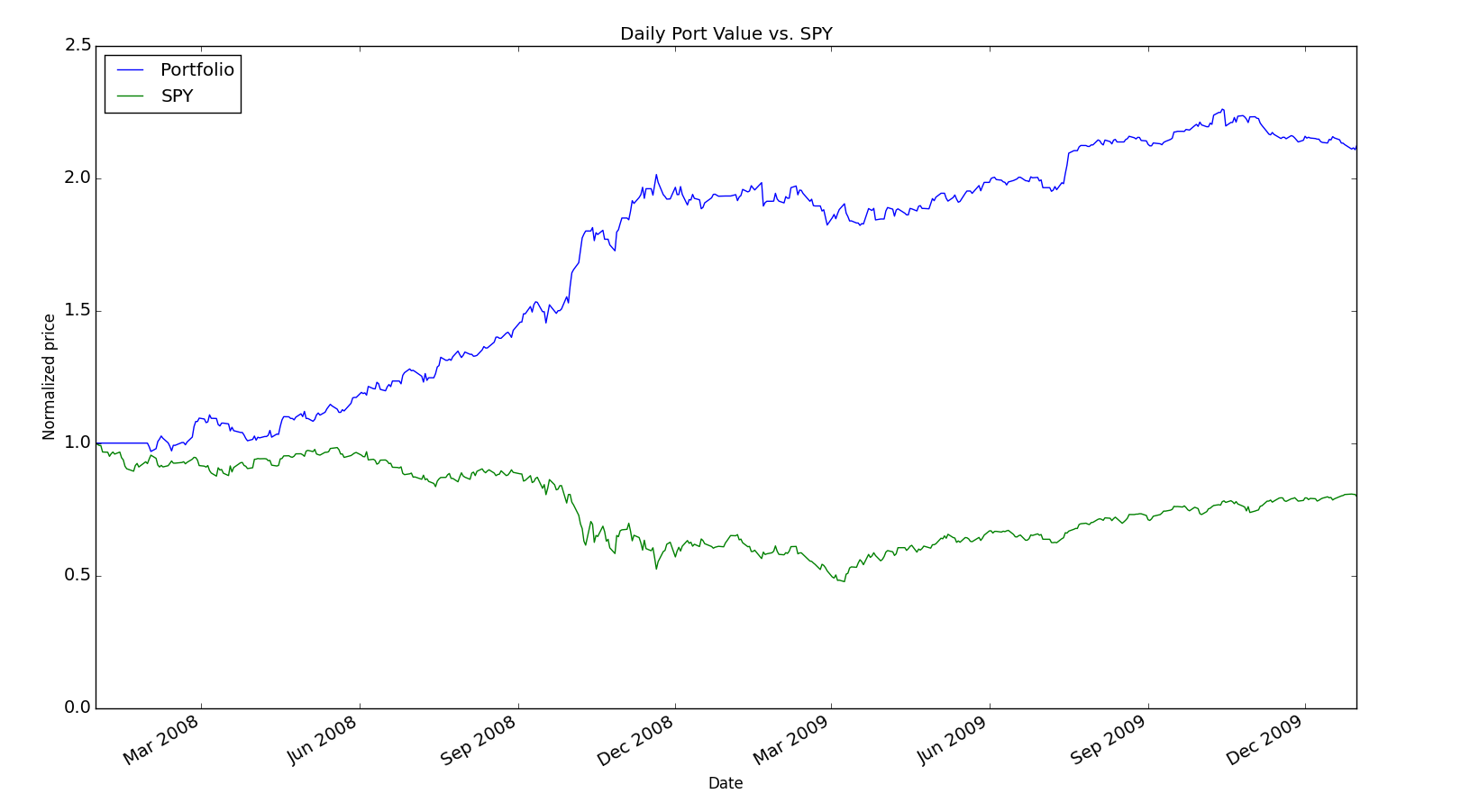
**Plot 6:** IBM Data Entries/Exits (In Sample)

Green Bar = long entry Red Bar = short entry Black Bar = short/long exit



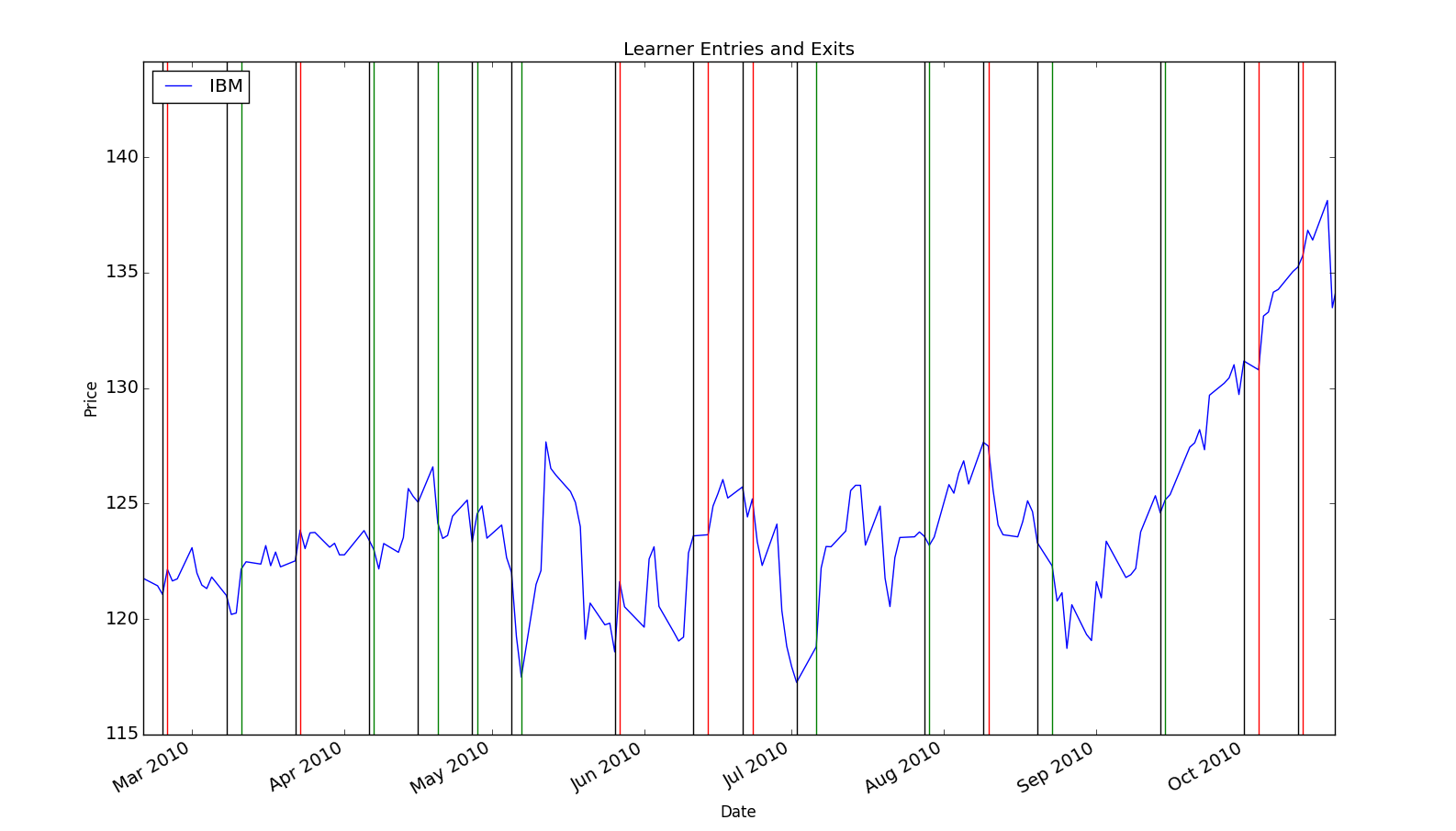
**Plot 7:** IBM Data Backtest (In Sample)

Green Line = SPY Blue Line = Portfolio



**Plot 8:** IBM Data Entries/Exits (Out of Sample)

Green Bar = long entry Red Bar = short entry Black Bar = short/long exit



**Plot 9:** IBM Data Backtest (Out of Sample)

Green Line = SPY Blue Line = Portfolio

