# An Evaluation of LSTM-based RNNs for Predicting Stock Price Movement

Mini Project 1

Kevin Flansburg
*Georgia Institute of Technology*
*kflansburg3@gatech.edu*

## 1 Introduction

The goal of this report is to explore the ability of neural networks to predict stock price movement. In particular, we focus on Long Short-Term Memory based Recurrent Neural Networks. The focus of this paper is to look at methods for reducing overfitting, a problem common to both stock prediction and RNNs, as well as evaluating possible portfolio performance for various trading strategies based on these predictions.

### 1.1 Problem Outline

## 2 Data

In order to train and evaluate the model, a large dataset of stock price movements, each with a window of history, must be constructed. We focus on S&P 500 symbols, in order to easily compare portfolio performance to the index.

We first identify the symbols which were on the index from January 1, 2008 through December 31, 2012. There are 404 symbols, of which 400 traded on every day that the S&P 500 did.

### 2.1 Annotating Data

We collect the High, Low, Volume and Adjusted Close for each of these symbols for this time period. At this point, any additional annotations, such as Moving Average, can be added. For this investigation we focus only on the four basic features provided by the dataset. In order to reduce the correlation between High, Low, and Adjusted Close, the High/Low columns are modified to be the difference between High/Low and Adjusted Close. The target value for each sample is taken as the percent change between the current and next adjusted value.

Table 1: Summary of Training/Testing Samples

| Seq. Len. | Training | Testing |
|---|---|---|
| 5 | 75,200 | 18,800 |
| 10 | 37,600 | 9,200 |
| 25 | 14,800 | 3,600 |

### 2.2 Training and Testing Sets

The data is next split into training and testing sets. We select 80 percent of the data set as training data, splitting the dataset in time such that the training samples precede the testing samples.

### 2.3 Rolling Window

Each input to the RNN must contain a certain amount of historical values for the sample. To discourage overfitting, samples are produced from non-overlapping windows of data, containing either 5, 10, or 25 historical values (sequence length). This will allow us to compare the performance of the model with more or less historical information.

### 2.4 Normalization

In order to normalize inputs from different symbols, each sample sequence is normalized by the first nonzero data point seen. Our method for decorrelating High and Low columns has a side effect of producing a lot of zero values. Finally, each input sample is normalized to have zero mean and unit variance.

Table 1 lists the number of training and testing samples obtained for various sequence lengths.

## 3 Network Design

To bound the scope of this experiment, we will limit ourselves to one simple network design. The general intu-

Table 2: Summary of Neural Network

| Layer Type | Output Shape | Activation |
|---|---|---|
| LSTM | L × n | TANH |
| LSTM | L × n | TANH |
| Mean Pool | n | NONE |
| DENSE | n | RELU |
| DENSE | n | RELU |
| DENSE | 1 | TANH |

ition behind our design is to extract high level time series features from two LSTM layers, then apply an average pooling layer to generalize these features over our sample window, followed by several dense layers to further extract features. We finally output a single value, with a hyperbolic tangent activation function, to output a direction and confidence between -1 and 1 of the percent change of the asset's value in the next time step. Table 2 summarizes the network. Here, *L* refers to the number of historical data points (sequence length) shown to the network for a given sample, and *n* refers to the number of features from each datapoint (4 for our experiments).

## 4 Training

A common problem in stock price models is overfitting on training data. With the model, we want to capture features that identify general stock movements, and are not specific to our training set. We address this problem in two ways. First we identify training behavior that we expect to see when there is no overtraining, and second we select the model design and training process with overfitting in mind.

### 4.1 Characteristics of Overtraining

To evaluate the performance of a given training process, we graph training and test loss (i.e. the RMSE error over the training and testing datasets respectively). Due to the nature of gradient descent, we can generally expect training loss to generally decrease and then remain fairly constant, when the step size of the optimizer is correct. When overtraining, we expect the test loss to fluctuate wildly, and so we want to find training parameters such that the test loss is consistent after initial training, and we do not have to worry about overtraining.

Therefore, we will evaluate the degree of overfitting by training the model for several epochs, and then evaluating the variance in test loss over several more epochs, once the training loss has generally converged. We also look at the mean test loss over this period to evaluate the expected performance of the model.

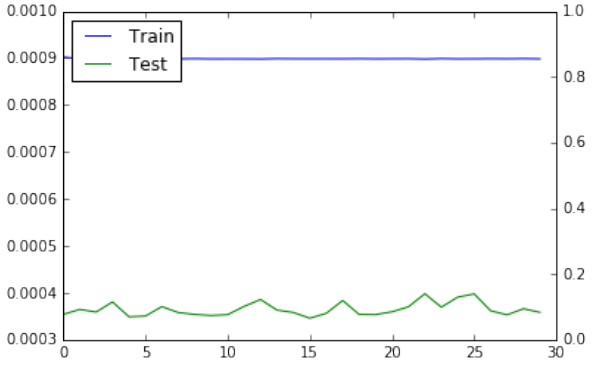


Figure 1: Training and Test Loss by Epoch with Overfitting (Batch Size 100)

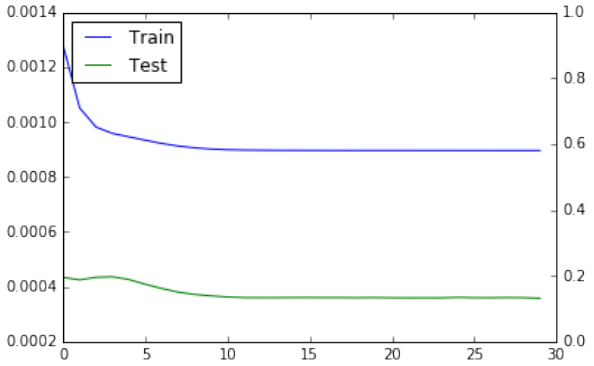


Figure 2: Training and Test Loss by Epoch without Overfitting (Batch Size 16000)

Figures 1 and 2 compare the test and training loss by epoch with and without overfitting respectively.

### 4.2 Training Strategy

We select basic batch learning for training the model. To reduce overfitting, we chose to train in very large batches. This can decrease training rate considerably, but we found that in general learning still occurred quickly. Our intuition here is that by training in large batches, we average out the localized features of stock indicators.

There are numerous methods for preventing overtraining in model design, including dropout and regularization layers. Here, we opt to simply use a small number of hidden units, to reduce the complexity of the model that can be fit.

We experiment with the number of hidden units, and training batch size. We first select a sequence length, number of hidden units, and batch size. We then train for 30 epochs, and then note the variance of the test loss over the last 10 epochs. Fig 4 (at end) summarizes these experiments, including the batch size found and the mean

test loss for that number of hidden units and sequence length. Values represent the average of three test runs.

The results are not extremely conclusive, however we note some general trends.

- In most cases, the simpler model ($HIDDEN = n$) performs better in terms of both mean and variance.
- A shorter sequence size performs generally better in terms of mean and variance, and has more stable performance as batch size varies.
- Some cases perform well for small batch sizes, but for the simple, short-sequence model, overfitting is generally reduced with larger batches while performance is reduced only slightly.
- Training loss is *greater* than test loss. This is not the case with even smaller batch sizes, but we prefer to err on the size of no overfitting.

Given these results, we select a model of sequence length 5, $n$ hidden units, and batch size $32,000$. We also note that there is a particularly well performing configuration of sequence length 25, $n$ hidden units, and batch size $1,000$. For the portfolio evaluation we will therefore evaluate this model as well.

# 5 Portfolio Performance

To evaluate our model, we simulate the portfolio performance when selecting stocks based on their projected movement. We compare this performance to the S&P 500.

## 5.1 Dataset

To produce this simulation, we use the test period, but with overlaping windows so that a predication can be made at each time step. We also keep each symbols data separate, so that a prediction can be made for each symbol at each time step.

## 5.2 Portfolio Selection

Based on the output of the model, there are several strategies for picking a portfolio. Figure 3 shows the predicted and actual percent movements for a single stock symbol over the simulation period. We notice that the prediction is biased below zero. We therefore propose several dimensions to explore:

### 5.2.1 Buy vs. Short vs. Both

We simulate the portfolio when only buying, shorting, or both. This looks to capture if the model does a better job of predicting upward or downward movement.

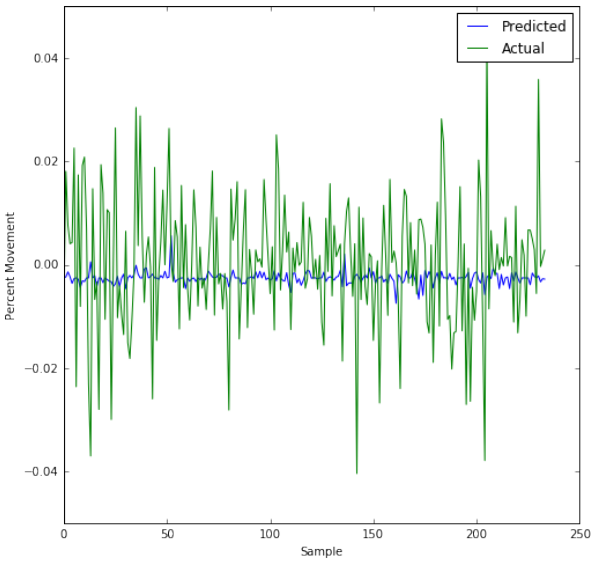


Figure 3: Predicted vs. Actual Movement

### 5.2.2 Uniform vs. Predicted Allocations

We look at allocating our portfolio uniformly (i.e. equally purchase all stocks which pass our confidence cutoff) or based on the confidence level. This captures how much fine detail we get by looking at confidence values.

### 5.2.3 Confidence

To select our portfolios, we look at a confidence cutoff, which is based on standard deviations from mean prediction for that time step. This in some ways tunes the aggressiveness of the model to act on its predictions.

# 6 Results

## 6.1 Sequence Length 5

Figures 5 - 8 (at end) illustrate the portfolio performance for various buying strategies.

### 6.1.1 Buy Only

The buy only strategy generally tracks the market. When a higher confidence cutoff is used, the portfolio outperforms the market quite well. In general it seems that it is best to ignore low confidence trades.

By allocating based on predicted values, less aggressive portfolios do slightly better.

#### 6.1.2 Short Only

This strategy only allows shorts in the portfolio, implying some cash balance as well. We see that the model does not have much predictive power here, and pretty much tracks a negative index, with more aggressive portfolios doing worse. The story is pretty much the same for predicted allocations.

#### 6.1.3 Buy and Short

To make this simulation, the buy component and short component of the portfolio are scaled such that leverage is roughly 2.

It appears that, despite poor shorting performance, that the combination of the two produces much stronger portfolios. The addition of predicted values leads to even more impressive results, with each portfolio delivering well over market performance, and the use of low cutoffs seems to only improve performance.

### 6.2 Sequence Length 25

We run the same simulations for the sequence length 25 model, Figures 11 - 14 (at the end).

#### 6.2.1 Buy Only

For this model, the performance for a buy only strategy was fairly poor, only just tracking the S&P 500 in the best case. Using predicted values for allocation does not improve this.

#### 6.2.2 Short Only

This model does present some interesting results for a short only strategy, with several high-cutoff portfolios keeping up with the market by only shorting a small number of stocks. By using predicted values, even low-cutoff portfolios are able to perform nearly as well as the market.

#### 6.2.3 Buy and Short

By combining buying and shorting, high-cutoff portfolios do well, and by using predicted value allocations, our portfolio either tracks the market or better, except for our portfolio with highest cutoff.

### 6.3 Conclusion

These results present some interesting conclusions. The short term model performed well for upward predictions, but poorly for downward predictions. The long-term model performed mediocre for upward predictions, and quite well for downward predictions.

This indicates that our general techniques for preventing overfitting may be too aggressive, preventing good prediction in both directions. Further exploration of model structure could produce better predictive capabilities in both directions.

Despite this, both models produced strong portfolios when combining both buying and shorting, with the simpler model performing significantly better. It appears that even better performance could be had by setting cutoffs for buying and shorting individually, to match the model's strengths. We do not cover this however, as these values may be highly dependent on the period of time the simulation is run over. It is clear that even very "simple" models, with very minimal training can have high predictive power, and may have an advantage over more complicated models.

Over all of the simulations, the portfolio never includes more than about 250 stocks. Indeed, it appears that predictions for certain stocks are always zero.

Selection of cutoff is tricky, because performance for predicting a single movement seems much higher when using a high cutoff, but combined buy/short models performed well for lower cutoffs, and in some cases the cutoff could be too high. This value would want to be selected to balance risk and performance, and evaluated over a large range of dates. It's also worth noting that cutoffs had wildly different Sharpe ratios, indicating that some would be more desirable from a risk standpoint.

The overfitting reduction techniques discussed appear to have resulted in a more consistent model, and not degraded its performance compared the model trained with smaller batch sizes, indicating that they may be effective for future training of such models.

The best results outperformed the market by over 100 %, however typical "good" performance for the model was about 10-20 % over the market. For some strategies, it appears that one could expect to track the market or better. There is a lot of room for improvement here, but the difficulty of training and debugging the model may make it not attractive, and at this point it is impossible to tell if this kind of performance could be consistently seen.

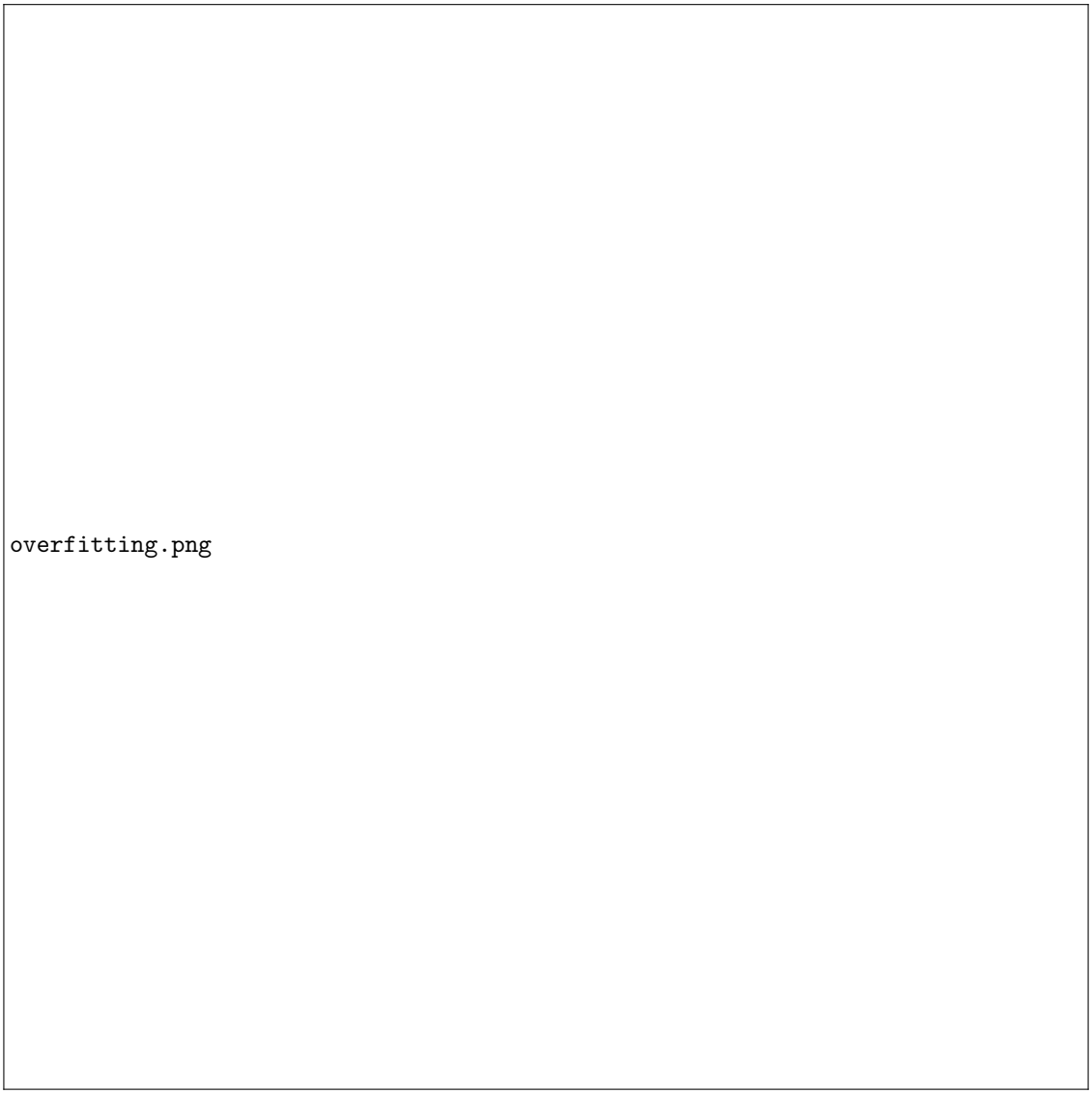


Figure 4: Test Loss Mean and Variance by Batch Size, Sequence Length, and Hidden Units

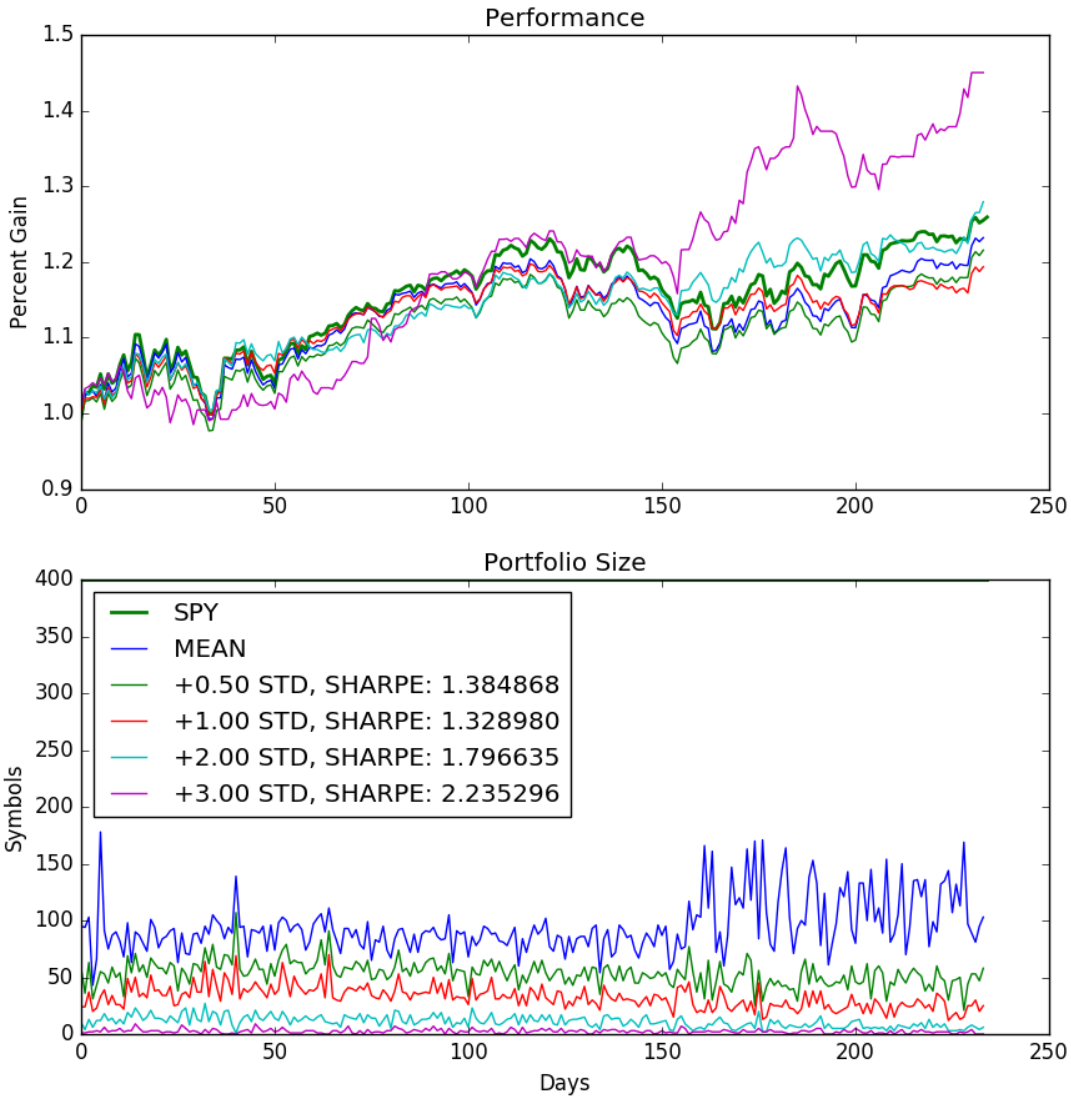


Figure 5: Buy Only, Uniform

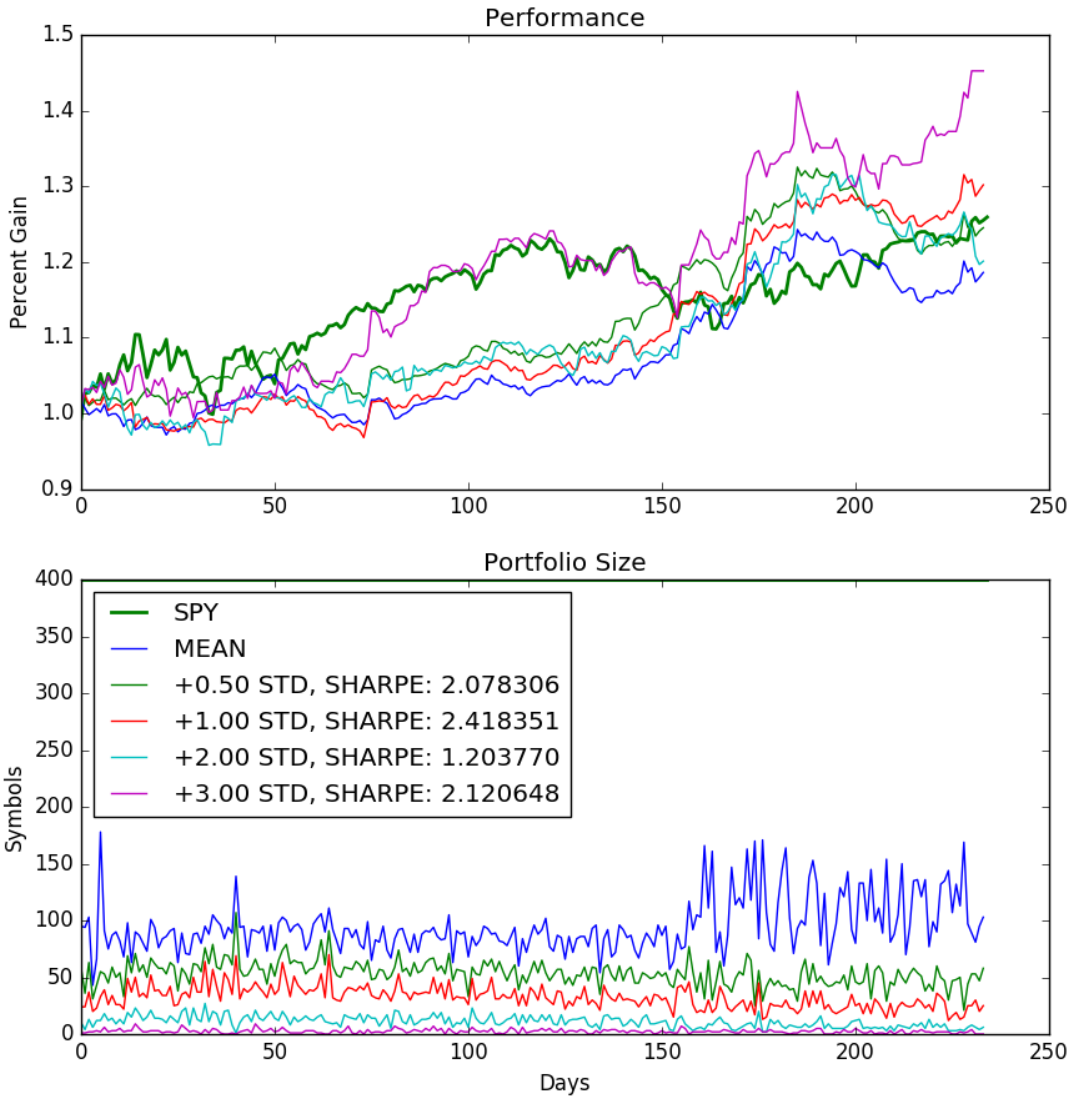


Figure 6: Buy Only, Predicted

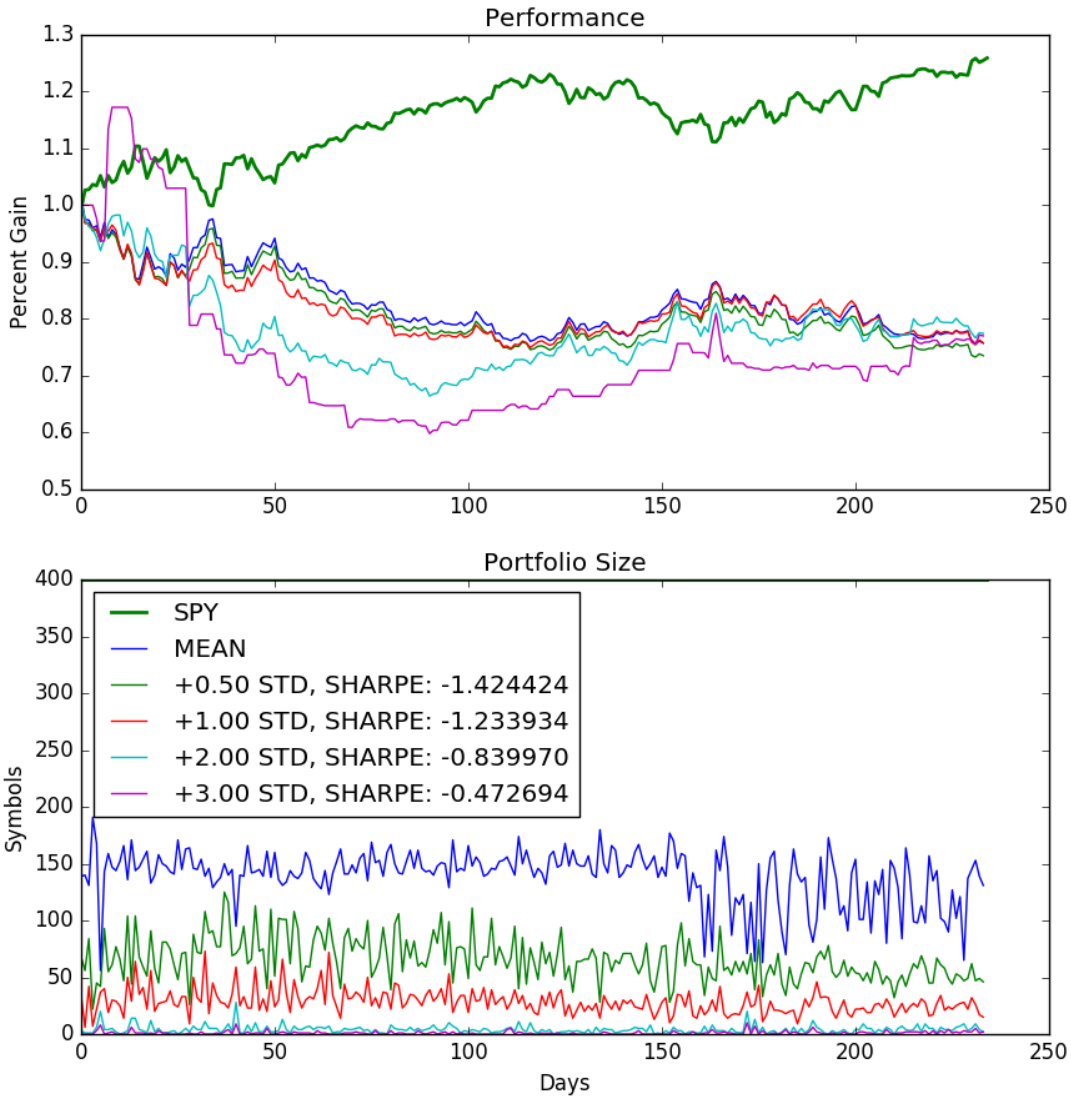


Figure 7: Short Only, Uniform

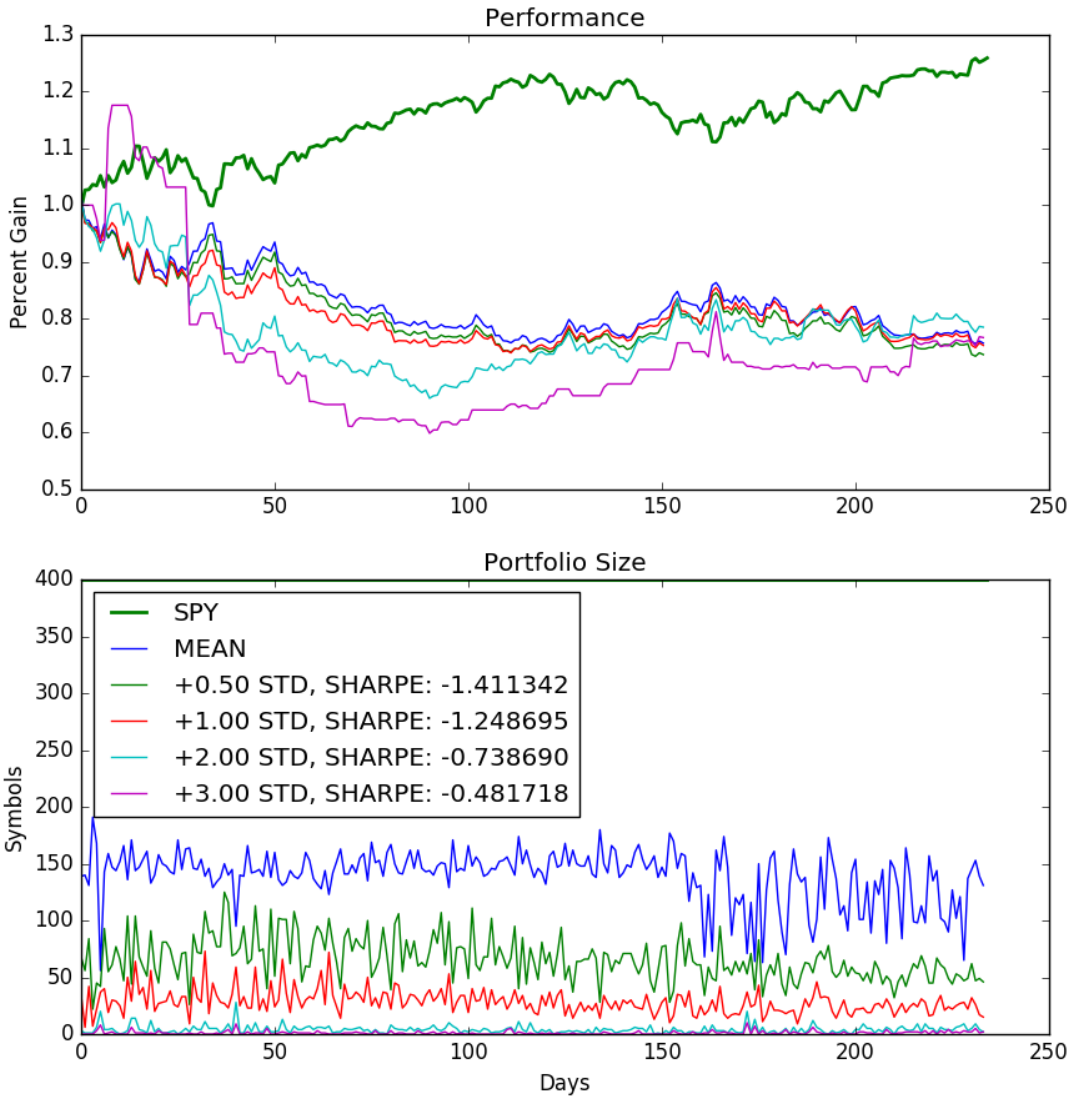


Figure 8: Short Only, Predicted

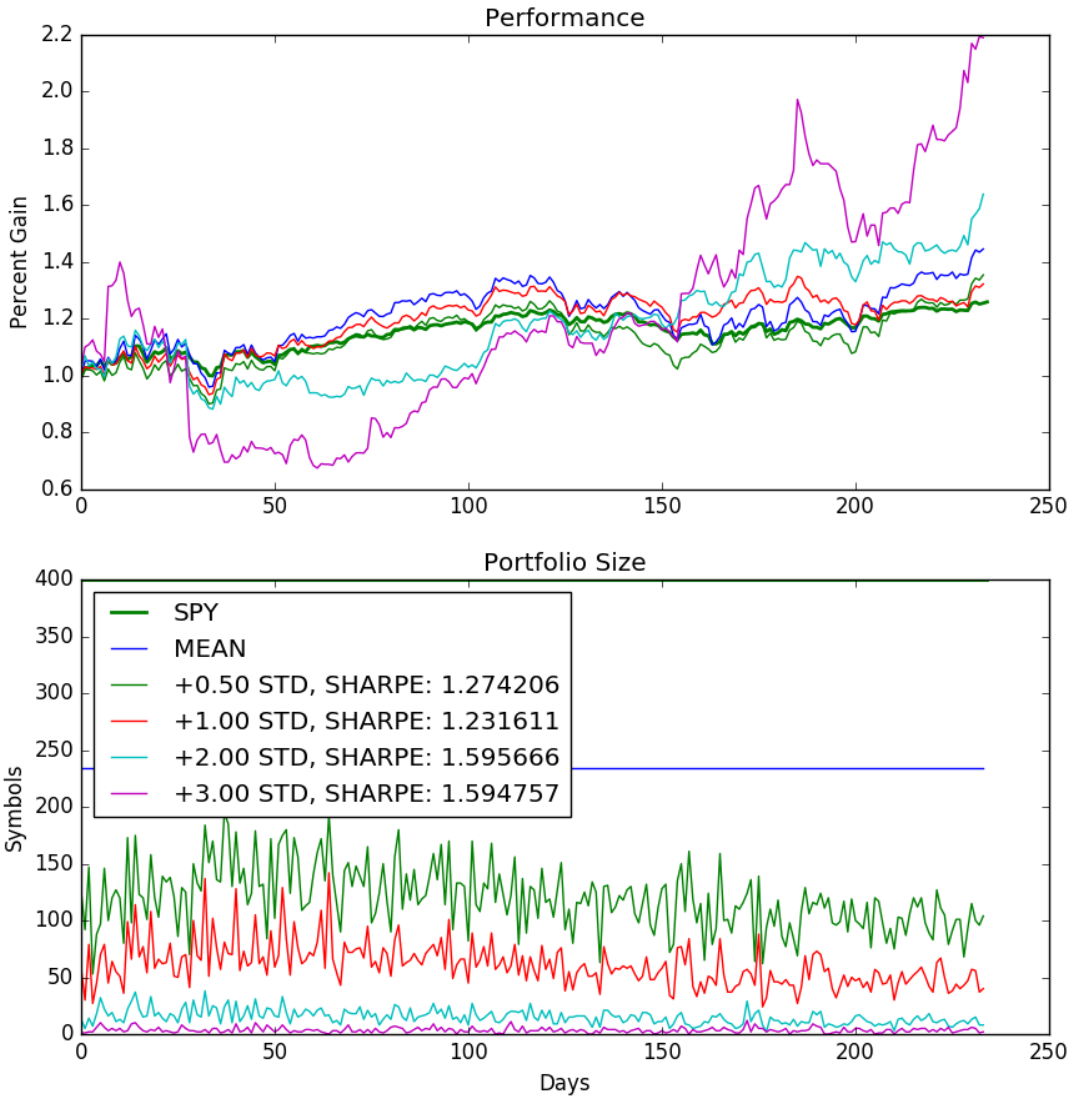


Figure 9: Buy and Short, Uniform

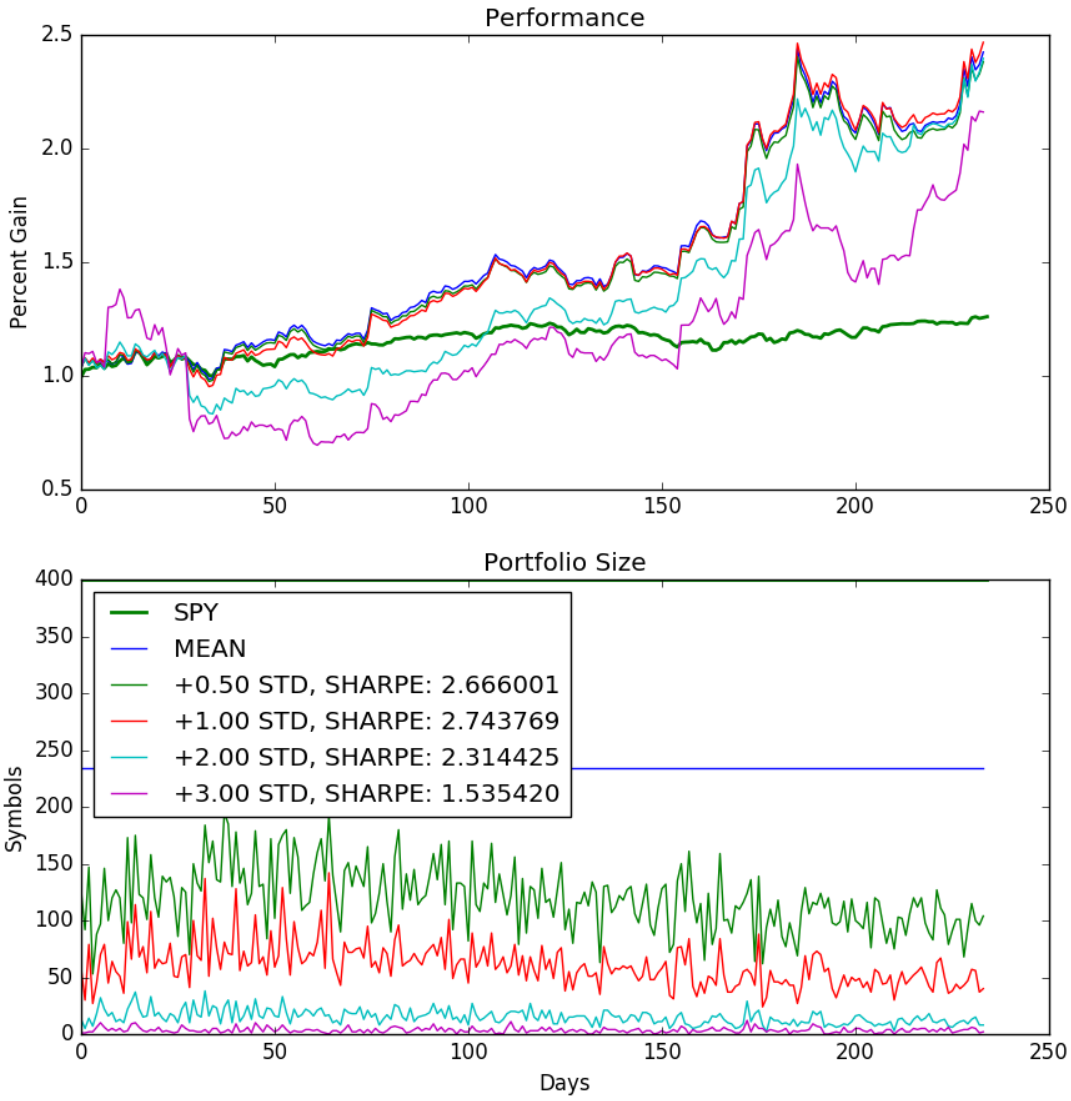


Figure 10: Buy and Short, Predicted

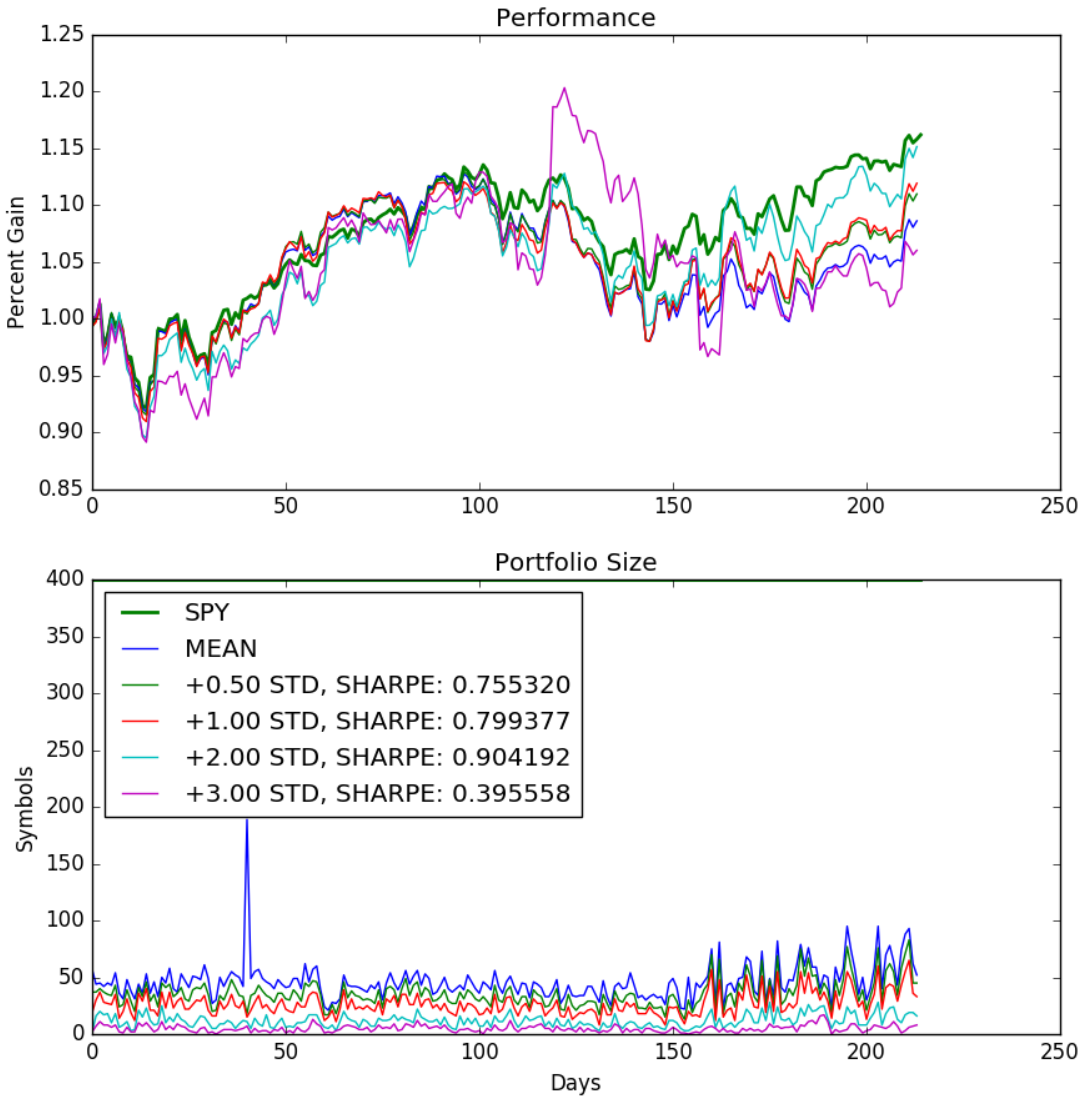


Figure 11: Buy Only, Uniform, Sequence Length 25

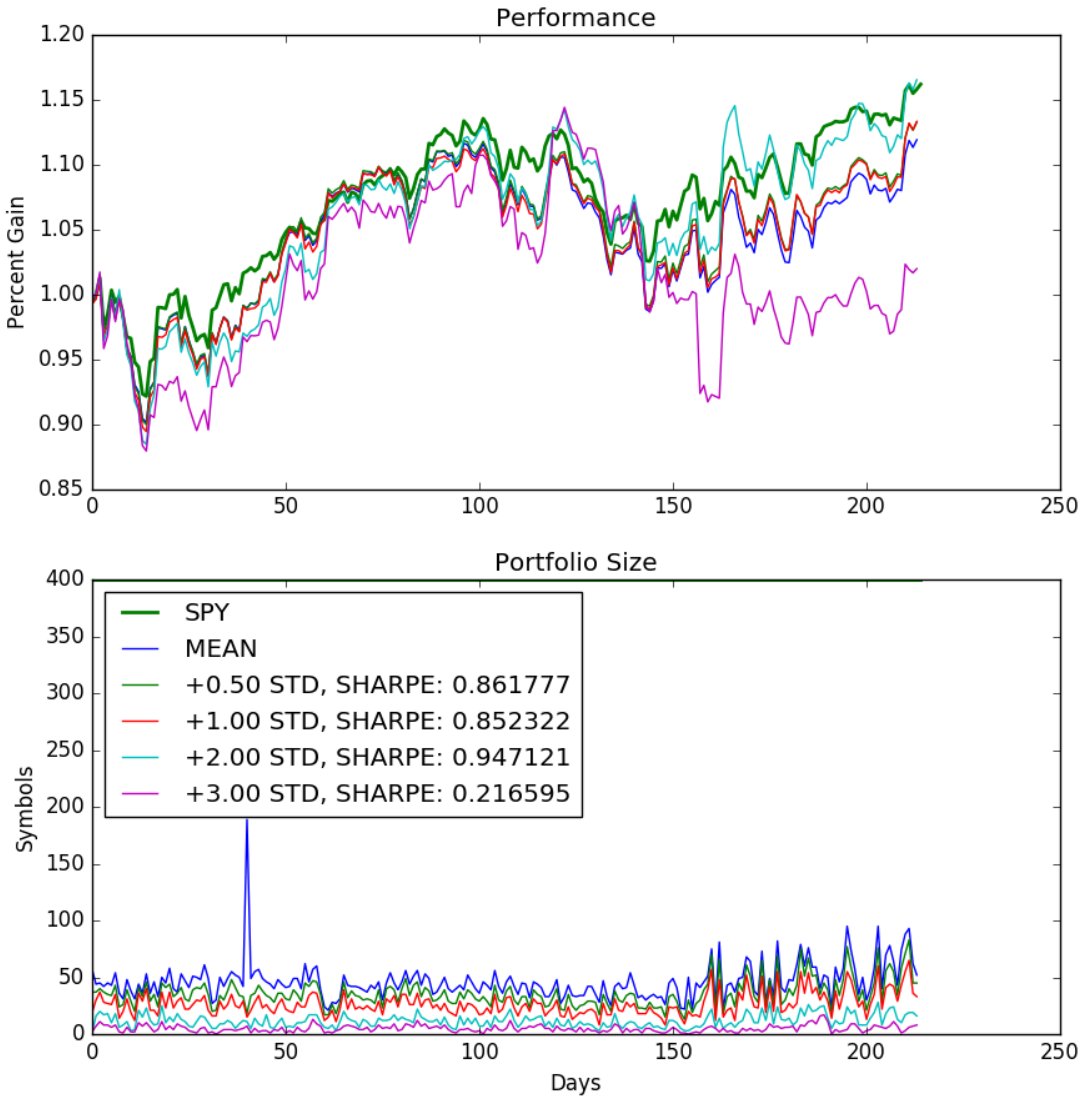


Figure 12: Buy Only, Predicted, Sequence Length 25

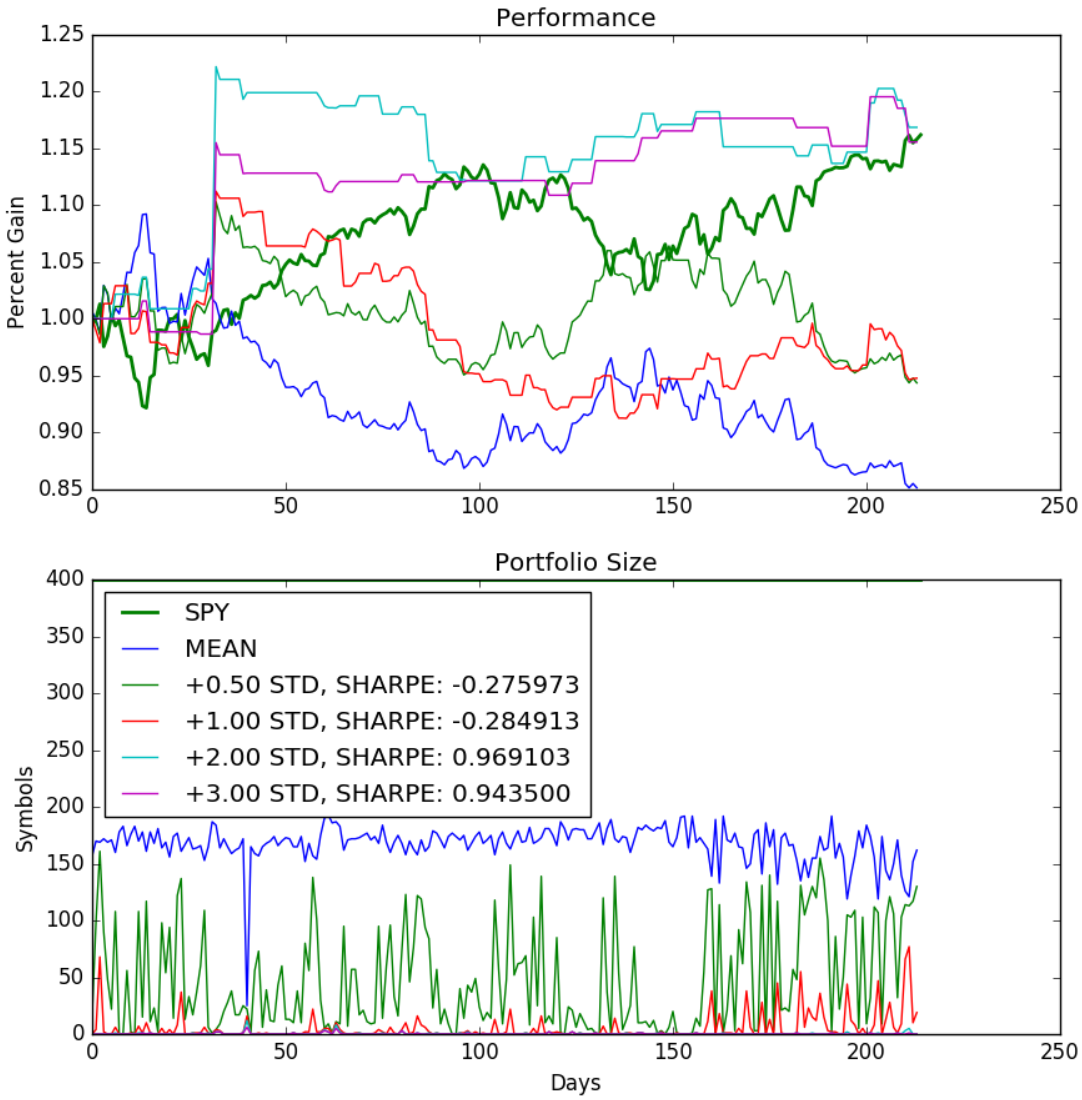


Figure 13: Short Only, Uniform, Sequence Length 25

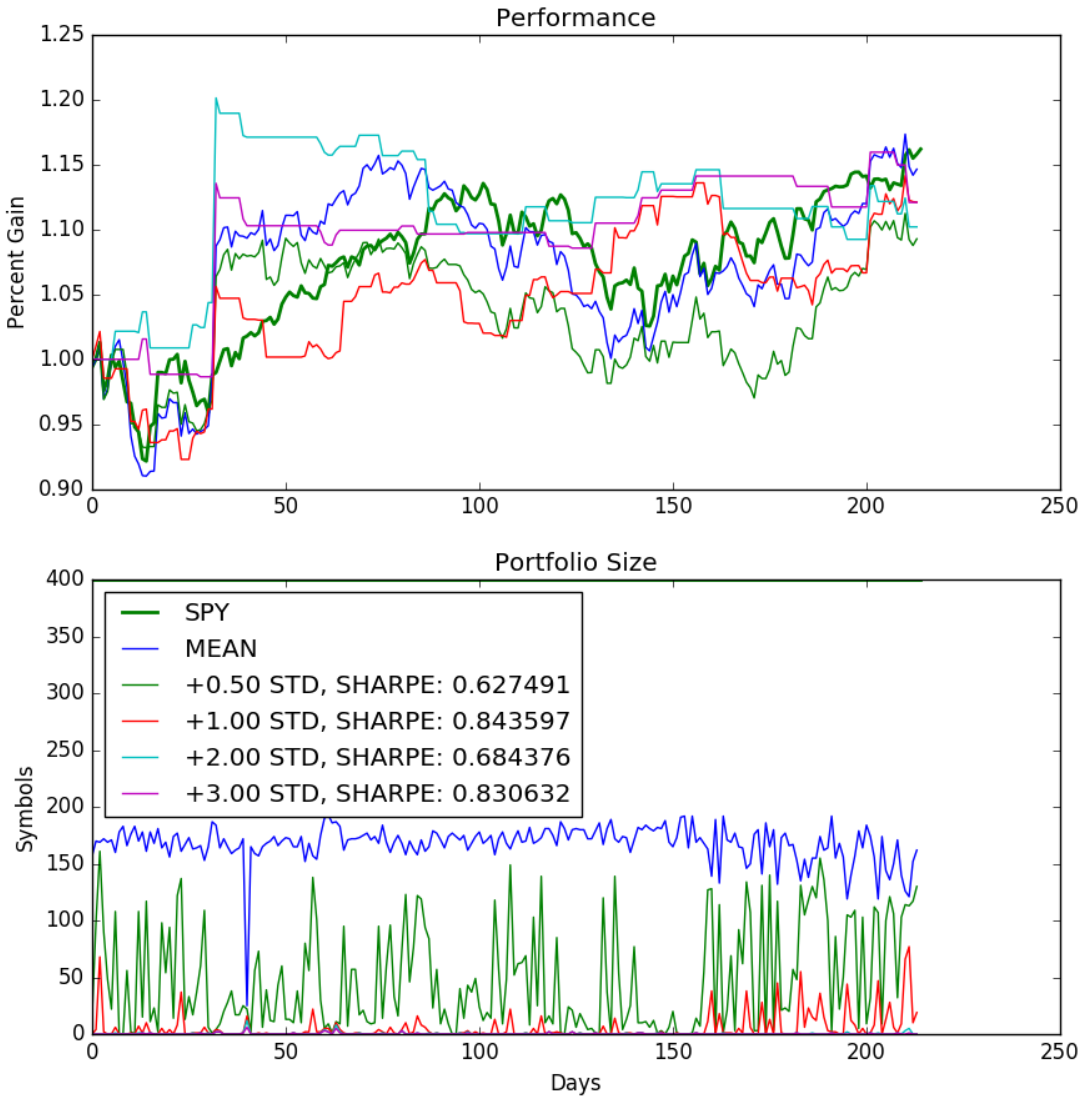


Figure 14: Short Only, Predicted, Sequence Length 25

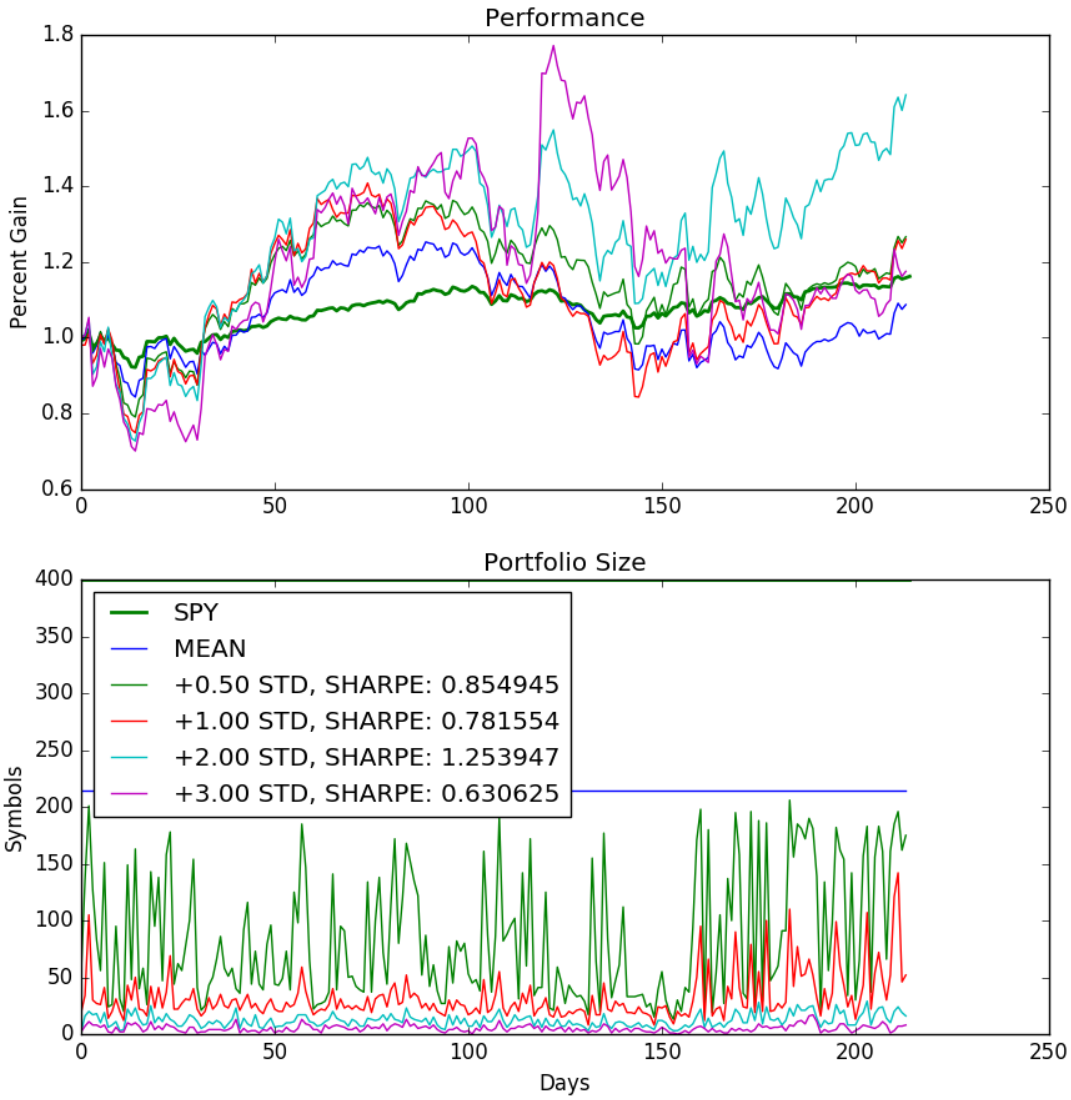


Figure 15: Buy and Short, Uniform, Sequence Length 25

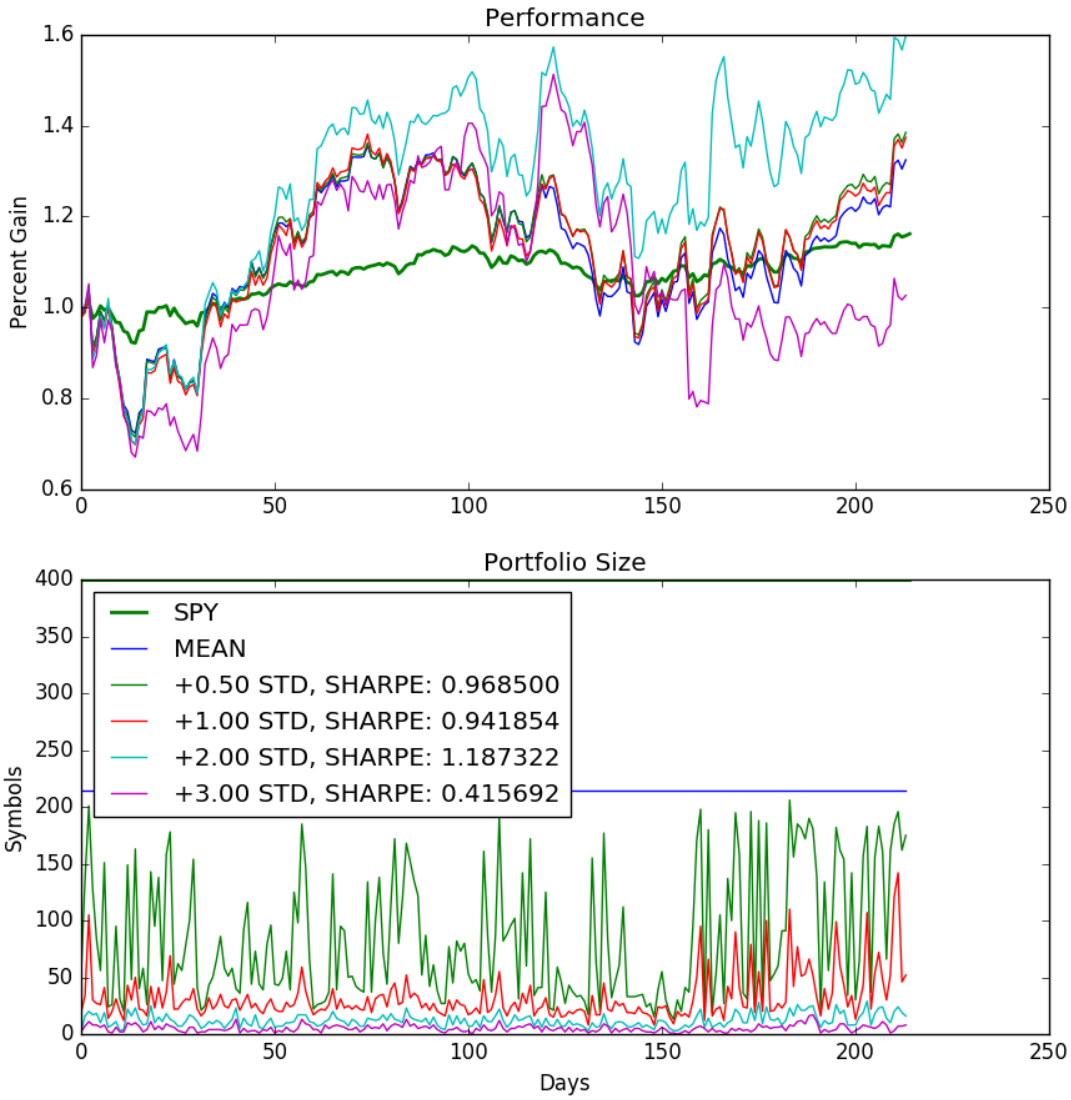


Figure 16: Buy and Short, Predicted, Sequence Length 25