Gaussian Process Regression Models for Predicting Stock Trends

M. Todd Farrell* Andrew Correa†

December 5, 2007

1 Introduction

Historical stock price data is a massive amount of time-series data with little-to-no noise. From all this relatively clean data it should be possible to predict accurate estimates of future stock prices. A number of different supervised learning methods have been tried to predict future stock prices, both for possible monetary gain and because it is an interesting research question. Examples of recent research are K. G. Srinivasa et al. [3] who used a neuro - genetic algorithm, M. A. Kaboudan [1] who used genetic programming, and Robert J. Yan et al. [5] who tried competetive learning techniques.

We use regression to capture changes in stock price prediction as a function of a covariance (kernel or Gram) matrix. For our purposes it is natural to think of the price of a stock as being some function over time. Loosely, a Gaussian processes can be cosidered to be defining a distrubution over functions with the inference step taking place directly in the space of functions [2]. Thus, by using GP to extend a function beyond known price data, we can predict whether stocks will rise or fall the next day, and by how much.

We conduct two experiments, both of which are carried out with data taken from 8 stocks[1] over the period of 10 years. Our experiments keep track of two important features of stocks: their price, and whether their prices are rising or falling. The former is much more important in many cases of stock trading since it is useful to know *by how much* you are winning or loosing money.

2 Method

Non-kernel regression problems try to predict a continuous response $y_t \in \mathbb{R}$ to some input vector $\underline{x}_t \in \mathbb{R}^d$. Kernel regression problems, on the other hand, map their input vectors into feature vectors via functions $\underline{\phi}(\underline{x}_t) \in \mathbb{R}^n$ for some feature expansion $\underline{\phi}(\cdot)$ and input vector $\underline{x}_t \in \mathbb{R}^d$. The continous predictions are $y_t = \underline{\theta}^T \cdot \underline{\phi}(\underline{x}_t) + \epsilon$ in this simplified model without bias and additive noise depends directly on the feature expansion used and thus the kernel $K = \underline{\phi}(\cdot)\underline{\phi}(\cdot)^T$. This is similar in the case of gaussian processes, and many of same rules for both linear and kernel regression apply.

There are two ways to view regression in the case of gaussian processes (GP), the weight-space view and function-space view [2]. We choose the function-space view.

Gaussian Process. *A Gaussian process is any collection of random variables where an arbitrary subset of variables have a joint Gaussian distribution.*

*mtfarrel@mit.edu
†acorrea@mit.edu

[1] These stocks are: adbe, adsk, erts, msft, orcl, sap, symc, and vrsn

A Gaussian distribution is entirely specified by the mean covariance and similarly a Gaussian distribution is completely specified by its mean function $m(\underline{x})$ and covariance function $k(\underline{x}, \underline{x}')$. That is,

$$m(\underline{x}) = \mathbb{E}[f(\underline{x})] = 0$$

and,

$$k(\underline{x}, \underline{x}') = \mathbb{E}[(f(\underline{x}) - m(\underline{x}))(f(\underline{x}' - m(\underline{x}')))] = K$$

We can write the Gaussian process in the form

$$f(\underline{x}) \sim \mathcal{GP}(m(\underline{x}), k(\underline{x}, \underline{x}')).$$

To simplify notation and calculation the mean $m(\underline{x})$ is taken to be 0[2]. Thus we have,

$$f(\underline{x}) \sim \mathcal{GP}(0, K)$$

The closing prices for stocks are assumed be be noise-free. This is a special case since most training data contains noise in realistic processes. In this case we know the pairs $\{(t, f_i^*(t)|t = 1, \ldots, n)\}$ for n training points, where t is the time at which we are evaluating the true price of the stock i, $f_i^*(t)$ at closing time.

There is a joint distribution over the training examples, f^* and the predictions $\hat{f}$. The prior is centered with mean on 0, so

$$\begin{bmatrix} f^* \\ \hat{f} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,\bar{X}) \\ K(\bar{X},X) & K(\bar{X},\bar{X}) \end{bmatrix}\right)$$

where the covariance term of X and $\bar{X}$ elements is an $n \times n'$ Gram matrix, X is the $n \times d$ set of training input vectors, and $\bar{X}$ is the $n' \times d$ set of test input vectors. Functions drawn from this joint prior distribution have the property that they agree with the observered training points. This limits the types of functions acceptable to make predictions on the data.

Function samples could be drawn from the distribution and matched with the training data points to see if there is a fit; however, for large data sets such as ours, this is computationally infeasible. Instead, consider calculating the conditional distribution for the values of $\hat{f}$ on the observations of $f^*, X, \bar{X}$. This takes the form,

$$\hat{f}|\bar{X}, X, f^* \sim \mathcal{N}(K(\bar{X},X)K(X,X)^{-1}f^*, K(\hat{X},\hat{X}) - K(\hat{X},X)K(X,X)^{-1}K(X,\hat{X}))$$

This is the noise-free prediction for $\hat{f}$ over the posterior [4].

2.1 Kernel Selection

For the stock price data set we chose to try several different kernels to see which ones produced the best fit of the training data. There are several types of kernel classes that can be used [2]. Below are the few considered in this project.

[2]Stock data does not have a mean of 0. It would not make sense, as stocks cannot have negative prices. However, we used a Gaussian that assumed a mean of 0 and a variance of our kernel (i.e. $\mathbf{f}\tilde{N}(0, K)$). We did this by shifting our stock price data such that its mean was 0. Not doing so would lead the GP process to believe that our predictive data's variance would be quite high, since the actual mean of a stock price is much higher than 0 (one would hope).

2.1.1 Squared Exponential Covariance

$$K(t,t') = \sigma_f^2 \exp\left(-\frac{(t-t')}{2\rho^2}\right)$$

where ρ is the characteristic length-scale of the covariance function and σ_f controls the "smoothness" of the curve. Squared Exponential Covariance is infinitely differentiable. This means there is some implicit assumption about how smooth the estimated functions are. That is, higher order noise will be smoothed over by this function.

2.1.2 Matern Class Covariance

$$K(t,t') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}(t-t')}{\rho}\right)^{\nu}$$

This covariance function is an alternative to the Squared Exponential Covariance (SE) function. While it can model higher order terms it is not infinitley differentiable and approximates the SE function as $\nu \to \infty$. In the our implementation we use only a specific case of this covariance function, where $\nu = \frac{3}{2}$ which is

$$K_{\nu=\frac{3}{2}}(t,t') = \left(1 + \frac{\sqrt{3}(t-t')}{\ell}\right)\exp\left(-\frac{\sqrt{3}(t-t')}{\ell}\right)$$

$K_{\nu=\frac{3}{2}}$ is more interesting in most machine learning applications since the value of ν is not too high. When ν is too high, the predicted function will have ver many jagged edges; on the other hand when it is too low the function will be too smooth. Either case makes unrealistic assumptions about $\hat{f}$, thus $\nu = \frac{3}{2}$ is a good value.

2.1.3 Rational Quadratic Covariance

$$k_{RQ}(t-t') = (1 + \frac{(t-t')^2}{2\alpha\ell^2})^{-\alpha}$$

The k_{RQ} covariance is the SE covariance function in the limit $\alpha \to \infty$. This is viewed as an infinite sum of SE covariance functions with different length scales.

2.1.4 Marginal Likelihood Gradient

Given a Kernel it is important to find the optimal set of hyperparameters fit the observed data. To do this, maximize the marginal likelihood,

$$\frac{\partial}{\partial\theta_j}\log p(\underline{y}|X,\underline{\theta}) = \frac{1}{2}tr\left(\left(\underline{\alpha}\underline{\alpha}^T - K^{-1}\right)\frac{\partial K}{\partial\theta_j}\right),$$

where $\underline{\alpha} = K^{-1}\underline{y}$ [2]. The inversion of the matrix K is very computationally intensive and of $\mathcal{O}(n^3)$ complexity. This is a large drawback, particularly for the large data sets such as ours. However, once the value of K^{-1} is known, the computational complexity drops to $(O)(n^2)$.

3 Experiments

We performed two experiments showing two different methods of stock prediction using Gaussian process. The first uses a ± 1 labeling to produce a simple metric of trend (where the label is $+1$ when rising and -1 when falling). The second uses regression on the value of the stock at the close of the trading day to predict the price of that stock for the following day(s). To fit the hyperparameters mentioned in Section 2.1 we used a 100 iteration optimization of the marginal liklehood with each different kernel.

3.1 Prediction of Up/Down Trend

In this experiment we see if we can predict stock prices by looking at only stock trend data (i.e. "Is this stock rising or falling?"). Prediction on this property is highly sensitive to the amount of training data used. As shown in the large time range in Figure 2 (from 10/06/2003 to 4/28/2006) the price fluctuates regularly and averages to a 50% chance that the predicted label, $\hat{y}$, will be $+1$ and a 50% chance it will be -1.

This particular prediction method is more useful on smaller data sets. Consider a Matern Class covariance function trained on one month of stock price data as in Figure 3. After regression is performed there is an indicaton for many stocks, take stock 6 and notice how the prediction curve is going down but remains above 0 indicating that it is becoming less sure there will be a gain in close price.

If the data is biased towards a $+1$ label then the next day's prediction is $+1$. As the variance increases for further predictions beyond the first day the trend moves to about half as likely to a $+1$ or -1. This is likely due to the uncertainty of our model to be able to predict a label effectively once the number of days passed has increased.

3.2 Single-Stock Price Prediction

This experiment consists of three parts: training data regression, test data price prediction, and prediction assessment. The model assumes that the training data has zero mean, so the training outputs were recentered by calculating

$$TrainingData = TrainingData - mean(TrainingData),$$

with the outputs centered about zero. After the means and variances are predicted we translate the data back. This gives results like those in Table 2.

Table 1: Experiment 2: Results

Time Period	Squared Exponential	Matern $\nu = 3/2$	Rational Quadratic
10 day	Stock 7, 0.24 regret	Stock 7, 0.24 regret	no stock predicted
1 month	Stock 7, 0.24 regret	Stock 7, 0.24 regret	Stock 7, 0.24
6 month	Stock 7, 0.24 regret	Stock 7, 0.24 regret	Stock 7, 0.24 regret
1 year	Stock 8, 0 regret	Stock 8, 0 regret	Stock 7, 0.24 regret
3 years	Stock 8, 0 regret	Stock 8, 0 regret	Stock 7, 0.24 regret

Table 2: Chosen stock paired with regret in Covariance function vs. time span table

We measured penalty in terms of regret. Since we had the actual next-day (day $t + 1$) prices of the test stock, we were able to find which stock would have been the best to invest in the day

before (day t). We then compared the difference in price between time t and $t+1$ for the "best" stock and the stock we predicted, and took the difference. In the case when our model predicted that no stock would gain any money the next day and no stock did, then we gave no penalty. If the model was wrong in this prediction, however, the penalty became the gain of the "best" stock, just as in the normal case.

4 Conclusions

The two experiments conducted here show the power of using Gaussian processes for predicting stock prices. There were a number of different ways to test how to make a good prediction based on the training data. By varying the lenghths of time considered for training, we achieved different results and accuracies. Using a long time period in experiment 2 resulted in good stock predictions, while using shorter times resulted in sub-optimal predictions. In experiment 1, on the other hand, converse was true.

The use of long time periods in the first experiment did not produce interesting results, however, because regression only predicted the mean value of 0 out of a long time period for predicting trend. For experiment 1 where the goal was to classify between two types of activity it was better to consider shorter time scales to avoid this averaging effect. A possible future direction is using a larger subset of trends rather than restricting ourselves to mere growth and loss trends.

The method in experiment 2 showed that a Matern class or Squared Exponential class covariance function predicted the optimal stock with less or equal regret compared with the rational quadratic covariance function. The Squared Exponential function makes some assumptions about the smoothness of the data as compared with the Matern Class covariance. The Rational Quadratic is a combination of Squared Exponential functions at different length scales. This must play a role in how ineffective this covariance function was at predicting the optimal stock pick. It seems that if a large enough training set is used the Rational Quadratic covariance will eventually give stock 8 as a result. However, there was not enough time to test this. Given the data it appears either the Matern Class or Squared Exponential class is appropiate for use in prediction.

Practially speaking however, using Gaussian process are not the most efficient method to use, because the computational load for predictions is high. To make money in the stock market using a machine learning technique, A faster method of computation would be required.

References

[1] M. A. Kaboudan. Genetic programming prediction of stock prices. *Comput. Econ.*, 16(3):207–236, 2000.

[2] C.E. Rasmussen and K. I. Williams. Gaussian processes for machine learning. 2006.

[3] K. G. Srinivasa, K. R. Venugopal, and L. M. Patnaik. An efficient fuzzy based neuro - genetic algorithm for stock market prediction. *Int. J. Hybrid Intell. Syst.*, 3(2):63–81, 2006.

[4] R. von Mises. Mathematical theory of probability and statistics. 1964.

[5] Robert J. Yan and Charles X. Ling. Machine learning for stock selection. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1038–1042, New York, NY, USA, 2007. ACM.

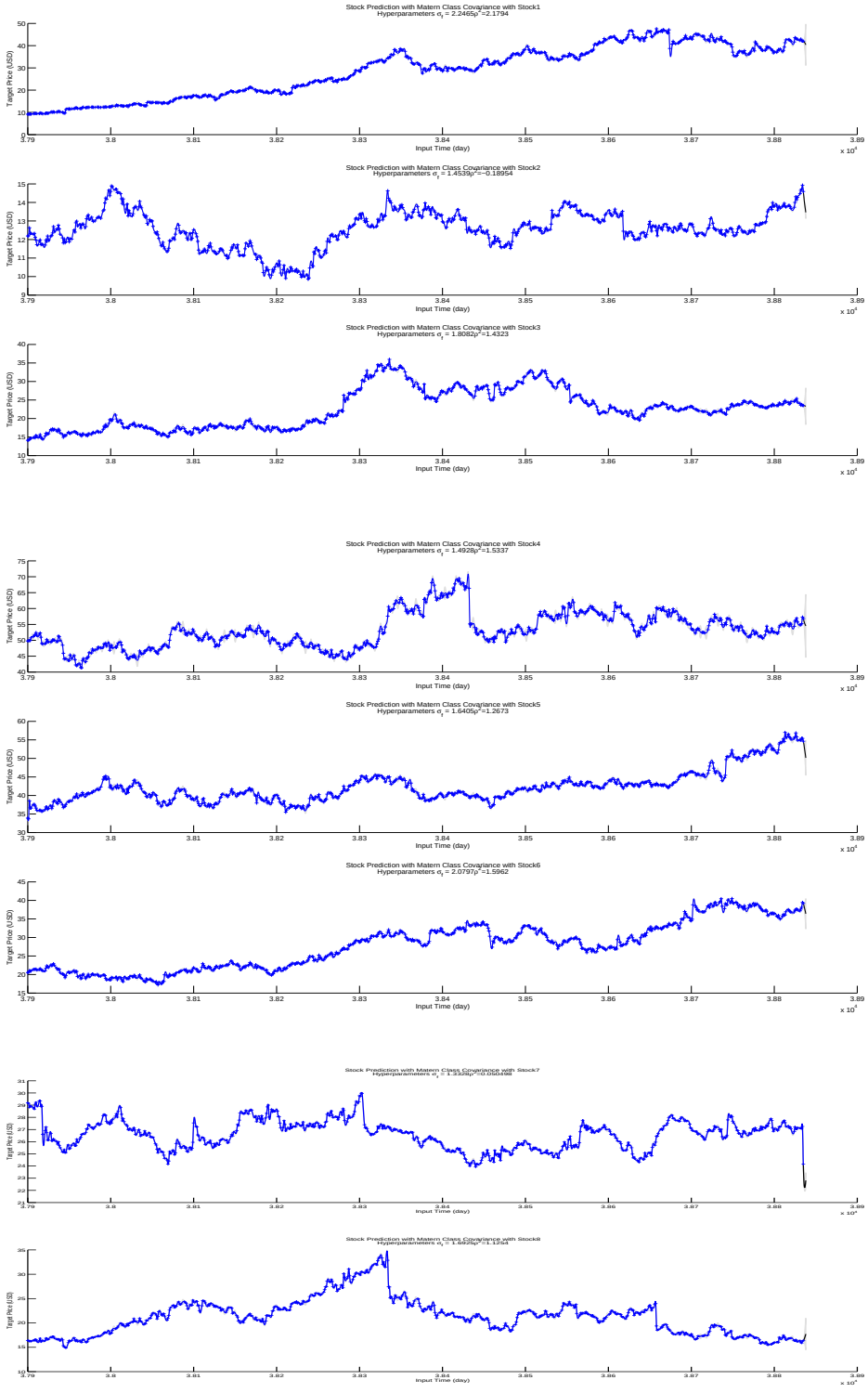

Figure 1: The above figures show the results of running a prediction on a large multi-year dataset.

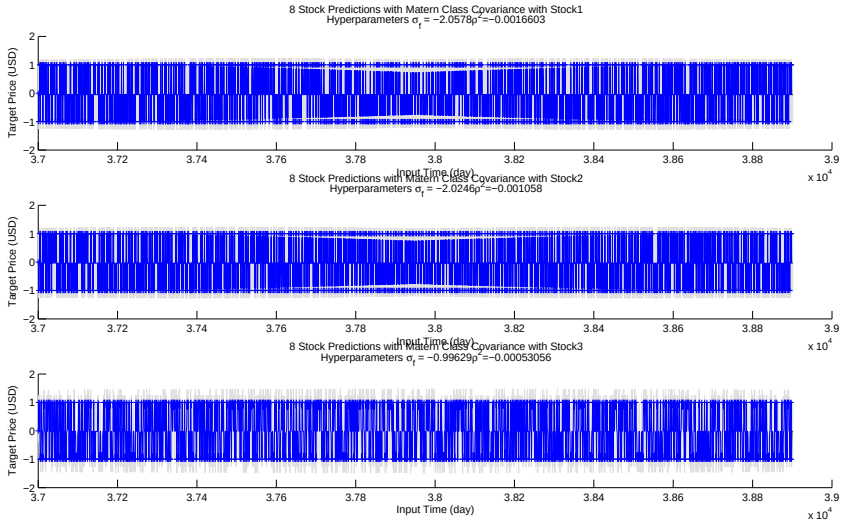

Figure 2: Running label classification on a large training set produces poor prediction. In a practical setting some stocks are highly regular and vary only a small amount for years at a time. This is reflected in the fact that the labels have what appears to be 50% chance of beeing labels +1 and 50% chace of being labeled -1 over time.

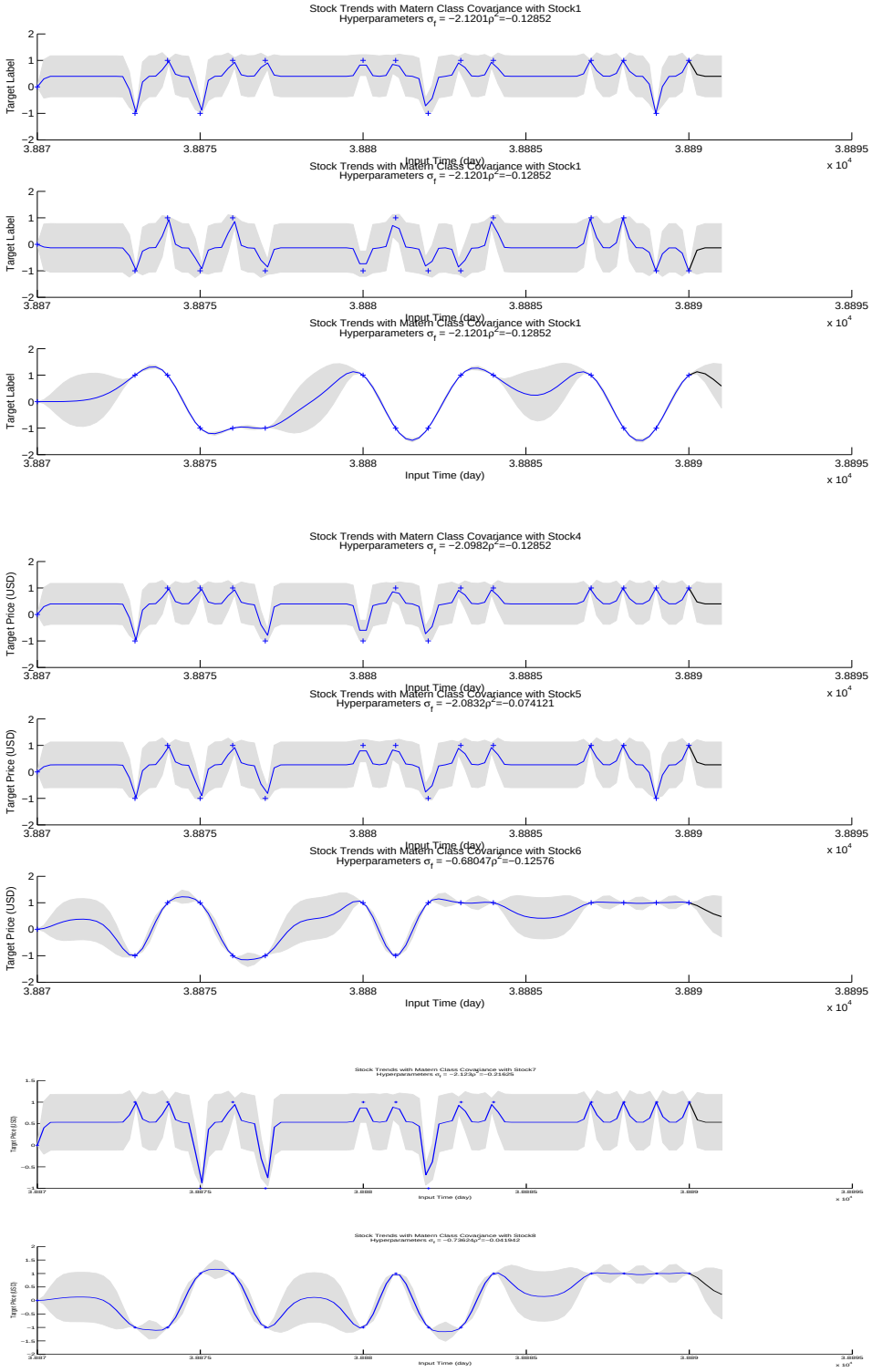

Figure 3: Running label classification on a smaller dataset produces more meaninful results. The 95% confidence intervals show a trend towards either a +1 or -1 label.

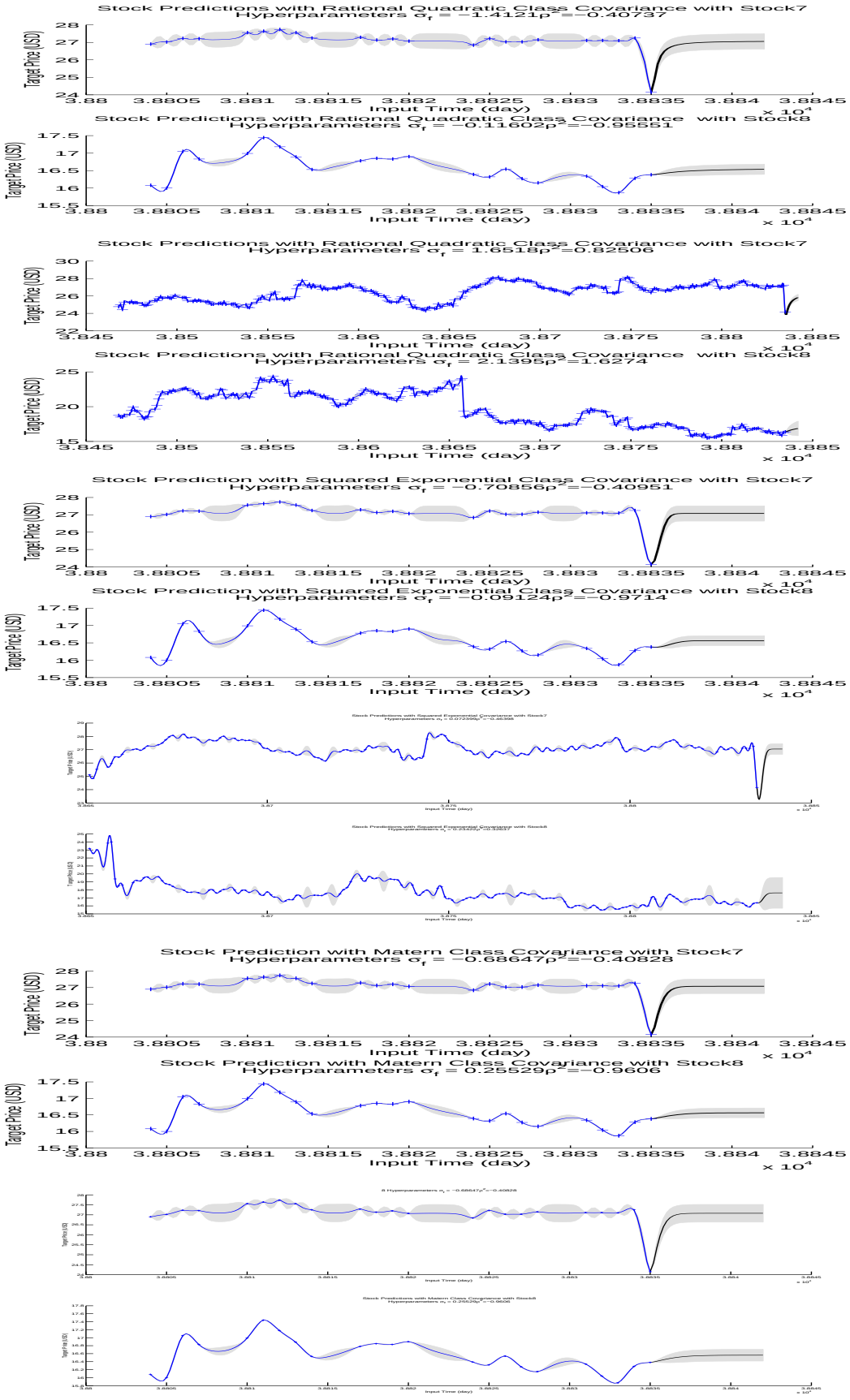

Figure 4: The comparison between stock 7 and stock 8. Both were picked by the model. choosing stock 7 caused a regreet of 0.24 and choosing stock 8 caused no loss.