# Machine Learning for Stock Return Prediction

Xiaokun Zhang
Department of Chemistry Emory University Atlanta, GA 30322, USA

**Abstract**
The problem of predicting future based on a pile of noisy time series data is complicated and difficult. In our project, we aim to predict the stock return trend from the datasets provided in The Winton Stock Market Challenge on Kaggle website. In the process of learning the stock return trend, a novel time series method will be brought up and we will also address the problem of noise interference entangled in the datasets.

*Keywords: Stock Return Prediction, Singular Value Decomposition, Stochastic Gradient Descent, Multiple Linear Regression*

## 1. Introduction

Predicting the stock return has always been an intriguing problem in financial analysis. The main hypothesis is that past data contain clues of future stock returns, which means that it is possible to predict the future movement of the stock returns through current and past financial data and other information about the companies. It is closely related to the prediction of stock price but it invloves more complexity. After traders set up the position, their return ratios are dependent on the change of stock prices and the types of positions traders hold. If we can predict the stock return ratios, we can adjust the positions and maximize the profit. For example, if the model predicts GOOGL will produce a 10% return with long position, then it gives traders the hint to buy GOOGL's stock. In contrast, if the model predicts NFLX might drop 5%, traders who hold its stocks can raise their stop loss or even clear up their long positions. Compared with predicting stock prices, it is more practical and prone to use.

The essential part of predicting stock return is to model time series observations, which has inspired many creative and reliable methods with its challenging nature. To predict time series data, three models, the ARIMA model, temporal features model, and rolling windows based regression, are mostly used. For the ARIMA model, the time series is broken into trend component, seasonality component etc, and each component is estimated. The model parameters need to be calibrated by experts. The temporal features model takes advantage of the auto correlation information so that the temporal aspects are considered in the model. Features include mathematical measures such as entropy, time between two events, collection between moving averages. For this temporal feature model, the choice of features also requires several techniques. The rolling window is to predict $X(t+1)$ not only with $X(t)$ but also $X(t-1)$ and $X(t-2)$ etc. So the data are transformed to the format with the features (the rolling windows) are a time sequence (Zhang (2003)). These models are only based on the moving of time steps, which do not involve other features. However, several features are likely to be crucial to a company. For example, the price-earnings ratio (P/E) and earnings per share (EPS) are usually used as criteria to judge the performance of a company during the earning seasons. Beating the estimated EPS with lower P/E will trigger the increase the stock price in most cases.

In this paper, we have proposed a two-step method to predict both the intraday return ratios as well as the daily return ratios. In both steps, we use an ensemble model that contains MLR, SGD Regressor and Decision Tree regressor to model the relation between inputs and outputs. In the first step, we propose a window method and we use this ensemble model to predict intraday return ratios recursively. The size of window is chosen as 10 to minimize the relative errors and achieve the best performance. In the second step, we use PCA firstly to drop uncessary features and use the previous ensemble model again to predict the daily return. For this step, the inputs combine the left features, previous daily returns and the known and unknown intraday returns. The outputs are only two variables, the return for tomorrow and the return for the day after tomorrow. By using this model, we predict the future daily return both based on the time series data and the features of company, which is a more realistic situation compared with using only time series data. Furthermore, we analyze pros and cons of this method and compare the method with autoregressivemoving-average (ARMA) model. Numerical experiments show that the method that we propose is more reliable and has a lower computational cost.

## 2. Background

## 2.1 Generative Models

### 2.1.1 Multiple Linear Regression (MLR)

Multiple linear regression attempts to model the relationship between two or more explana- tory variables and a response variable by fitting a linear equation to observed data. The value of the variable y is dependent on the list of the independent variables X by the factor β. This relationship can be expressed as:

$$y = X\beta + \varepsilon,$$

where ε represents the noise. The regression equation for columns of X, X1, X2, ..., Xp, can be written as $y = \beta 0X0 + \beta 1X1 + \beta 2X2 + ... + \beta pXp + +\varepsilon$. This line describes how the mean response y changes with the explanatory variables. The choice of regressors might influence the output. The ordinary least squared model aims to minimize the squares of deviations from each data points to the line (Aiken et al. (2003)).

### 2.1.2 Stochastic Gradient Descent (SGD) Regressor

The stochastic gradient descent (SGD) regressor aims to minimize a regularized empirical loss with SGD. The empirical loss function is penalized by the regularizers, of which the two most common ones are squared Euclidean norm l2 or the abosulte norm l1. When the gradient of the loss function is estimated for each sample, the model is updated in order to decrease the learning rate (Bottou (2010)).

### 2.1.3 Multi-output Regression

Multi-output regression is the method to predict multiple output variables from a list of input. This method can combine with most single-output regression methods to predict multiple targets. Common methods are categorized into two approaches. One is to trans- form the problem into independent, single-output problems and each is solved separately. The second one is to adapt single-output algorithms directly to include the processing of multi-output data set. For the adaption method, the difficulty lies interpreting the depen- dencies among the targets (Borchani et al. (2015)).

### 2.1.4 Decision Tree Regression

The decision tree regression uses the decision tree as a predictive model. The method needs to automatically decide on the splitting variables, splitting points, and topology of the tree. For the data with p inputs and a response for each of N observations, if we suppose a splitting of M regions $R_1, R_2, ..., R_M$, then the response in each region can be modeled M as $f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$. In order to find the best splitting for the minimum sum of m=1 squares $\sum(y - f(x_i))^2$ , a greedy algorithm is used to determine the split point for each splitting variable. Then this process is repeated to split the binary areas further into more regions (Friedman et al. (2001)).

**2.2 Time Series Method**

2.2.1 Autoregressivemoving-average Model (ARMA)

To model this time series dependence, two components are taken into consideration. First, for a series Xt, we suppose that the current observation has a dependency on the past observations and model this dependency by an AR (autoregressive) model, which regresses on the past observations. We can define the length of the interaction by the order of the model, which we denote as p. The AR(p) model can be written as:

$$x_t = \varphi x_{t-1} + \varphi x_{t-1} + \cdots + \varphi x_{t-p} + \varepsilon_t.$$

Secondly, the current observations of a random variable is affected by both the current error terms and the past error terms, which can be represented by an MA (moving average of order q) model. The MA(q) can be written as:

$$x_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-p}.$$

If we combine the AR(p) model and MA(q) models, then we can obtain a ARMA(p,q) model:

$$x_t = \varphi x_{t-1} + \varphi x_{t-1} + \cdots + \varphi x_{t-p} + x_t + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-p}.$$

For the forecast of a random variable yt+h, if the criterion is to minimize the mean squared

error, then the best forecast is the conditional expectation (Wei (1994)).

**3. Data Description**

In this project, datasets are taken from a Kaggle contest sponsored by Winton Capital (Win). The datasets contain various market related data, and the goal is to forecast future unseen intraday and daily returns. The dataset has 40,000 observations (stocks), each of which includes 25 features, 2 intraday return ratios, 179 intraday returns. We separate this dataset into two parts, training and test sets. The size of training set is 30,000 and the size of test set is 10,000. For the training dataset, those features were referred to by their ID columns, which is a mix of 25 unlabeled continuous and discrete features, and 183 ordered time series returns. In the 183 ordered time series returns, these are −D1 return, −D2 return, 179 intraday returns (1min interval), +D1 return, and +D2 return, in which D1 and D2 returns were prior daily returns and +D1 and +D2 were post daily returns. The rest three features are weights for intraday return and +D1 and +D2 returns, which were not part of the prediction model but were used to estimate the errors of the model. The goal is to predict the later part of the intraday returns (60 returns) and post two-day returns (+D1 and +D2) based on the 25 features, −D1 return, −D2 return and the prior part of the intraday returns (119 returns) provided in the test datasets. For training set, the numeric properties are described as below:

- Feature 1 - Feature 25: they are either continuous or discrete numbers. They represent the P/E, EPS, Capitals and etc. for each stock.
- Ret MinusTwo, Ret MinusOne: they are continuous variables and they represent re- turns the previous days.
- Ret 2 - Ret 120 (intraday returns): they are continuous variables and they represent the known intraday returns.
- Ret 121 - Ret 180: target variables (intraday returns): they are continuous variables and they represent the unknown intraday returns.
- Ret PlusOne, Ret PlusTwo: target variables (+D1,+D2 returns): They are continu- ous variables and they are unknwon returns for tomorrow and the day after tomorrow.
- Weight Intraday, Weight Daily: Large continuous varibles and it can represent the position for each stock.

For the test set, the numeric properties are described as bellow:

- Feature 1 - Feature 25: the represent the features for test data, which have the same data properties as training data.
- Ret MinusTwo, Ret MinusOne: the previous daily returns are already known for test dataset.
- Ret 2 - Ret 120: the known intraday returns. We need to use these data to predict unknown intraday returns.

## 4. Data Preprocessing

For this part, three popular methods, dropping unnecessary features and filling missing values, SVD decomposition and scaling, are used to preprocess the data.

### 4.1 Dropping Unnecessary Features and Filling Missing Values

At first, we should notice that missing values dominate feature 1. With regular method of filling missing values, we might introduce much noise into the data. So we drop this feature. Moreover, feature 16 is full of ones and if every entry of a feature has the same value, it is not representative. In this case, feature 16 is also dropped.
In addition to these features, other features and return data have many missing values. Before we fill in these values, we should see that the numeric properties of these data are different. For features, several of them only have discrete numbers while others are continuous variables. For the return data, they only have continuous variables and the adjacent entries are similar in most situation. So we fill in these missing values with different methods:
• For features 4,8,9,12 and 18 that are discrete, we fill in with linear interpolation of neighboring and use only discrete numbers.
• For other features, we still use linear interpolation to fill in but with continuous numbers.
• For the daily and intraday data, we fill in missing values with piecewise cubic spline interpolation to impose the continuity.

### 4.2 SVD Decomposition

So far, we have obtained a full rectangle matrix with consistent data type in each column and without any missing values. However, we have not solved the problems of saving and denoising. Actually, the size of original data is over 70MB and it is full of noise. The sources of noise are both from the original data and the filling values from the last step. Luckily, these two problems can be solved coherently by using truncated singular value decomposition (TSVD). Basically, we separate the matrix obtained in last step into three small size matrices, U , Σ and V , where U and V are orthogonal and Σ is diagonal. In this case, the original huge matrix has been transformed into a product of three small matrices. The total size of these three matrices is about 3MB. Meanwhile, we truncate small singular values when we construct these three matrices. These small singular values keep the original problem ill-posed so this step is equivalent to filtering out the noise. For this problem, we only conduct TSVD for the intraday return because these data occupy the major part of all data and the datatypes of corresponding columns are consistent. To implement TSVD, we use a Matlab package called 'PROPACT' (Larsen (1998, 2000)) to find the largest 28 singular values and the corresponding singular vectors. With these matrices, the reconstruced matrix is significant close to the original one. Actually, the relative difference based on F-norm of these two matrices is around 1%.

## 4.3 Scaling

Another problem that we are concerned about is how to scale the data. The intraday return ratios are much smaller than the daily return data and different features have various numeric properties. Moreover, several algorithms such as multi-layer perceptron regressor are not scaling invariant. So we use min-max scalers to scale all columns for the training data and we reuse the previous scalers to predict the test data.

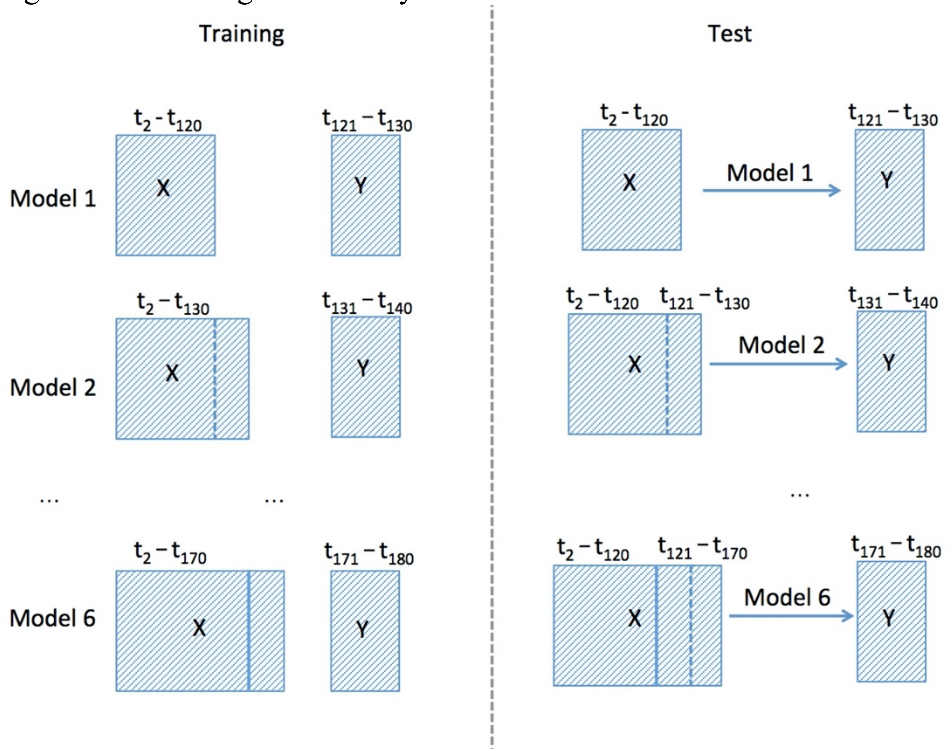
## 5. Prediction

As we can see, this problem can be formulated as a prediction problem. Based on the features, daily returns and parts of intraday returns, we want to predict the rest of intraday returns and future returns. This prediction problem has multiple inputs and multiple out- puts. In addition, these multiple outputs might be dependent on each other. To handle this situation, we propose a two-step method that is based on ensemble of different regressors. At first, we only use intraday returns to predict the unknown intraday returns. In the second step, both the features, daily returns, known intraday returns and predicted intraday returns are used to predict the future daily returns. Why do we prefer to use a two-step method rather than to predict all outputs together? This is because the unknown intraday returns count for around 1/3 of overall data. We have too many unknown variables so that the prediction might not be accurate. Moreover, daily return ratios are usually greater than the intraday return ratios. In addition, since we divide the trading time into 180 blocks, then it might not change drastically from one time block to another time block.

## 5.1 Intraday Return Prediction: Window Method

For this part, we have already used SVD decomposition to filter out noise and to compress the data. Moreover, each column has been scaled between 0 and 1. To predict the unknown intraday return ratios, we propose a ’window’ method that moves the window recursively. This process can be explained in Figure 1.

Figure 1: Predicting the Intraday Return Ratios



Basically, this ’window’ method can be implemented into two steps:

- For the training data, we fit the model recursively. For example, the first model is fitted with columns t2 − t120 as inputs and t121 − t130 as output. the second model is fitted with columns t2 − t130 as inputs and t131 − t140 as output. By keeping iterating it, the last model is fitted with columns t2 − t170 as inputs and t171 − t180 as output. These models are saved to reuse in the second step.
- For the test data, we only know the columns t2 −t120 so we predict the next 60 columns

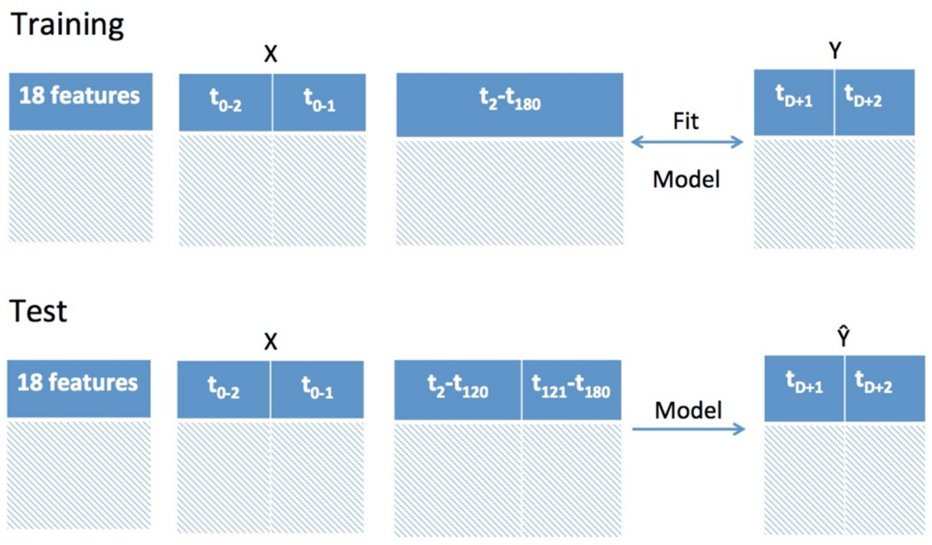
recursively. For example, we predict t121 − t130 for test data with inputs t2 − t120 and the first model. Then we use the known columns t2 − t120 and the previous predicted columns t121 − t130 as the new imput to predict t131 − t140 with second model. By keeping iterating it, we will obtain all intraday return ratios for test data.

As we can see, the size of ’window’ is chosen as 10. Meanwhile, other sizes have also been tested. If we choose the size of window as 1, then it takes too long to finish all processes. Based on the experiment results, it does not improve the accuracy significantly if we shrink the window size. Moreover, if we use larger windows, it takes less time but not accurate and it will influence the results of next step.

## 6. Daily Return Prediction

For test data, we have obtained all intraday return ratios so far. The next goal is to predict the daily return ratios tD+1 and tD+2. To predict daily return ratios, we will include all intraday return ratios, previous daily return ratios and parts of features as inputs. The underlying logistics is that we assume that daily return ratios are closely connected with these variables. Before we fit the model, we still need several preparations.
After dropping two features, 23 features are kept for the next step. Among these 23 features, several of them might not be typical so we can drop these features to both simplify the model and avoid introduction of noise. We use principal component analysis (PCA) to select the features that can explain 95% of overall features. After running PCA, 18 features are kept for prediction. To keep these features consistent with the return ratios, we also scale these features such that they are between 0 and 1.
Then we fit the model and predict the daily return ratios tD+1 and tD+2. The process can be explained in Figure 3.3. Again, we fit the model with training data that include 18 scaled features, previous daily return ratios tD−2 and tD−1 and intraday return ratios t2 − t180 as inputs and daily return ratios, tD+1 and tD+2, as outputs.

Figure 2: Predicting the Daily Return Ratios



and predict tD+1 and tD+2 for test data.

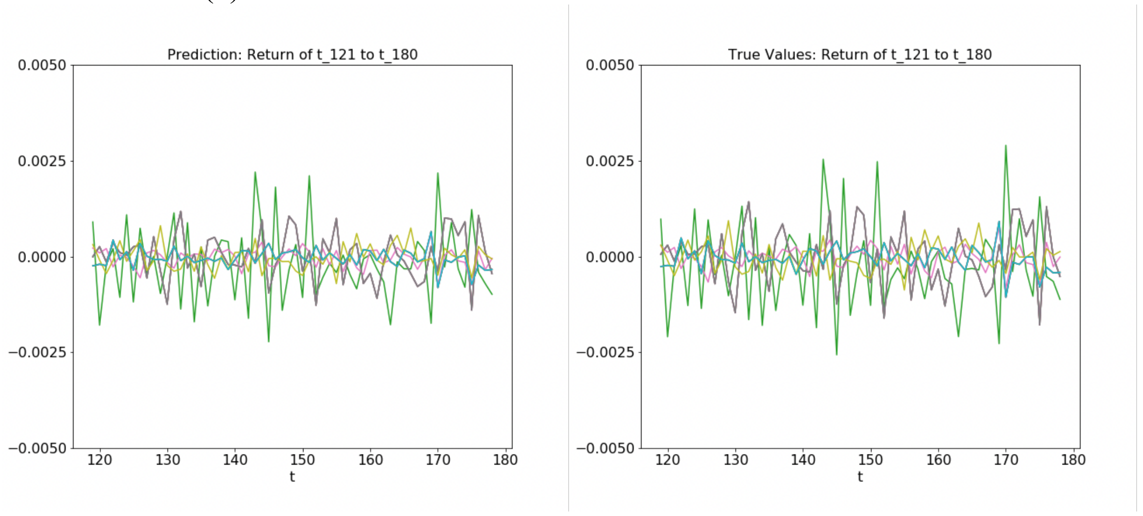
## 7. Ensemble and Numerical Experiments

When considering what estimator to use for the prediction, the ensemble method becomes the best choice because it provides the optimal performance at a cost we can bear in our scenario. For each of the six models mentioned above, we use multiple linear regression estimator, decision tree regressor and SGD regressor and then we ensemble the three es- timators by averaging the three outputs with different weights. The reason for the choice of estimators is that the multiple linear regression is prone to over-fitting, so we add in stochasticity by the SGD regressor and generalizability by decision tree regressor. We can assign random weights to the estimators because the target variables are continuous. In the process of learning, the weights are adjusted according to performance of the prediction. Figure 3 shows the comparison of the predicted values and true values for the later part of the intraday returns for five stocks, which are chosen randomly as examples. Figure 4 shows the comparison of the predicted values and true values for the +D1 and +D2 returns for all the stocks. The pattern in the figures indicates that the prediction follows the trend of the true values. Then we further evaluated the model we established by the relative error and the weighted mean absolute error (WMAE). The relative error is the Frobenius norm defined as equation (1):

$$\text{Relative Error} = \frac{\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\|_F}{\|\boldsymbol{Y}\|_F}$$

After the optimization of the window method parameters and ensemble method parame- ters, for the training dataset, we achieve an optimal prediction with a relative error of 13.0% for the intraday returns (t121 − t180) and 27.6% for the +D1 and +D2 returns. The weight given to the three estimators are 9/10 for multiple linear regression, 1/20 for decision tree regressor and 1/20 for SGD regressor. The window size is expanding 10 one-min intervals

Figure 3: Plot of intraday returns for five stocks chosen randomly. (a) The prediction of returns for t120 − t180 (b) The true values of returns for t120 − t180.

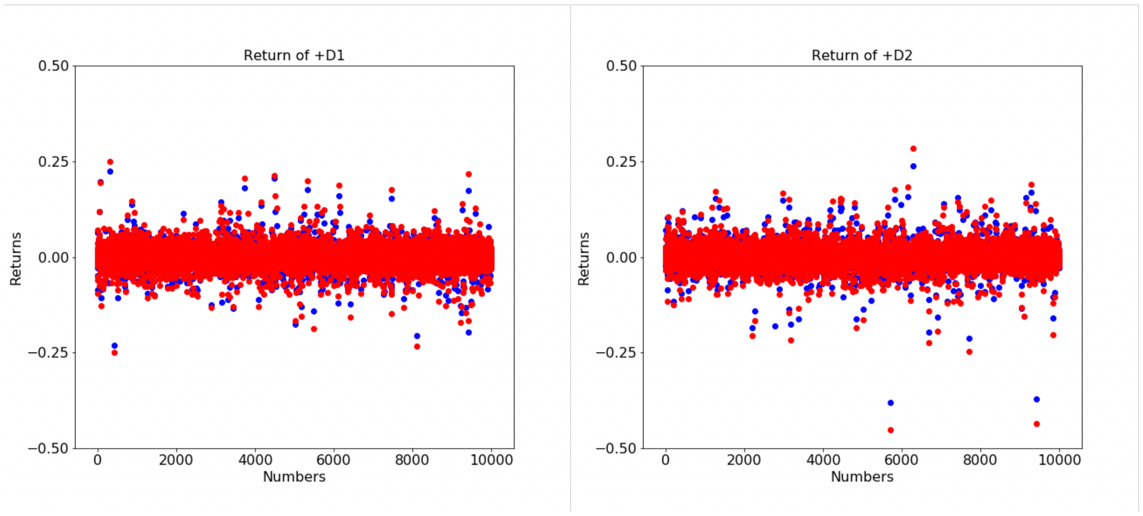


at a time. The WMAE is the mean of the errors weighted by an pre-assigned weight defined in equation (2):

$$\text{WMAE} = \frac{1}{n}\sum_{i=1}^{n} w_i|y_i - \hat{y}_i|$$

The WMAE is 6501.3 WMAE for the prediction of +D1 returns in the training datasets and 7483 for the +D2 prediction. These results are incomparable with the leaderboard results in Kaggle competition because the true values for the test are not revealed so we cannot do the calculation for the test datasets.

Figure 4: Stock returns from (a) +D1 (b) +D2. The blue dots represent the predicted returns and the red dots represent the true values of returns.



8. Comparison with ARMA model

8.1 Data preprocessing and model setup

The datasets after the same pre-processing procedures are used for the ARMA model. The data frame is transposed to have each column indicate a stock and the row index indicates the time sequence to fit in the model. The intraday returns from t2 to t120 are used to train the ARMA model. Multiple combinations of (p,q) parameters are tested to find an optimal set.

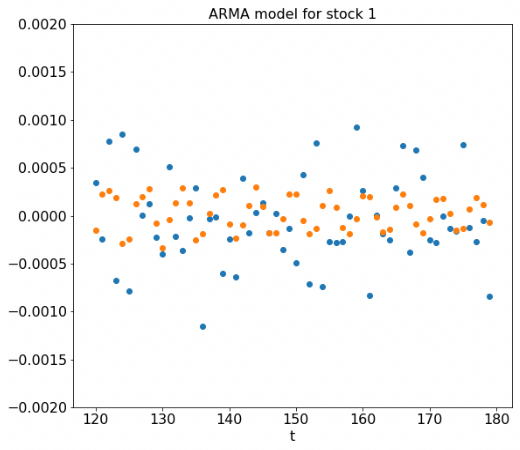
8.2 Results of prediction

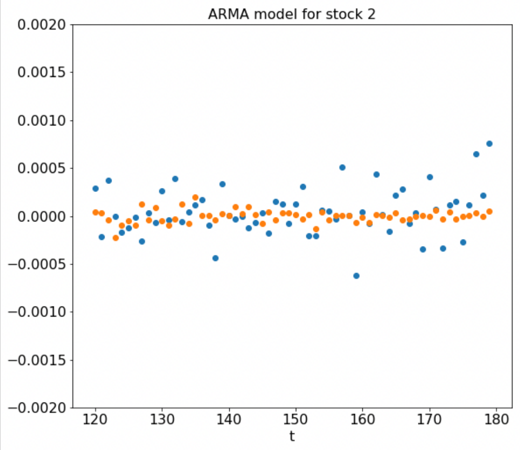
The ARMA model consumes much more time than our method because it cannot process a batch of stocks, which means we need to train a single model for every stock. As the orders of the model goes higher, the prediction shows to have better agreement with the true values, however, the computational cost is too huge for us to bear. For Stock one, we assign (p,q) as (10,5) to obtain a prediction as shown in Figure 5(a), which is still not a

satisfying prediction but takes 12 seconds to train the model. For stock 2, the prediction with parameters (20,4) gives us a prediction shown in Figure 5(b) and it takes 214 seconds to run the training process.

**8.3 Comments**

The ARMA model is more time-consuming and computationally expensive than our window method. The ARMA model might be able to give us precise predictions if the sample sizes (number of stocks) are relatively small, but it requires good expertise to train the model.
Figure 5: Stock returns for t121−t180. The yellow dots represent the predicted returns and the blue dots represent the true values of returns.





## References

The winton stock market challenge. https://www.kaggle.com/c/ the-winton-stock-market-challenge. Accessed: 2017-04-30.
Leona S Aiken, Stephen G West, and Steven C Pitts. Multiple linear regression. Handbook of psychology, 2003.
Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larran˜aga. A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(5):216–233, 2015.
L´eon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer, 2010.
Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learn- ing, volume 1. Springer series in statistics Springer, Berlin, 2001.

References

The winton stock market challenge. https://www.kaggle.com/c/ the-winton-stock-market-challenge. Accessed: 2017-04-30.

Leona S Aiken, Stephen G West, and Steven C Pitts. Multiple linear regression. Handbook of psychology, 2003.

Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larran˜aga. A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(5):216–233, 2015.

L´eon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer, 2010.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learn- ing, volume 1. Springer series in statistics Springer, Berlin, 2001.

Rasmus Munk Larsen. Lanczos bidiagonalization with partial reorthogonalization. DAIMI Report Series, 27(537), 1998.

Rasmus Munk Larsen. Computing the svd for large and sparse matrices. SCCM, Stanford University, June, 16, 2000.

William Wu-Shyong Wei. Time series analysis. Addison-Wesley publ Reading, 1994.

G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. Neurocomputing, 50:159–175, 2003.