Project: App Average Usage Forecasting

Dora Zhu

Abstract:
This project is a full analysis of a dataset “AppData.Rdata”, which contains averages of daily data usage (MB) of 9 categories (namely App i, for i = 1, . . . , 9) of mobile apps of a mobile carrier operating in Chengdu, Sichuan. The data spans December 1, 2019 to April 30, 2020, during which the city government imposed three levels of public health restrictions: Policy-level 1, starting January 25, when households were limited to daily grocery trips; Policy-level 2, starting February 26, when most mobility restrictions were lifted; Policy-level 3, starting March 25, when all restrictions were lifted.
The goal of this project is to forecast the average usage starting from Mar 1, 2020 to April 30, 2020. xs,i is the true average of data usage of App i on Day s and x^s,i is my forecast. The accuracy of forecast is decided by the mean squared errors, which need to be minimized.

Contents

1 Preprocessing

1.1 Overview

The dataset "StockData.Rdata" contains weekly log returns of SP500 index and 10 stocks for 260 weeks. The first $n_1 = 208$ weeks are for training and data for last $n2 = 52$ weeks is the test data.

1.2 Visualization

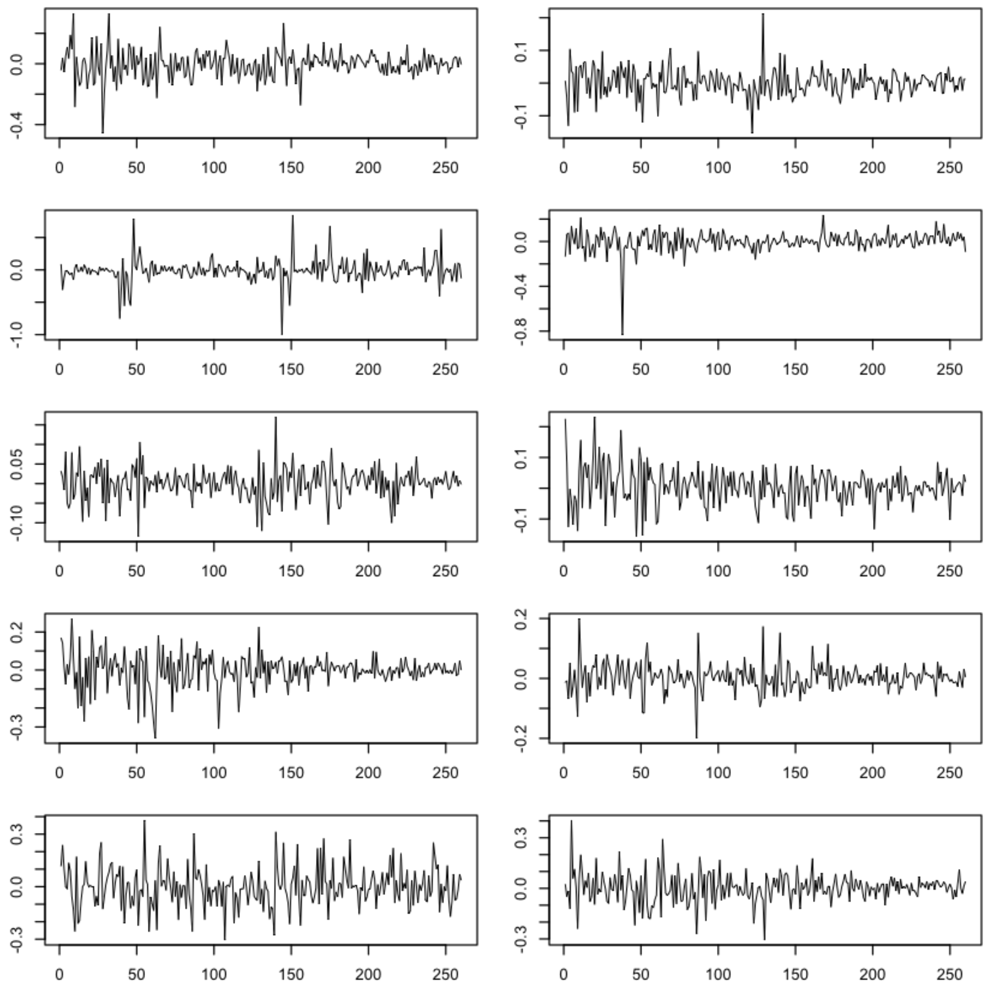

Figure 1: Log return of the stocks

From Figure 1, we can notice the all of stocks seems stationary, they are roughly with stable variance and trend. There is no need to transform the data in this step.

2 Linear Models

2.1 ARIMA Model

First, we use the ARIMA model to fit the data. ARIMA, short for auto-regressive integrated moving average model, can help us to capture some linear structure inside the time series. It is made of the following three parts:

- AR: Auto-regressive. Using the relationship between an observation and some lagged observations. An autoregressive model of order p, abbreviated $AR(p)$, is of the form

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t, \quad (1)$$

where x_t is stationary, $w_t \sim wn\left(0, \sigma_w^2\right)$, and $\phi_1, \phi_2, \ldots, \phi_p$ are constants.

- I: Integrated. Differencing to make the time series stationary.
- MA: Moving average. Using the dependency between an observation and a residual error from a moving average model applied to lagged observations. The moving average model of order q, or $MA(q)$ model, is defined to be

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}, \tag{2}$$

where $w_t \sim wn\left(0, \sigma_w^2\right)$, and $\theta_1, \theta_2, \ldots, \theta_q$ $(\theta_q \neq 0)$ are parameters.

- A process x_t is said to be $ARIMA(p, d, q)$ if

$$\nabla^d x_t = (1-B)^d x_t \tag{3}$$

is $ARMA(p, q)$. Where $ARMA(p, q)$ is defined by

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q} \tag{4}$$

These three parts correspond to the three parameters of this model: p, d, q. To build this model, we must determine these three parameters, i.e. the order of the model. This is the most important part in constructing the models. We can build the ARIMA model in two ways:

1. Find the orders step by step.
2. Just use the existing auto.arima() function.

2.1.1 Model 1: Step-by-Step ARIMA

The general procedure to choose the order is:

- Choose d: The purpose of d is to make the time series stationary. Hence, we need to find the smallest number of time of differentiation, which make the time series stationary. We can use the unit root test to check if the time series is stationary after several times of differentiation. Because the test for x_t needs the assumption that ∇x_t is stationary, we search from $d = 5$, where $\nabla^5 x_t$ is always stationary, and try smaller and smaller d then until $\nabla^d x_t$ is not stationary.
- Choosing p and q: After finding d, we can fit an ARMA model to the stationary time series after differencing. Usually we can determine p and q from the sample autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. However, there are so many models to fit and it is not possible to choose one by one by eyes. Hence, we can choose another approach. We can go through the possible p and q combinations and use the Akaike information criterion (AIC) to assess each model to find the one with the smallest AIC, i.e. the best one.

Methodology Explanation

- **Unit Root Test**

  Assume x_t follows $AR(p)$ and ∇x_t is stationary. To test the hypothesis that the process has a unit root at 1 (i.e., the AR polynomial $\phi(z) = 0$ when $z = 1$, Rewrite it as

$$\nabla x_t = \gamma x_{t-1} + \sum_{j=1}^{p-1} \psi_j \nabla x_{t-j} + w_t \tag{5}$$

  where $\gamma = \sum_{j=1}^p \phi_j - 1$ and $\psi_j = -\sum_{i=j}^p \phi_i$ for $j = 2, \ldots, p$.
  We can test $H_0 : \gamma = 0$ by estimating γ in the regression of ∇x_t on $x_{t-1}, \nabla x_{t-1}, \ldots, \nabla x_{t-p+1}$, and forming a Wald test based on $t_\gamma = \hat{\gamma} / \operatorname{SE}(\hat{\gamma})$.

- **Akaike's Information Criterion (AIC)** AIC is an estimator of prediction error and thereby relative quality of statistical models for a given set of data. It penalize the error variance by a term proportional to the number of parameters. It provides means for model selection.

$$\text{AIC} = \log \hat{\sigma}_k^2 + \frac{n+2k}{n} \tag{6}$$

  where

$$\hat{\sigma}_k^2 = \frac{\text{SSE}(k)}{n}, \quad SSE = \sum_{t=1}^{n} \left(x_t - \hat{\beta}^T z_t \right)^2 \tag{7}$$

  k is the number of parameters in the model.

Model Building Using this method, we build the model and fit it to the testing data to predict, the performance of the model is shown in Figure 2. The detailed MSE of the model is

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0023	0.00062	0.024	0.0037	0.00085	0.0013	0.0010	0.00060	0.0097	0.0014
Step 2	0.0024	0.00057	0.025	0.0034	0.00085	0.0013	0.0010	0.00059	0.010	0.0014
Step 3	0.0022	0.00054	0.026	0.0034	0.00083	0.0012	0.0010	0.00054	0.010	0.0013

The average MSE is 0.0045481823475522 for 1-step forecast, 0.00466822494963931 for 2-step forecast, 0.00473907836636021 for 3-step forecast.

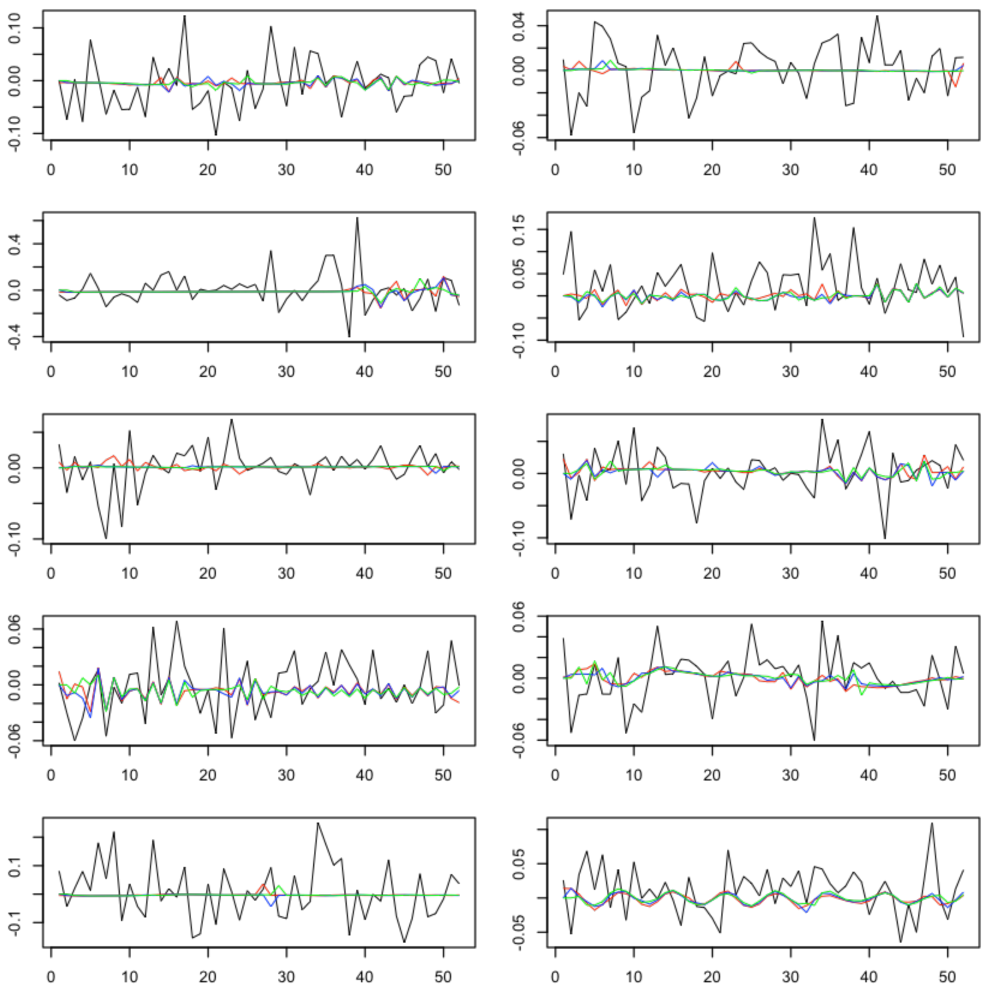

Figure 2: Prediction of Model 1

From the Figure 2, we can observe that the fitted line is somehow flat. In some stocks, it is even close to a straight line of 0. In order to find the reason of this phenomenon, we can take a look at the order of the model in each moving training window.

Because there are too many groups of orders, we just show the orders of ARMA in stock 1. We can notice that in the first half of stock 1, a large section of the order are (0, 0, 0), which means they are just white noise. These ARIMA(0, 0, 0)s cannot capture any useful structure for prediction. In other stocks, there are also many (0, 0, 0)s. These make no sense and shows that ARIMA may not able to capture the features in the time series.

p	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0	0	2	2	2	0	2	0
d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0	0	2	2	2	0	2	0

p	0	0	0	0	0	0	0	0	3	3	3	0	0	3	3	0	2	2	2	2	2	2	2	2
d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	3	0	0	0	3	3	0	0	3	2	2	2	2	2	2	2	2

p	2	2	2	2
d	0	0	0	0
q	2	2	2	2

2.1.2 Model: SARIMA

The multiplicative seasonal autoregressive integrated moving average model, as known as SARIMA, is given by

$$\Phi_P\left(B^s\right)\phi(B)\nabla_s^D\nabla^d x_t = \delta + \Theta_Q\left(B^s\right)\theta(B)w_t \tag{8}$$

where w_t is the usual Gaussian white noise process.

It performs well for time series whose dependence on the past tends to occur most strongly at multiples of some underlying seasonal lag s. To check if it is necessary to use the SARIMA model, we first test the seasonality of the time series. Using the isSeasonal() function in the “seastests” package, we can easily test the seasonality for each moving training window in each stock. The function the WO-test(Ollech and Webel (2019)) is used to assess the seasonality of a time series and returns a boolean. Surprisingly, we can find that none of the time series have seasonality. Hence, it is not necessary to use the SARIMA model.

2.1.3 Model 2: Auto ARIMA

The function auto.arima in the “forecast” package helps to fit the best ARIMA model to univariate time series. It returns the best ARIMA model according to either AIC, AICc or BIC value. The function conducts a search over possible model within the order constraints provided. Also, it can analyze the seasonal factor of the time series which is not used in the data. The general procedure is similar to Model 1. The difference is that we use the existing function and use the function to find the proper (p, d, q) for us.

Using this method, the MSE is:

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0021	0.00060	0.025	0.0035	0.00086	0.0012	0.00095	0.00065	0.0095	0.0013
Step 1	0.0023	0.00059	0.025	0.0035	0.00083	0.0013	0.00099	0.00066	0.0096	0.0013
Step 1	0.0022	0.00053	0.025	0.0031	0.00083	0.0013	0.00094	0.00061	0.0098	0.0013

The average MSE is 0.004562307988185 for 1-step forecast, 0.00455274598011819 for 2-step forecast, 0.00455566433141999 for 3-step forecast.

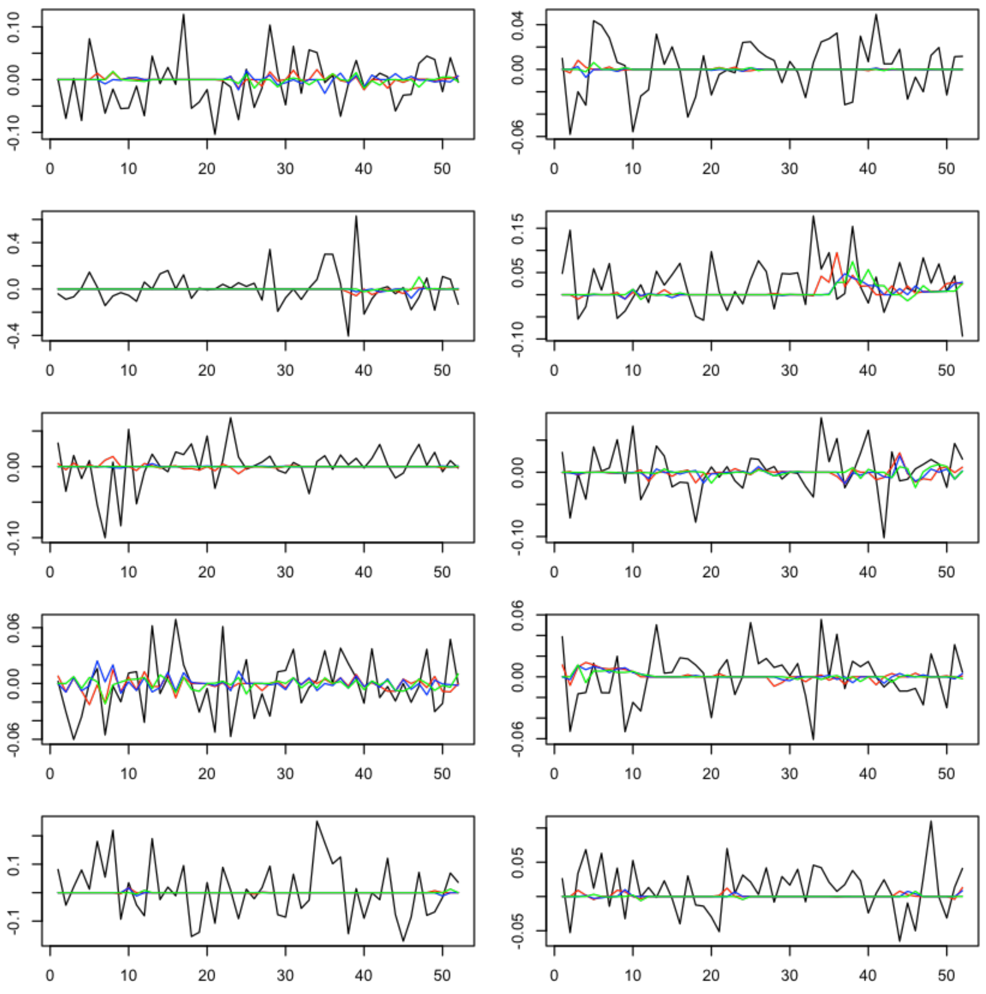
Figure 3: Prediction of Model 2

Compare the MSE and plot of Model 1 and Model 2, we can observe that they are really close. It is reasonable because these two models are two approaches based on the same method. These two models also corroborate each other. ARIMA may not be a good choice, we need to find some more complex model.

2.2 GARCH Model

GARCH model is short for generalized autoregressive conditionally heteroscedastic model and is a model describing the variance. The $GARCH(p, q)$ model is like

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{p} \alpha_j r_{t-j}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{9}$$

Typically, the return for financial series does not have a constant conditional variance. Hence, we can use the ARMA model for the mean with a GARCH model for the errors.

2.2.1 Model 3: ARIMA-GARCH

Using the "rugarch" package in R, we can fit an ARIMA-GARCH model. For the ARIMA part, we can pick up the order from our Model 2. Using the ARIMA orders in ugarchspec() function to fit the mean model and let the computer to choose the best variance model. The MSE of this model is

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0022	0.00060	0.025	0.0034	0.00088	0.0013	0.00090	0.00065	0.0096	0.0012
Step 2	0.0023	0.00060	0.025	0.0036	0.00084	0.0013	0.00094	0.00065	0.0097	0.0013
Step 3	0.0023	0.00054	0.025	0.0031	0.00084	0.0012	0.00098	0.00058	0.0099	0.0012

The average MSE is 0.004532368 for 1-step forecast, 0.004658536 for 2-step forecast, 0.004582621 for 3-step forecast.

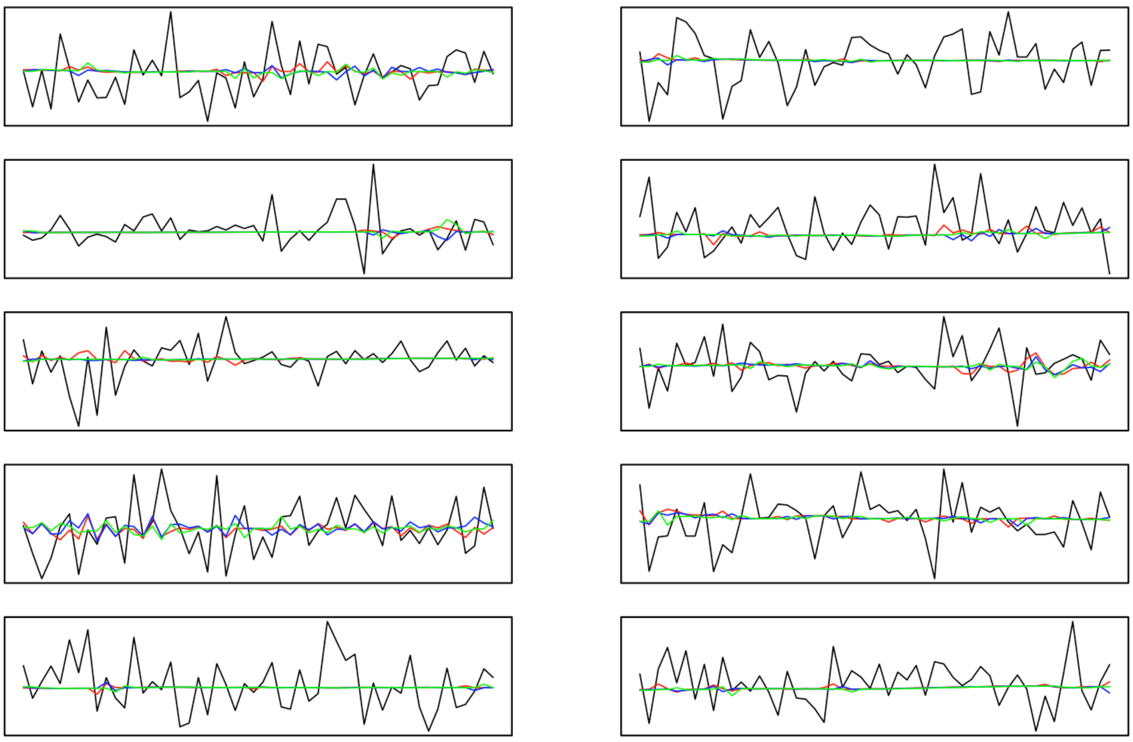

Figure 4: Prediction of Model 3

We can see that the GARCH model does not perform better significantly than ARIMA. Furthermore, the curves are even flatter. For some of the plots, they look just like straight lines. Still, it is not satisfying.

These linear models did not predict the stock's trend well. So we can't help but guess that there may be more structures in this set of time series data that cannot be captured by the linear model. Therefore, we can use some machine learning methods to train the model and make predictions.

3 Nonlinear Models by Machine Learning

Machine learning (ML) is the study of computer algorithms that can be automatically improved through experience. For this problem, in order to predict the future performance of the time series, we can use the supervised learning method to learn a function from the training data set, and predict the result based on this function when new data arrives. Our training set needs to include input and output. Our current row data does not include these, we need to adjust and build our training set as well as the predicting set.

3.1 Building Data Set

The principle of time series is to find the past laws of data to predict its future development trend, i.e. autoregression. Therefore, we need to predict future performance based on past data. Our training_x set should contain past data, and our training_y set should contain future data which we want to predict. We use the first 1 to n_1 weeks to train our model and the last n_1 to $n_1 + n_2$ weeks to test.

To build a model of autoregression based on the past p weeks and to h weeks ahead, i.e. forecast the return of week t based on the past observations in Week $t-h-p$ to Week $t-h-1$, where $h=1,2,3$. For such a (p,h) model, we need to build a dataset of $p+h$ columns. For row i, the data are returns from week i to week $i+p+h-1$. The training data is like:

Training_x					Training_y
x_1	x_2	x_3	...	x_{p+h-1}	x_{p+h}
x_2	x_3	x_4	...	x_{p+h}	x_{p+h+1}
...	...	...	...	...	...
$x_{208-p-h}$	$x_{209-p-h}$	$x_{210-p-h}$	...	x_{206}	x_{207}
$x_{209-p-h}$	$x_{210-p-h}$	$x_{211-p-h}$	...	x_{207}	x_{208}

In the same way, the testing dataset is like:

Testing_x					Testing_y
$x_{210-p-h}$	$x_{211-p-h}$	$x_{212-p-h}$	...	x_{208}	x_{209}
$x_{211-p-h}$	$x_{212-p-h}$	$x_{213-p-h}$	...	x_{209}	x_{210}
...	...	...	...	...	...
$x_{260-p-h}$	$x_{261-p-h}$	$x_{262-p-h}$	...	x_{258}	x_{259}
$x_{261-p-h}$	$x_{262-p-h}$	$x_{263-p-h}$	...	x_{259}	x_{260}

3.2 Model 4: Neutral Network Model

Neural network is a nonlinear statistical data modeling tool. It abstracts the human brain neuron network from the perspective of information processing, establishes a certain simple model, and composes different networks according to different connection methods. In supervised learning, the data of training samples is added to the input of the network, and the corresponding expected output is compared with the output of the network to obtain an error signal to control the adjustment of the weight connection strength. After multiple trainings, it converges to one determined weight.

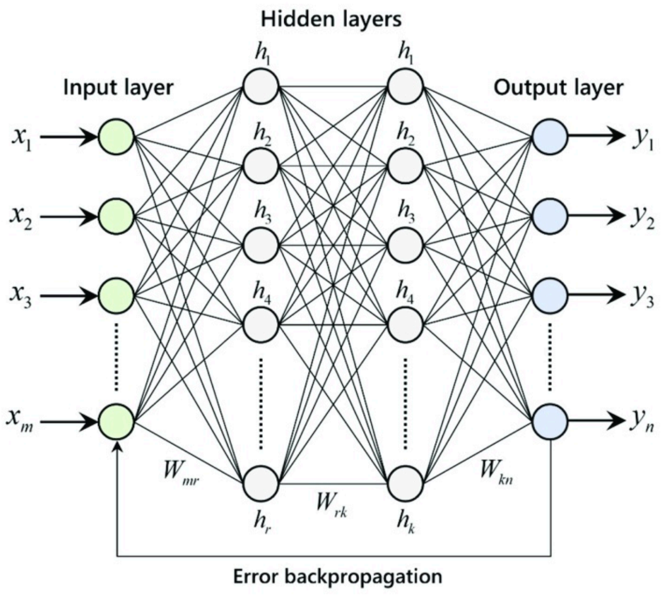

Figure 5: Neutral Network

It is based on a collection of nodes or neurons. Each connection is like a synapse in a biological brain, which can transmit signals to other neurons. The node that receives the signal then processes it and can signal the node connected to the node. These connections are called edges. The weights of neurons and edges are usually adjusted as learning progresses. The weight increases or decreases the strength of the signal at the connection. Normally, neurons gather into layers. The layer performs a transformation on its input. The signal propagates from the first layer to the last layer, and may propagate after traversing the layers multiple times.

We can use the R package "neuralnet" to fit the Neutral Network model. Train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). We will using the function neuralnet to fit neural network with 3 neurons in each layer, learning rate 0.01 used by traditional backpropagation.

We have no idea with the number of past training samples (p) we based on. Therefore, we can try to use different p to see the performance of the models.

3.2.1 Model 4.1: Neutral Network with p = 10

First, we choose $p = 10$, which means predict based on the past 10 weeks. The inspection of the model is shown as Figure 5. The results of the prediction are visualized in Figure 6.

The MSE of fit is:

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0021	0.00063	0.043	0.0057	0.00082	0.0013	0.0011	0.00070	0.013	0.0014
Step 2	0.0031	0.00058	0.026	0.0071	0.00083	0.0013	0.0017	0.00070	0.012	0.0014
Step 3	0.0041	0.00053	0.041	0.0039	0.00086	0.0012	0.0012	0.00066	0.015	0.0013

The average MSE is 0.00693626845526679 for 1-step forecast, 0.0054702149420352 for 2-step forecast, 0.00693933011658757 for 3-step forecast.

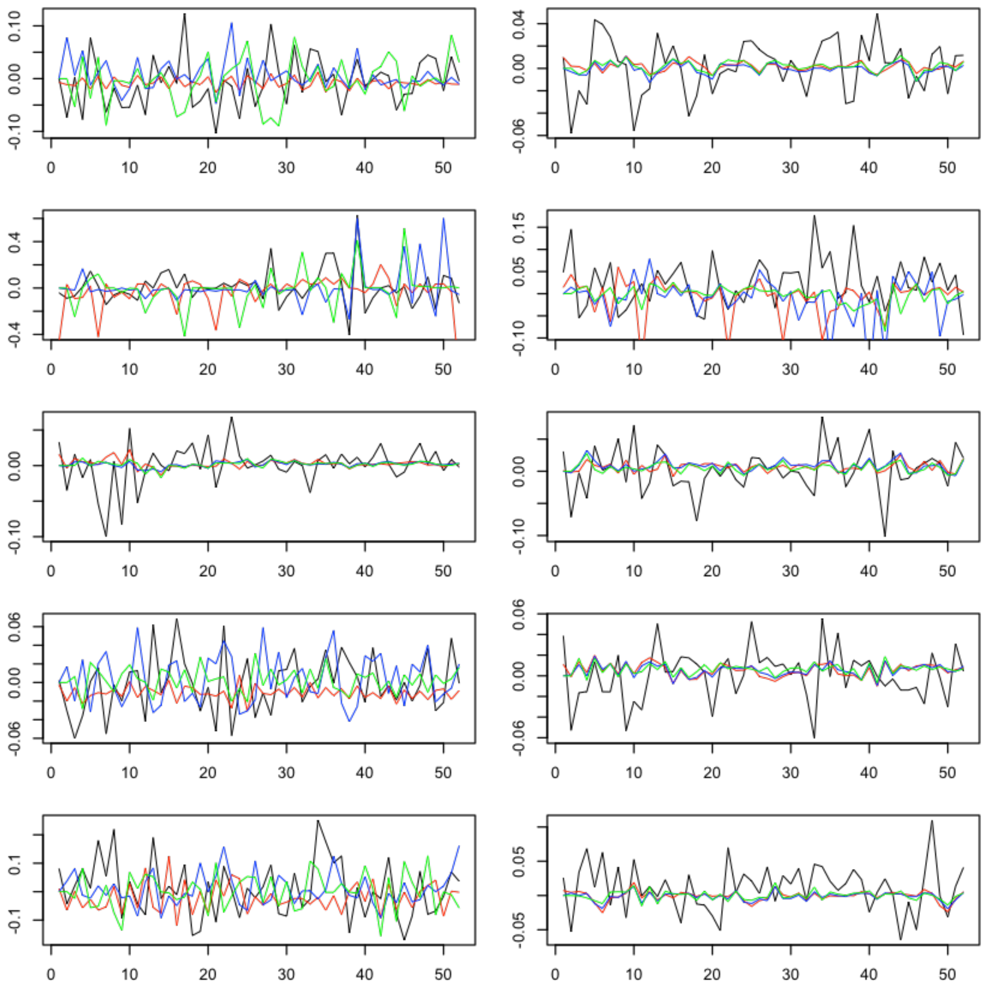

Figure 6: Prediction of Model 4.1

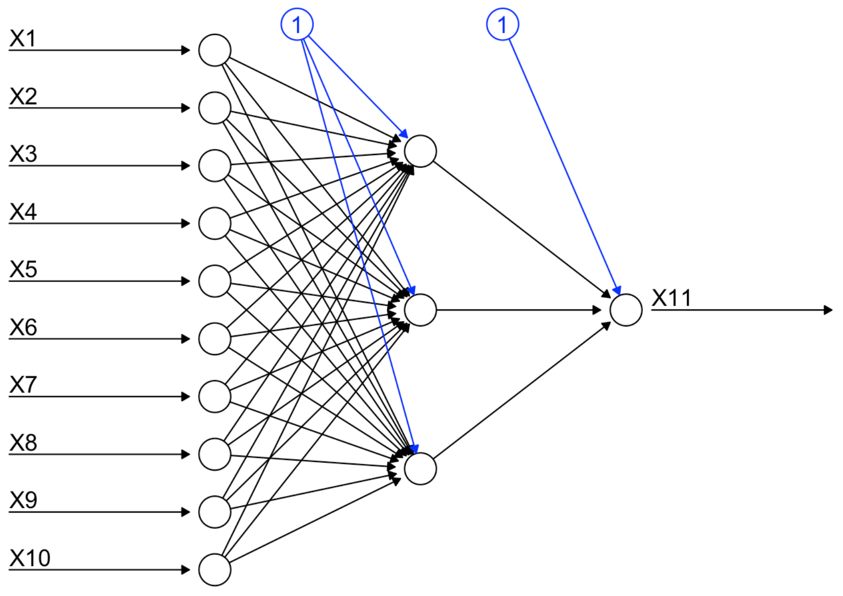

Figure 7: Inspection of Model 4.1

We can see that the MSE is quite large compared to the linear models. It fits quite bad. In some of the stocks it is far from the true value, for example, stock 4. In some other stocks, the fit is really flat and straight, which shows that it contains little information. We can try some other *p*.

3.2.2 Model 4.2: Neutral Network with p = 5

The inspection of the model is shown as following:

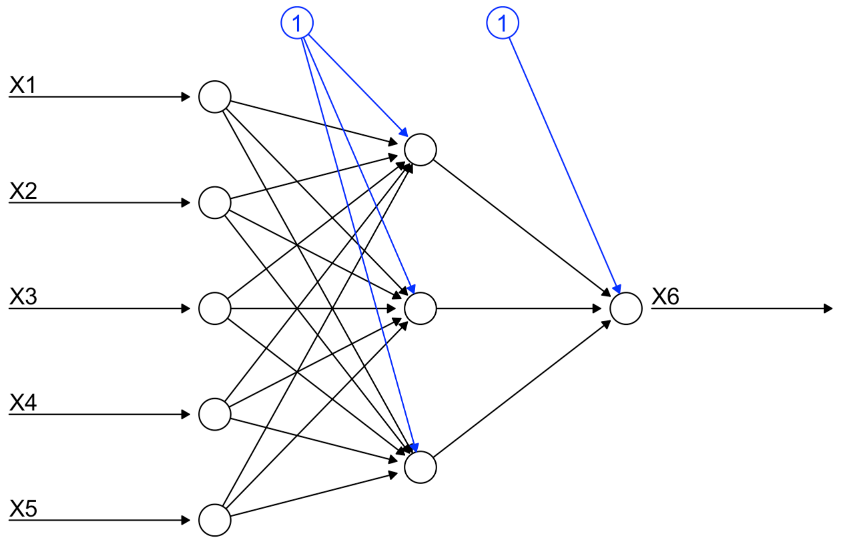

Figure 8: Inspection of Model 4.2

The MSE of fit is:

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0035	0.00057	0.033	0.0043	0.00080	0.0015	0.00100	0.00069	0.0120	0.0012
Step 2	0.0026	0.00061	0.035	0.0046	0.00084	0.0013	0.00120	0.00070	0.0092	0.0013
Step 3	0.0025	0.00057	0.028	0.0037	0.00086	0.0012	0.00099	0.00064	0.0096	0.0014

The average MSE is 0.0058558735600036 for 1-step forecast, 0.00573842700110681 for 2-step forecast, 0.00494526760643023 for 3-step forecast. The 3-step forecast is significantly better than the other two forecast. Different random seeds will have different results.

This fit is also not so good, the MSE is also larger than the linear models. But from the MSE and plot(see next page, Figure 8), it is much better than the Model 4.1. Set different random seed, the overall MSE is around the MSE in above table and smaller than that of Model 4.1. It also provide more information than the linear models. We may guess if a bit smaller p is better. Hence, we try $p = 3$.

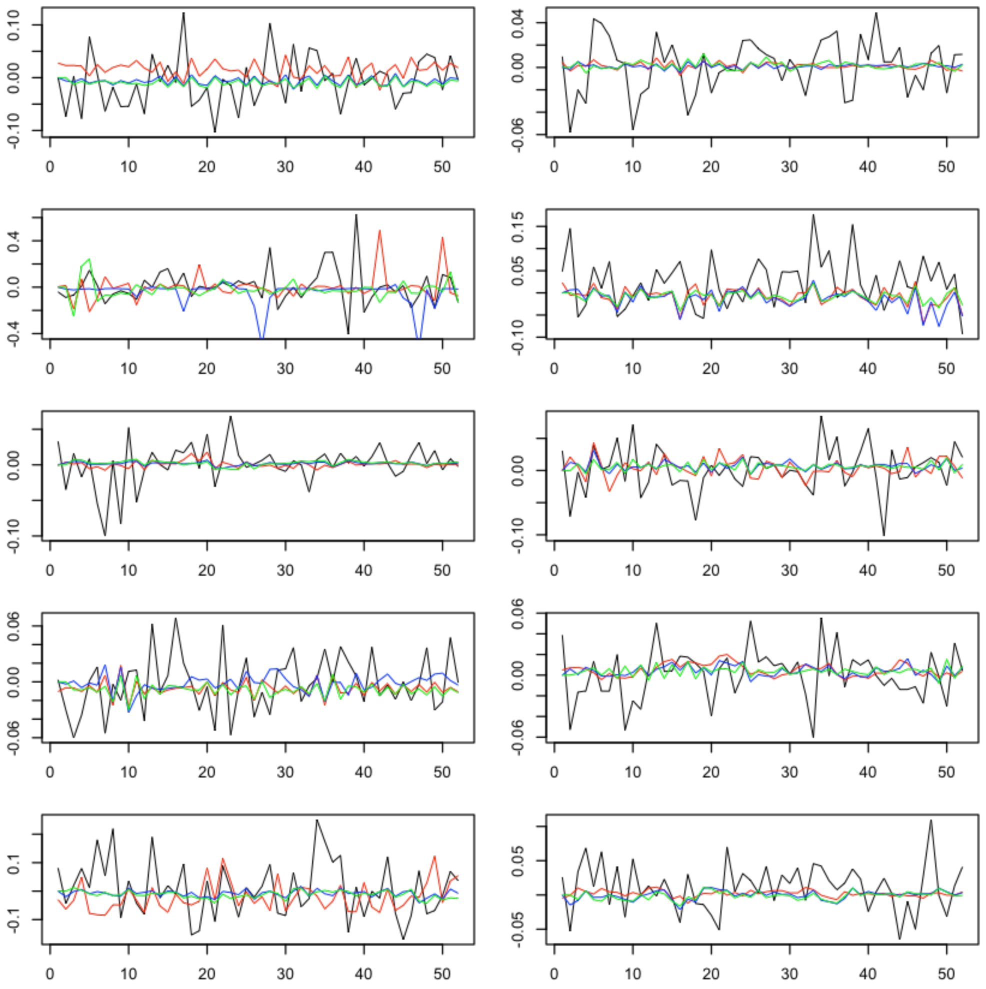

Figure 9: Prediction of Model 4.2

3.2.3 Model 4.3: Neutral Network with p = 3

The inspection of the model is shown in Figure 10.

The MSE of fit is:

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0024	0.00056	0.035	0.0035	0.00079	0.0013	0.0010	0.00069	0.0092	0.0013
Step 2	0.0026	0.00061	0.046	0.0040	0.00081	0.0013	0.0011	0.00066	0.0094	0.0013
Step 3	0.0025	0.00055	0.025	0.0042	0.00081	0.0012	0.0010	0.00058	0.0110	0.0013

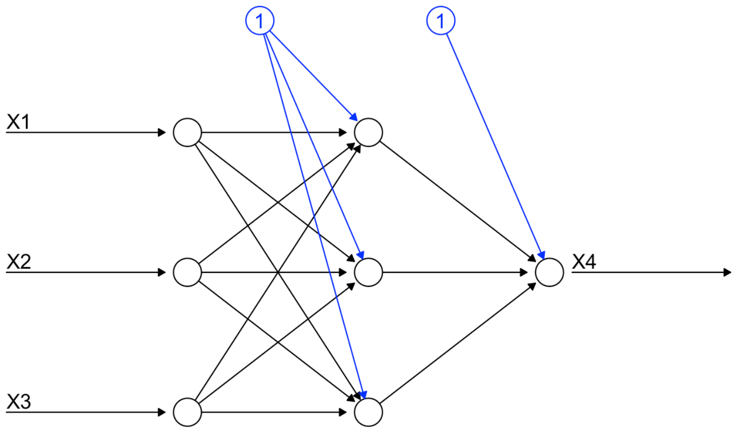

Figure 10: Inspection of Model 4.3

The average MSE is 0.00444418979547168 for 1-step forecast, 0.00469304625667426 for 2-step forecast, 0.00482399634817727 for 3-step forecast. The 1-step forecast is better than the other two forecast for this try. Different random seeds leads to different output. After trying several times, I choose a try with a relatively small MSE.

We can see that the MSE is better than Model 4.1 and 4.2. However, it is more stable without large fluctuations. Model 4.3 performance is closer to the linear model, both from the image and MSE point of view.

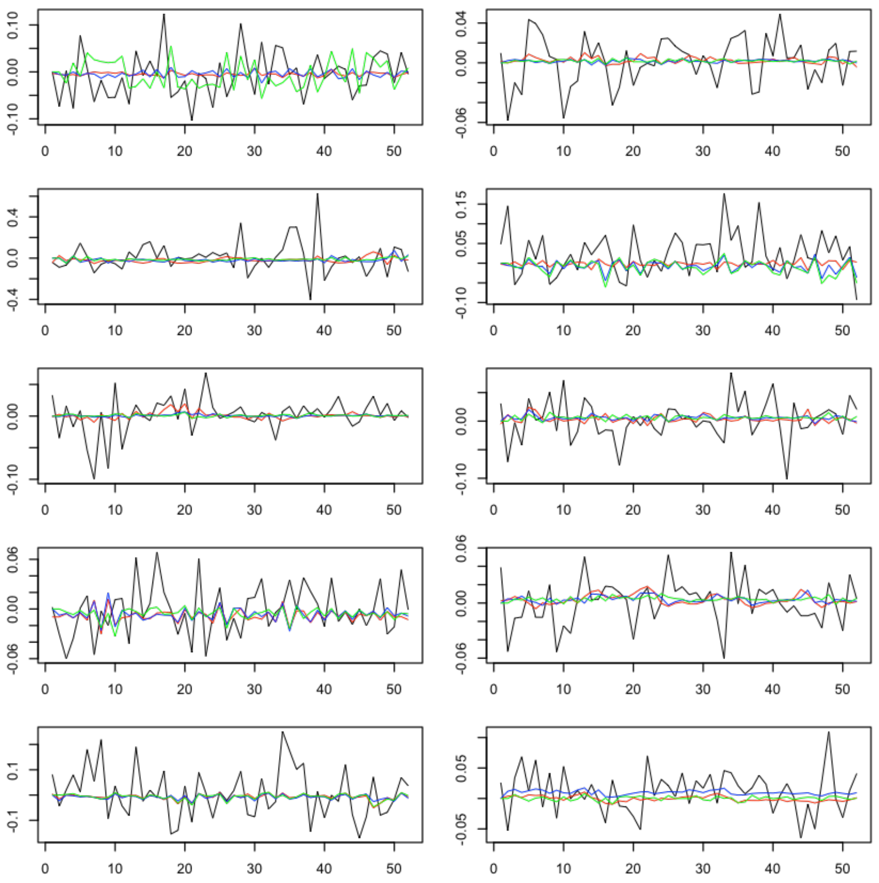

Figure 11: Prediction of Model 4.2

3.2.4 Summary of Neutral Network

From Model 4.1, Model 4.2 and Model 4.3, we can see that to a certain degree with smaller p, the predict will have less volatility and more similar to the Model we mentioned in the Linear part. However, the MSE will be better with smaller p. Hence, we still tends to choose small p for Machine Learning models.

3.3 Model 5: Random Forest

Random forest is an algorithm that integrates multiple trees through the idea of ensemble learning. Its basic unit is a decision tree, and its essence belongs to a major branch of machine learning, an ensemble learning method.

The essence of the random forest algorithm is a classifier ensemble algorithm based on decision trees, in which each tree depends on a random vector, and all vectors in the random forest are independent and identically distributed. Random forest is to randomize the column variables and row observations of the data set, generate multiple classification numbers, and finally summarize the results of the classification tree.

Compared with neural networks (Model 4), random forest reduces the amount of calculations and improves the prediction accuracy. The algorithm is not sensitive to multivariate collinearity and is more robust to missing and unbalanced data, and can adapt well to up to several thousand. Explain the variable data set.

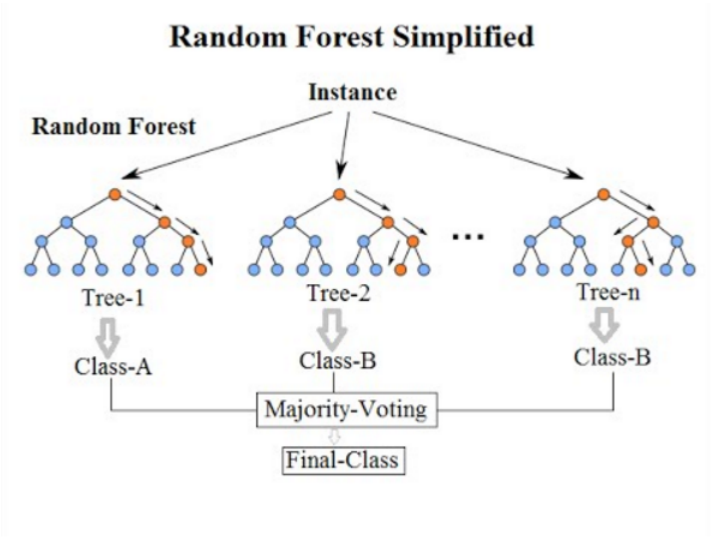

Figure 12: Random Forest

The estimation process of random forest

1. Specify the value of m, i.e. randomly generate m variables for the binary tree on the node, and the choice of the binary tree variables still meets the minimum impurity principle of the node;

2. The Bootstrap method is used to randomly select k sample sets with replacement in the original data set to form k decision trees, and the unsampled samples are used for the prediction of a single decision tree;

3. According to the random forest composed of k decision trees to classify or predict the sample to be classified, the principle of classification is the voting method, and the principle of prediction is simple average.

Building the model We can use the "randomForest" package in R to do the modelling. It implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. Learning from Model 4, this time we only choose $p = 3$ to train the model.

Fitting the model, we can get the following result. The MSE of this model is

	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	Stock10
Step 1	0.0037	0.00071	0.026	0.0040	0.00120	0.0019	0.0014	0.00087	0.012	0.0015
Step 2	0.0041	0.00072	0.028	0.0042	0.00110	0.0014	0.0016	0.00080	0.013	0.0019
Step 3	0.0033	0.00063	0.030	0.0040	0.00081	0.0013	0.0016	0.00077	0.013	0.0018

The average MSE is 0.00531817675574107 for 1-step forecast, 0.00563276208050509 for 2-step forecast, 0.00571730255432806 for 3-step forecast.

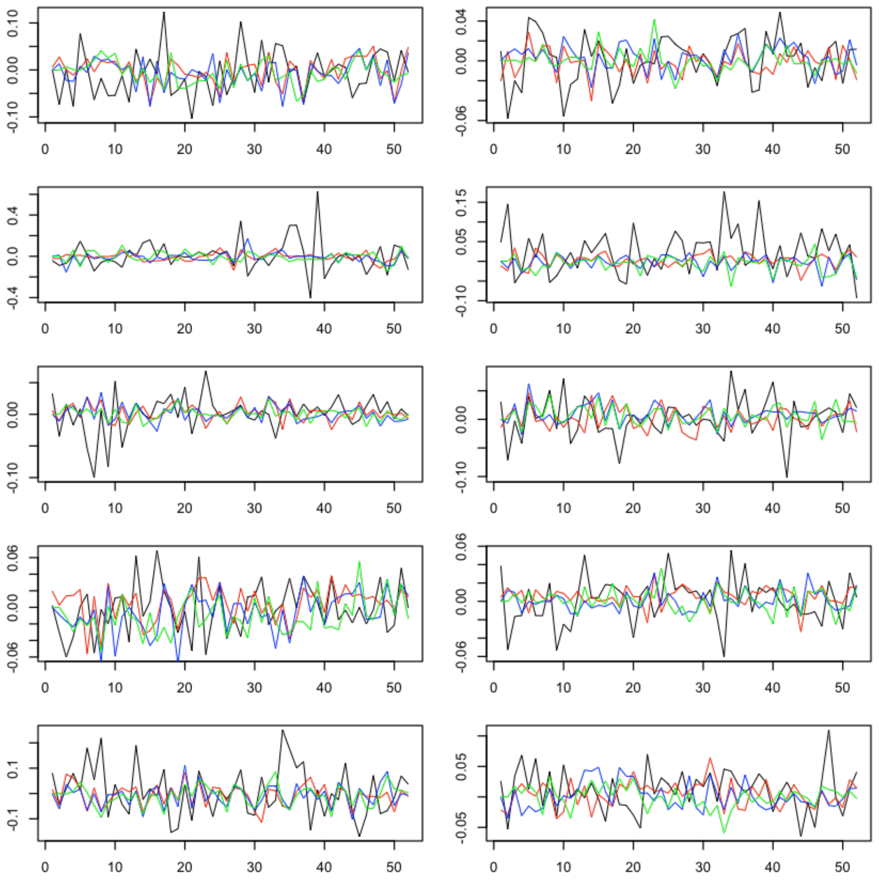

Figure 13: Prediction of Model 5

Although From the figure, the model looks good. However, the MSE is disappointing. The MSE is worse than the linear models and the NN model with the same p. This model also fell short of our expectations.

4 Summary for Forecasting

4.1 Model Choice

We can collect all the average MSE of the models and compare them.

Model	1-Step Forecast	2-Step Forecast	3-Step Forecast
Step-by-Step ARIMA	4.55×10^3	4.67×10^3	4.74×10^3
Auto ARIMA	4.56×10^3	4.55×10^3	4.56×10^3
ARIMA-GARCH	4.53×10^3	4.65×10^3	4.58×10^3
Neutral Network (p=10)	6.94×10^3	5.47×10^3	6.94×10^3
Neutral Network (p=5)	5.86×10^3	5.74×10^3	4.95×10^3
Neutral Network (p=3)	4.44×10^3	4.69×10^3	4.82×10^3
Random Forest	5.32×10^3	5.63×10^3	5.72×10^3

From the table we can see that, from the perspective of MSE. Linear models, ARIMA and ARIMA-GARCH, performs good. Also, Neutral Network with $p = 3$ also does a good job as the linear models. The other models are not satisfying.

However, as we stated before, as the plots show, the models with small MSE often lacks volatility. Sometimes the fitted curves even look like a straight line. In fact, all of the models are not satisfying. Through exploration to the step-to-step ARIMA model, we can find that sometimes the models do not use any features. I speculate that the reason for this result is that this set of data is too close to white noise and it is difficult to capture useful features.

Anyway, I would like to choose the Model 1 step-by-step ARIMA (1 step forecast) as my final model because of its small MSE and the relatively fluctuating performance.

5 Portfolio Construction

In this question, we are asked to construct a portfolio to maximize the Sharpe ratio. We can use several different strategy to construct the portfolio.

5.1 Strategy 1

There is an useful function in the "tseries" package called portfolio.optim(). It can help us to computes an efficient portfolio from the given return series x in the mean-variance sense, which is exactly matching our instruction. Hence, it is a good idea to use this function. The step of this method is like

1. Starting from week n_1, we apply the portfolio.optim() to the observed stock data of week $t - i$ to week t, to get a portfolio for week $t + 1$.
2. Repeat step 1 until $t = n_1 + n_2 - 1$ and get the portfolio for the last week.
3. Try different i to get the best Sharpe ratio

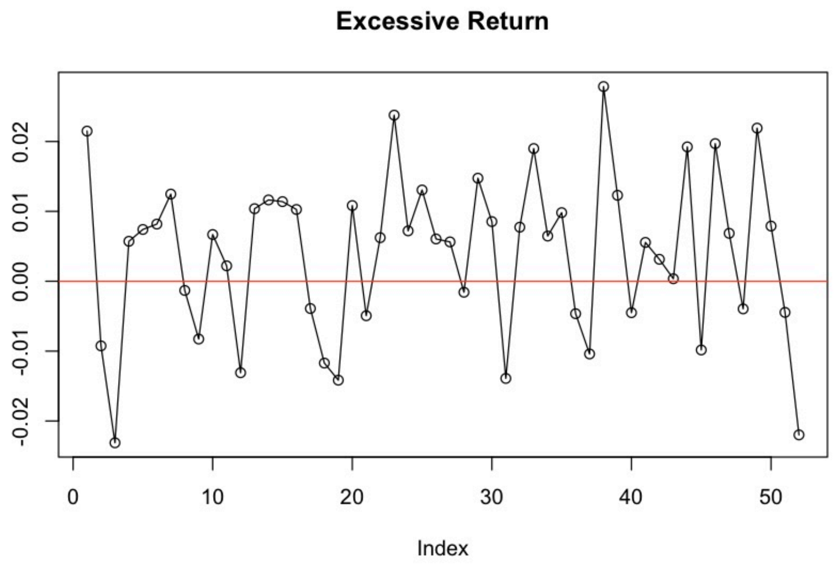

Figure 14: Excessive Return of Strategy 1

It is quite convenient. Then we can check the Sharpe ratio of it and we can get a result of 0.342895.

5.2 Strategy 2

Using our forecasting results, we can simply assign weight to the portfolio by it predicted value. Consider a simple tactics, we can reject to buy any stock that the predicted value is negative in this week. For the positive returns, we can just assign weight by their number.

The step of this method is like

1. Dealing with the forecasting set (for example, the forecasting set of Auto ARIMA). Turning the negative returns to zero.
2. The weight of each stock is equal to its value divided by the sum of row.
3. If all the forecasting returns in this week are negative, we simply assign $w_i = 0.1$ for $i = 1, \cdots, 10$.

The Sharpe ratio is $\hat{\mu}_e/\hat{\sigma}_e = 0.15277$. It is not good.

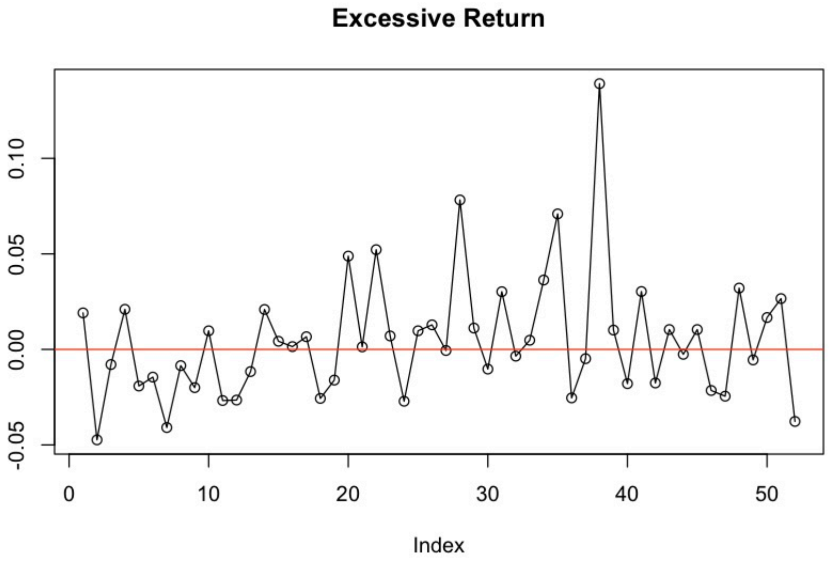

Figure 15: Excessive Return of Strategy 2

5.3 Strategy 3

We can use our forecast to maximize the Sharpe ratio every week. That is, for each week, we try to fit a $\mathbf{w}$ to maximize the accumulated ratio, week by week.

The step of this method is like

1. Starting with a proper w_{109}. For example, we can use $w_{109,i} = 1/10$.
2. Go to the next week (assume week t). Here is a optimization problem. The constraint is $\sum_{i=1}^{10} w_{t,i} = 1$ and the object is to maximize
$$\hat{\mu}_e^t/\hat{\sigma}_e^t \tag{10}$$
where $\hat{\mu}_e^t$ is the sample mean of $\{e_{n_1+1}, \ldots, e_t\}$ and $\hat{\sigma}_e^i$ is the standard deviation.
3. Keep doing until t goes to $n_1 + n_1$.

However, because the nonlinear optimization packages in R are few and hard to use. We cannot deal with this strategy using R. Hence, it has not been used.