MCAST

Data Driven Automated Algorithmic Trading

by

Gabriel Gauci Maistre

A thesis submitted in partial fulfillment for the
degree of Bachelor of Science

in the

Information and Communications Technology
Malta College of Art, Science, and Technology

May 2017

Declaration of Authorship

I, Gabriel Gauci Maistre, declare that this thesis titled, 'Data Driven Investing: Advanced Risk and Portfolio Management in Quantitative Finance' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"If past history was all there was to the game, the richest people would be librarians."

Warren Buffett

MCAST

Abstract

Information and Communications Technology
Malta College of Art, Science, and Technology

Bachelor of Science

by Gabriel Gauci Maistre

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

Alan Gatt l-ahjar lecturer - Nicholas Gollcher

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables ix

Abbreviations x

Physical Constants xi

Symbols xii

1 Introduction 1
- 1.1 Efficient Market Hypothesis - EMH . . . 2
  - 1.1.1 Breaking down EMH . . . 2
  - 1.1.2 What EMH means for investors . . . 2
- 1.2 Zero-Sum Game . . . 3
  - 1.2.1 Zero-Sum Game & Economics . . . 3

2 Background Theory 5
- 2.1 Time Series Analysis . . . 5
  - 2.1.1 ARMA . . . 6
  - 2.1.2 ARIMA . . . 7
- 2.2 Statistical Machine Learning . . . 8
  - 2.2.1 Classification . . . 8
  - 2.2.2 Support Vector Machines . . . 9
  - 2.2.3 Regression . . . 9
  - 2.2.4 Decision Trees . . . 10
- 2.3 Bayesian Statistics . . . 10
  - 2.3.1 Markov Chain Monte Carlo . . . 11

3 Experminetal Setup **13**
3.1 Data Tidying . . . 13
3.2 Stock Selection . . . 13
3.3 Time Series Analysis . . . 13
3.3.1 Random Walk . . . 14
3.3.2 Ordinary Least Squares (OLS) . . . 14
3.3.3 Auto Regressive (AR) . . . 14
3.3.4 Moving Average (MA) . . . 14
3.3.5 Auto Regressive Moving Average (ARMA) . . . 15
3.3.6 Auto Regressive Integrated Moving Average (ARIMA) . . . 15
3.4 Machine Learning . . . 15
3.4.1 Classification . . . 15
3.4.1.1 Decision Tree . . . 15
3.4.1.2 Boosted Decision Tree . . . 16
3.4.1.3 Support Vector Machine (SVM) . . . 16
3.4.1.4 Random Forest . . . 16
3.4.1.5 K-Nearest Neighbour . . . 17
3.4.1.6 Logistic Regression . . . 17
3.4.1.7 Bernoulli Naive Bayes . . . 18
3.4.1.8 Gaussian Naive Bayes . . . 18
3.4.1.9 Neural Network . . . 18
3.4.1.10 Stochastic Gradient Descent . . . 19
3.4.2 Regression . . . 19
3.4.2.1 Decision Tree . . . 19
3.4.2.2 Boosted Decision Tree . . . 20
3.4.2.3 Random Forest . . . 20
3.4.2.4 Linear Regression . . . 20
3.4.2.5 Neural Network . . . 21
3.4.2.6 Stochastic Gradient Descent . . . 21
3.5 Bayesian Statistics . . . 21
3.5.1 No-U-Turn Sampler (NUTS) . . . 21
3.5.2 Metropolis-Hastings . . . 21
3.6 Strategy . . . 22
3.6.1 Classification . . . 22
3.6.2 Regression . . . 22

4 Research Findings **24**
4.1 Time Series Analysis . . . 26
4.1.1 Random Walk . . . 26
4.1.2 Ordinary Least Squares (OLS) . . . 31
4.1.3 Auto Regressive (AR) . . . 37
4.1.4 Moving Average (MA) . . . 47
4.1.5 Auto Regressive Moving Average (ARMA) . . . 57
4.1.6 Auto Regressive Integrated Moving Average (ARIMA) . . . 67
4.2 Machine Learning . . . 77
4.2.1 Classification . . . 77
4.2.1.1 Decision Tree . . . 77

4.2.1.2 Boosted Decision Tree . . . 77
4.2.1.3 Support Vector Machine (SVM) . . . 77
4.2.1.4 Random Forest . . . 78
4.2.1.5 K-Nearest Neighbour . . . 78
4.2.1.6 Gaussian Naive Bayes . . . 78
4.2.1.7 Bernoulli Naive Bayes . . . 78
4.2.1.8 Neural Network . . . 79
4.2.1.9 Logistic Regression . . . 79
4.2.1.10 Stochastic Gradient Descent . . . 79
4.2.2 Regression . . . 79
4.2.2.1 Decision Tree . . . 79
4.2.2.2 Boosted Decision Tree . . . 85
4.2.2.3 K-Nearest Neighbour . . . 90
4.2.2.4 Random Forest . . . 96
4.2.2.5 Linear Regression . . . 101
4.2.2.6 Neural Network . . . 107
4.2.2.7 Stochastic Gradient Descent . . . 112
4.3 Bayesian Statistics . . . 118
4.3.1 No-U-Turn Sampler (NUTS) . . . 118
4.3.2 Metropolis-Hastings . . . 123

5 Discussion and Suggestions for Future Research 131

6 Conclusion 132

A An Appendix 133

Bibliography 135

List of Figures

List of Tables

Abbreviations

EMH	**E**fficient **M**arket **H**ypothesis
RWH	**R**andom **W**alk **H**ypothesis
OLS	**O**rdinary **L**east **S**quares
AR	**A**uto **R**egressive
MA	**M**oving **A**verage
ARMA	**A**uto **R**egressive **M**oving **A**verage
ARIMA	**A**uto **R**egressive **I**ntegrated **M**oving **A**verage
ADF	**A**ugmented **D**ickey **F**uller
SVM	**S**upport **V**ector **M**achine
SGD	**S**tochastic **G**radient **D**escent
ADF	**A**ugmented **D**ickey **F**uller
SMA	**S**imple **M**oving **A**verage
AIC	**A**lkaline **I**nformation **C**riterion
IC	**I**nformation **C**riterion
NUTS	**N**o- **U**- **T**urn **S**ampler

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

To my parents, Keith & Christine Gauci Maistre. Without them, and their unconditional love and support, none of this would have been possible.

Chapter 1

Introduction

"Investing in stocks is just like gambling" - is a phrase commonly heard coming from people describing stock investing. But is it really just like gambling? Were all those investors who made billions from the stock market just lucky? To understand this, we must first review what it means to buy stocks. In the stock market, buying stocks means owning a share of the company. It entitles the stock holder to a claim on assets as well as a fraction of the profits which the company generates. It is unfortunately very common for investors to misunderstand this concept, often thinking of shares as simply a trading vehicle and forget that stock represents the ownership of a company.

So how are stocks valued? The value of a companys stock depends on a number of factors, and assessing the value is not an easy practice. Investors are constantly trying to assess the profit that will be left over to shareholders. This is why stock prices fluctuate. The outlook for business conditions is always changing, and so are the future earnings of a company. Since many people fail to understand this concept, it is far too common to believe that the short-term price movements of a company are random. However, it is the long-term price movements of a company which reflect the value of a company as it is supposed to be worth the present value of the profits it will make. A company can survive without profits in the short term, as long as expectations of future earnings exist. A company may try to fool investors in the beginning, however a companys stock price will eventually be expected to show the true value of the firm.

So how is this all different to gambling? Gambling is a zero-sum game. It merely takes money from a loser and gives it to a winner. No value is ever created. By investing, we increase the overall wealth of an economy. As companies compete, they increase productivity and develop products that can make our lives better. This does not mean that stock investing cannot be a gamble, as it is extremely common for many people to

skip the due diligence before spending a huge chunk of their life savings on stock, often losing it all in the process.

This thesis aims to disprove two hypotheses, the Efficient Market Hypotheses - EMH, and the Zero-Sum Game theory.

1.1 Efficient Market Hypothesis - EMH

In financial economics, the efficient market hypothesis (EMH) is an investment theory which states it is impossible for an investor to "beat the market" because stock market efficiency causes existing share prices to always incorporate and reflect all relevant information. In accordance to the EMH, stocks always trade at their fair value on stock exchanges and only react to new information or charges in discount rates, making it impossible for investors to make a profit by either purchasing undervalued stocks or selling stocks for inflated prices. This means that as such, it should be impossible to outperform the overall market through expert stock selection or market timing, and the only way an investor can possibly obtain higher returns is by purchasing riskier investments.

1.1.1 Breaking down EMH

Although it is a cornerstone of modern financial theory, the EMH is highly controversial and often disputed. Believers argue it is pointless to search for undervalued stocks or to try to predict trends in the market through either fundamental or technical analysis.

While academics point to a large body of evidence in support of EMH, an equal amount of dissension also exists. For example, investors such as Warren Buffett have consistently beaten the market over long periods of time, which in itself is impossible by definition according to the EMH. Detractors of the EMH also point to events such as the 1987 stock market crash, when the Dow Jones Industrial Average (DJIA) fell by over 20% in a single day, which was clear evidence that stock prices can seriously deviate from their fair values.

1.1.2 What EMH means for investors

Proponents of the EMH conclude that, because of the randomness of the market, investors would be better off by investing in a low-cost, passive portfolio such as one comprising of various low risk index funds. Data compiled by Morningstar Inc. through its June 2015 Active/Passive Barometer study supports the conclusion. Morningstar

compared active managers returns in all categories against a composite made of related index funds and exchange-traded funds (ETFs). The study found that year-over-year, only two groups of active managers successfully outperformed passive funds more than 50% of the time. These were U.S. small growth funds and diversified emerging markets funds.

In all of the other categories, including U.S. large blend, U.S. large value, and U.S. large growth, among others, investors would have fared better by investing in low-cost index funds or ETFs. While a percentage of active managers do outperform passive funds at some point, the challenge for investors is being able to identify which ones will do so. Less than 25% of the top-performing active managers are able to consistently outperform their passive manager counterparts.

1.2 Zero-Sum Game

Zero-sum, not to be confused with Empty sum, or Zero game, is a mathematical representation of a situation found in game theory and economic theory, in which one persons gain is equivalent to anothers loss, so the net change in wealth or benefit is zero. A zero-sum game may have as few as two players, or millions of participants.

Zero-sum games are found in game theory, but are less common than non-zero sum games. Poker and gambling are popular examples of zero-sum games since the sum of the amounts won by some players equals the combined losses of the others. Games such as chess and tennis, in which there is one winner and one loser, are also zero-sum games. In the financial markets, options and futures are examples of zero-sum games, excluding transaction costs. For every person who gains on a contract, there is a counter-party who loses.

1.2.1 Zero-Sum Game & Economics

When the Zero-Sum Game theory is applied specifically to economics, there are multiple factors to consider in oder to understand a zero-sum game. A zero-sum game assumes a version of perfect competition and perfect information; that is, both opponents in the model have all the relevant information to make an informed decision. To take a step back, most transactions or trades are inherently non zero-sum games because when two parties agree to trade they do so with the understanding that the goods or services they are receiving are more valuable than the goods or services they are trading for it, after transaction costs. This is called positive-sum, and most transactions fall under this category.

Options and futures trading is the closest practical example to a zero-sum game scenario. Options and futures are essentially informed bets on what the future price of a certain commodity will be in a strict timeframe. While this is a very simplified explanation of options and futures, generally if the price of that commodity rises within that timeframe, you can sell the futures contract at a profit. Thus, if an investor makes money off of that bet, there will be a corresponding loss. This is why futures and options trading often comes with disclaimers to not be undertaken by inexperienced traders. However, futures and options provide liquidity for the corresponding markets and can be very successful, for the right investor or company that is.

It is important to note that the stock market overall is often considered a zero-sum game, which is a misconception, along with other popular misunderstandings. Historically and in contemporary culture the stock market is often equated with gambling, which is definitely a zero-sum game. When an investor buys a stock, it is a share of ownership of a company that entitles that investor to a fraction of the company's profits. The value of a stock can go up or down depending on the economy and a host of other factors, but ultimately, ownership of that stock will eventually result in a profit or a loss that is not based on chance or the guarantee of someone else's loss. In contrast, gambling means that somebody wins the money of another who loses it.

There are other such myths regarding the stock market, some of which include: falling stocks must go up again at some point and stocks that go up must come down, as well as that the stock market is exclusively for the extremely wealthy.

Chapter 2

Background Theory

Computational finance is a branch of applied computer science that deals with problems of practical interest in finance. Some slightly different definitions are the study of data and algorithms currently used in finance and the mathematics of computer programs that realize financial models or systems. Using computational finance in order to allocate assets in a portfolio is not at all unheard of and was first documented 1952.[1] Markowitz first introduced the concept of portfolio selection as an exercise in mean-variance optimisation. This required more computer power than was available at the time, so he worked on useful algorithms for approximate solutions. He theorised that risk-averse investors could construct portfolios to optimise or maximise expected return based on a given level of market risk, emphasising that risk is an inherent part of higher reward. According to his theory, it's possible to construct an "efficient frontier" of optimal portfolios offering the maximum possible expected return for a given level of risk.

2.1 Time Series Analysis

A time series is a series of data points which may be indexed, listed, or graphed, in a time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average. Brockwell et al provide a formal description of time series as having a set of observations xt, each one being recorded at a specific time t. A discrete-time time series (the type to which this book is primarily devoted) is one in which the set T0 of times at which observations are made is a discrete set, as is the case, for example, when observations are made at fixed time intervals. Continuous time series are obtained

when observations are recorded continuously over some time interval, e.g., when T0 = [0, 1].[2]

Significant "time series momentum" has also been documented in equity index, currency, commodity, and bond futures for each of the 58 liquid instruments they consider.[3] They find persistence in returns for 1 to 12 months that partially reverses over longer horizons, consistent with sentiment theories of initial under-reaction and delayed over-reaction. A diversified portfolio of time series momentum strategies across all asset classes delivers substantial abnormal returns with little exposure to standard asset pricing factors and performs best during extreme markets. Examining the trading activities of speculators and hedgers, they find that speculators profit from time series momentum at the expense of hedgers.

2.1.1 ARMA

In the statistical analysis of time series, autoregressivemoving-average (ARMA) models provide a parsimonious description of a weakly stationary stochastic process in terms of two polynomials, one for the autoregression and the second for the moving average. The notation ARMA(p, q) refers to the model with p autoregressive terms and q moving-average terms. This model contains the AR(p) and MA(q) models,

$x_t = c + \varepsilon_t + \sum_{i=1}^{q} \varphi_i X_{t-i} \theta_i \varepsilon_{t-i}$

Forecasting interest rates is of great concern for financial researchers, economists, and players in the fixed income markets. A study was carried out to develop an appropriate model for forecasting the short-term interest rates, implicit yield on 91 day treasury bill, overnight MIBOR rate, and call money rate.[4] The short-term interest rates are forecasted using univariate models such as the Random Walk, ARIMA, ARMA-GARCH, and ARMA-EGARCH. The appropriate model for forecasting is determined considering a six-year period from 1999. The results show that interest rates time series have volatility clustering effect and hence GARCH based models are more appropriate to forecast than the other models. Radha et al found that for commercial paper rate ARIMA-EGARCH model is the most appropriate model, while for implicit yield 91 day Treasury bill, overnight MIBOR rate, and call money rate, the ARIMA-GARCH model is the most appropriate model for forecasting.

2.1.2 ARIMA

In time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an ARMA model. Given a time series of data X_t where t is an integer index and the X_t are real numbers, an ARMA(p, q) model is given by

$$x_t - \alpha_1 X_{t-1} - \ldots \alpha_{p'} X_{t-p'} = \varepsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

The aforementioned model is fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationarity.

The existence of weak-form efficiency in the Russian stock market is examined for the period 1st September 1995 to 1st May 2001 using daily, weekly and monthly Russian Trading System index time series.[5] Several different approaches are used to assess the predictability of the RTS index time series. Unit root, autocorrelation and variance ratio tests are conducted for the null hypothesis of a random walk model. The results support the null hypothesis for the monthly data only. Further analysis is performed for the daily and weekly data. Linear and non-linear modelling of the serial dependence is conducted using ARIMA and GARCH models estimated on the in-sample period 1st September 1995 to 1st January 2001. Forecasts based on the best fitting models are performed for the out-of-sample period 2nd January 2001 to 1st May 2001. Comparisons of the forecasts reveal that none of the models outperforms the others, and the most accurate forecasts are obtained for just the first out-of-sample observation. Whilst our research results provide some limited evidence of short-term market predictability on the RTS, there is insufficient evidence to suggest that it would lead to a profitable trading rule, once transaction costs and risk are taken into account.

The main intention of this paper is to investigate, with new daily data, whether prices in the two Chinese stock exchanges (Shanghai and Shenzhen) follow a random-walk process as required by market efficiency.[6] We use two different approaches, the standard variance-ratio test of Lo and MacKinlay (1988) and a model-comparison test that compares the ex post forecasts from a NAIVE model with those obtained from several alternative models (ARIMA, GARCH and Artificial Neural Network-ANN). To evaluate ex post forecasts, we utilize several procedures including RMSE, MAE, Theil's U, and encompassing tests. In contrast to the variance-ratio test, results from the model-comparison approach are quite decisive in rejecting the random-walk hypothesis in both Chinese stock markets. Moreover, our results provide strong support for the ANN as a potentially useful device for predicting stock prices in emerging markets.

2.2 Statistical Machine Learning

A machine learning algorithm is an algorithm that is able to learn through examples from data. To understand what an algorithm is, Cormen et al informally describe algorithms as "any well-defined computational procedures which takes some value, or set of values, as input and produce some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output."[7] In simple terms, it is possible to say that an algorithm is a sequence of steps which allow to solve a certain task. Similarly to a normal algorithm, a machine learning algorithm as defined formally by Tom M. Mitchell, states that "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."[8]

2.2.1 Classification

Classifiers are a class of machine learning algorithms which identify in which category a new observation belongs to, on the basis of a training set of data containing observations whose category membership is known. A model based on discriminant analysis was sought out to categorise a stock as manipulated or non-manipulated based on certain key variables that capture the characteristics of the stock.[9] The model in which Murugesan et al chose, helps them identify stocks witnessing activities that are indicative of potential manipulation irrespective of the type of manipulation, such as action-based, information-based, or trade-based. The model which they proposed, helps investigators to arrive at a shortlist of securities that are potentially manipulated and which could be subject to further detailed investigation to detect the type and nature of the manipulation, if any. In a market like India, where there are about 5000 plus securities listed on its major exchanges, it becomes extremely difficult to monitor all securities for potential market abuse. Academics who have earlier used discriminant analysis have used the Linear Classification Function without validating the assumption that governs the model. Through their research, they have tested the assumption on data from the Indian capital market and found that the data does not comply with the assumptions that govern the use of the linear classification function. This therefore resulted in them using the Quadratic Classification Function, which is the appropriate technique for instances where the data does not meet the sated assumptions, to categorise stocks into two categories, namely manipulated and non-manipulated.

2.2.2 Support Vector Machines

Support vector machines (SVM), are a class of machine learning algorithms that have become incredibly popular in the past few years. SVMs are very similar to classifiers in the sense that they also classify data by drawing a line, called a decision boundary, to separate them. However, SVMs go a step further by calculating a vector from the data point with the smallest margin to the decision boundary. This is called a support vector. There exists vast research articles which predict the stock market as well pricing of stock index financial instruments but most of the proposed models focus on the accurate forecasting of the levels of the underlying stock index. There is a lack of studies examining the predictability of the direction of stock index movement. Given the notion that a prediction with little forecast error does not necessarily translate into capital gain, the authors of this research attempt to predict the direction of the S&P CNX NIFTY Market Index of the National Stock Exchange, one of the fastest growing financial exchanges in developing Asian countries.[10] Random forest and Support Vector Machines (SVM) are very specific type of machine learning method, and are promising tools for the prediction of financial time series. The tested classification models, which predict direction, include linear discriminant analysis, logit, artificial neural network, random forest, and SVM. Empirical experimentation suggests that the SVM outperforms the other classification methods in terms of predicting the direction of the stock market movement and random forest method outperforms neural network, discriminant analysis and logit model used in their study.

2.2.3 Regression

Regression analysis is widely used for prediction and forecasting to understand which among the independent variables are related to the dependent variable while also exploring the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables. Regression analysis helps one understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Kakushadze et al provide a systematic quantitative framework in what is intended to be a pedagogical fashion for discussing mean-reversion and optimisation.[11] In their paper, they start off their research with pair trading and add complexity by following the sequence mean-reversion via demeaning, regression, weighted regression, (constrained) optimization, factor models. They discuss in further detail how to conduct mean-reversion based on this approach, including common pitfalls encountered in practical applications, such as the difference between

maximising the Sharpe ratio and minimising an objective function when trading costs are included. Kakushadze et al also discuss explicit algorithms for optimization with linear costs, constraints and bounds, and also illustrate their discussion on an explicit intraday mean-reversion alpha.

2.2.4 Decision Trees

Decision tree learning uses a decision tree as a predictive model which maps observations about an item, represented in the branches, to conclusions about the item's target value represented in the leaves. Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values, typically real numbers, are called regression trees. Creamer et al propose a multi-stock automated trading system which relies on a layered structure consisting of a machine learning algorithm, an online learning utility, and a risk management overlay.[12] An alternating decision tree (ADT), which is implemented with Logitboost, was chosen as their underlying algorithm. One of the strengths of their approach is that the algorithm is able to select the best combination of rules derived from well-known technical analysis indicators and is also able to select the best parameters of the technical indicators. Additionally, their online learning layer combines the output of several ADTs and suggests a short or long position. Finally, the risk management layer in which they implemented, can validate the trading signal when it exceeds a specified non-zero threshold and limit the application of their trading strategy when it is not profitable. They tested the expert weighting algorithm with data of 100 randomly selected companies of the S&P 500 index during the period 20032005. They found that their algorithm generates abnormal returns during the test period. Their experiments show that the boosting approach was able to improve the predictive capacity when indicators were combined and aggregated as a single predictor. Even more, the combination of indicators of different stocks demonstrated to be adequate in order to reduce the use of computational resources, and still maintain an adequate predictive capacity.

2.3 Bayesian Statistics

Bayesian Statistics, a form of probabilistic programming, describes probabilistic models and then performs inference in those models. Probabilistic reasoning is a foundational technology of machine learning and has been used by companies such as Google, Amazon, and Microsoft. Probabilistic reasoning has been used for predicting stock prices,

recommending movies, diagnosing computers, detecting cyber intrusions, and image detection. Gelman et al defines bayesian inference as the process of fitting a probability model to a set of data and summarising the result by a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations.[13]

2.3.1 Markov Chain Monte Carlo

In statistics, Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution of its equilibrium distribution. The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.

A 1993 paper was prepared for the purpose of presenting the methodology and uses of the Markov Chain Monte Carlo simulation technique as applied in the evaluation of investment projects to analyse and assess risk.[14] In the first part of his paper, he highlights the importance of risk analysis in investment appraisal. The author follows by presenting the various stages in the application of the risk analysis process and examines the interpretation of the results generated by a risk analysis application including investment decision criteria and various measures of risk based on the expected value concept. In the final part of his paper, he draws some conclusions regarding the usefulness and limitations of risk analysis in investment appraisal.

Stochastic volatility models are increasingly important in practical derivatives pricing applications, yet relatively little work has been undertaken in the development of practical Monte Carlo simulation methods for this class of models. This paper considers several new algorithms for time-discretization and Monte Carlo simulation of Heston-type stochastic volatility models.[15] The algorithms are based on a careful analysis of the properties of affine stochastic volatility diffusions, and are straightforward and quick to implement and execute. Tests on realistic model parameterizations reveal that the computational efficiency and robustness of the simulation schemes proposed in the paper compare very favourably to existing methods.

Bond yields today are well below and stock market valuations are well above their historical average.[16] There are no historical periods in the United States where comparable low bond yields and high equity valuations have occurred simultaneously. Both current bond yields and stock values have been shown to predict near-term returns. Portfolio returns in the first decade of retirement have an outsize impact on retirement income strategies. Traditional Monte Carlo simulation approaches generally do not incorporate

market valuations into their analysis. In order to simulate how retirees will fare in a low return environment for both stocks and bonds, Blanchett et al incorporate the predictive ability of current valuations to simulate its impact on retirement portfolios. Blanchett et al estimate bond returns through an autoregressive model that uses an initial bond yield value where yields drift in the future. Blanchett et al use the cyclically adjusted price-to-earnings (CAPE) ratio as an estimate of market valuation to predict short-run stock performance. Our simulations indicate that the safety of a given withdrawal strategy is significantly affected by the initial bond yield and CAPE value at retirement, and that the relative impact varies based on the portfolio equity allocation. Using valuation measures current as of April 15, 2013, which is a bond yield of 2.0% and a CAPE of 22, Blanchett et al find the probability of success for a 40% equity allocation with a 4% initial withdrawal rate over a 30 year period is approximately 48%. This success rate is materially lower than past studies and has sobering implications on the likelihood of success for retirees today, as well as how much those near retirement may need to save to ensure a successful retirement.

This paper provides a numerical approach based on a Monte Carlo simulation for valuing dynamic capital budgeting problems with many embedded real options dependent on numerous state variables.[17] Schwartz et al propose a way of decomposing a complex capital budgeting problem with many options into a set of simple options, suitably accounting for interaction and interdependence among them. The decomposition approach is numerically implemented using an extension of the Least Squares Monte Carlo algorithm, presented by Longstaff and Schwartz (2001) applied to our multi-option setting. Schwartz et al also provide a number of applications of our approach to well-known real options models and real life capital budgeting problems. Moreover, Schwartz et al present a set of numerical experiments to provide evidence for the accuracy of the proposed methodology.

Chapter 3

Experminetal Setup

3.1 Data Tidying

A data set containing end of day stock prices, dividends, and splits for 3,000 US companies, curated by the Quandl community, and released into the public domain, was used. The date column in the CSV file was loaded into memory, and said column was converted to a date data type. The DataFrame was then sorted using the date column, starting from the oldest date, ending with the latest. The date column was also set to the index. The DataFrame was split into two, the training data set consisting of 80%, and the test data set consisting of 20% of the original DataFrame.

3.2 Stock Selection

The pairwise correlation of all the columns in the DataFrame was computing and stored in a new DataFrame. A list of stock pairs with low correlation were extracted.

3.3 Time Series Analysis

The selected stocks was extracted from the data set and stored in a DataFrame. The log returns of the stocks were calculated by calculating the logarithm of the stock's adjusted close price divided by the following day's adjusred close price. The resulting values from the calculation were then stored in a new column in the DataFrame and all infinite values were dropped from the series.

3.3.1 Random Walk

The first difference of the stocks were calculated abd stored in a new column in the Dataframe and all infinite valued were dropped.

3.3.2 Ordinary Least Squares (OLS)

The series was fitted to an OLS model. The 15 day SMA was used as a nobs x k array where nobs is the number of observations and k is the number of regressors, to train the model, while the adjusted close price of the stock was used as the dependent variable for the model to predict. The mean absolute error, mean squared error, median absolute error, and r2 score were used as metrics in order to rank the performance of the model's prediction capabilities in in-sample testing.

3.3.3 Auto Regressive (AR)

The notation AR(p) indicates an autoregressive model of order p. The AR(p) model is defined as

$x_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$

$\varphi_1, \ldots, \varphi_p$ are the parameters of the model, c is a constant, and ε_t is white noise.

The series was fitted to an AR(p) model with a maximum lag value of 30. The IC was used to fit the AR model in order to select the optimal lag length. No constants were passed when fitting the AR model to the series. The optimal lag for the fit of the AR model was then calculated using the same paramaeters passed when fitting the AR model.

3.3.4 Moving Average (MA)

The notation MA(p, q) indicates a moving average model of order (p, q). The MA(p, q) model is defined as

$x_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$

where the $\theta_1, \ldots, \theta_q$ are the parameters of the model, μ is the expectation of X_t (often assumed to equal 0), and the $\varepsilon_t, \varepsilon_{t-1}, \ldots$ are again, white noise error terms.

The series was fitted to an MA(p, q) model with an order selected based on the lowest AIC. A maximum lag of 30 was once again passed to the MA model with no constant.

The exact loglikelihood for the fit of the MA model was maximized via the Kalman Filter.

3.3.5 Auto Regressive Moving Average (ARMA)

The series was fitted to an ARMA(p, q, r) model with an order selected based on the lowest AIC. No constants were passed to the ARMA model. The exact loglikelihood for the fit of the ARMA model was maximized via the Kalman Filter.

3.3.6 Auto Regressive Integrated Moving Average (ARIMA)

The series was fitted to an ARIMA(p, q, r) model with an order selected based on the lowest AIC. No constants were passed to the ARIMA model. The exact loglikelihood for the fit of the ARIMA model was maximized via the Kalman Filter.

3.4 Machine Learning

The data set was split for training and testing purposes when fitting and predicting data. The training data set consisted of 80% of the whole data set, while the test data set consisted of 20%.

3.4.1 Classification

2, 3, 4, 5, and 6 day SMAs were used as the features to train the models, while a binary value used to determine whether the current day's adjusted close has risen or not from the previous day was used as the target valued for the model to predict. In-sample testing was carried out using the models predict function, passing the test data set's features in order to predict the output. The classification report and confusion matrix were used as metrics in order to rank the performance of the model's prediction capabilities in in-sample testing. An algorithm was developed for out-of-sample testing. The algorithm iterates for a number of n steps, increasing the index by 1 each step, forecasting the next day's outcome, and calculating the SMAs.

3.4.1.1 Decision Tree

The decision tree classifier was fit using the Gini impurity criterion to measure the quality of the split. The model was given the liberty to select the best strategy in order to split

the tree at each node. The maximum allowed depth of the tree was left unrestricted, which was the same as for the maximum leaf nodes. A threshold of 1e-7 was used to terminate the tree growth to determine if a node is a leaf, if the impurity of a node is below the threshold, the node is a leaf. The minimum number of samples required to be at a leaf node was set to 1, while the minimum number of samples required to split an internal node was set to 2. The data fitted to the model was not pre-sorted, and the random number generator used by the model was that of Numpy's RandomState.

3.4.1.2 Boosted Decision Tree

The boosted decision tree was fit using the 'SAME.R' real algorithm which converges at a faster rate, achieving a lower test error with fewer boosting iterations. The maximum number of estimators at which boosting is terminated was set to 50; in case of perfect fit, the learning procedure is stopped early. The learning rate which is the contribution of each classifier of the model was shrunk by 1. The best estimator used to fit the data to the model was a Decision Tree Classifier, and the random number generator used by the model was that of Numpy's RandomState.

3.4.1.3 Support Vector Machine (SVM)

The C-Support Vector Classification implementation was used for the SVM model with a 0 penalty parameter of the error term. A kernal type of 'rbf' was used when fitted the model to the data, along with a polynomial kernel function degree of 3. The gama 'rbf' Kernel coefficient was calculated by dividing the number of features by 1, while no probability estimates were used when fitting. A shrinking heuristic was used and a tolerance of 1e-3 was used for stopping criterion. A cache size of 200MB was used for the kernel when fitting the model, and all classes were assigned a weight of one. Verbose output was not used, and the random number generator used by the model was that of Numpy's RandomState. No limits were set on the iterations within the solver, and a one-vs-rest decision function of shape (number of samples, number of classes) was returned.

3.4.1.4 Random Forest

The random forest was fit with bootstrap samples when building the trees, while all the weights associated were set to 1. The 'gini' function to measure the quality of a split, and no maximum depth of the tree was set, allowing the nodes to expand until all leaves are pure or until all leaves contain less than the minimum split samples. The number of

features to consider when looking for the best split was the square root of the number of passed, and no limit on the maximum leaf nodes for growing trees was set. A threshold of 1e-7 was used to terminate the tree growth to determine if a node is a leaf, if the impurity of a node is below the threshold, the node is a leaf. The minimum number of samples required to be at a leaf node was set to 1, and the minimum number of samples required to split an internal node was set to 2. The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node was set to 0, and the number of trees in the forest was set to 10. The number of jobs to use for the computation was set to 1, making use of only 1 CPU core, and out-of-bag samples to estimate the generalization accuracy were not used. The verbosity of the tree building process was not controlled, and the random number generator used by the model was that of Numpy's RandomState. The model was built using a cold start by not making use of the previous call to fit and add more estimators to the ensemble, meaning a whole new forest was fit instead.

3.4.1.5 K-Nearest Neighbour

The k-nearest neighbour was fit with a number of 5 neighbors to use for k-neighbours queries, with a uniform weight making all points in each neighbourhood weighted equally when carrying out predictions. The model was left at liberty to select the most appropriate algorithm based on the values passed when fitting the model to the data. The lead size of the model was set to 30, and a minkowski distance metric with p equal to 2 which equivalent to the standard Euclidean metric. The number of jobs to use for the computation was set to 1, making use of only 1 CPU core.

3.4.1.6 Logistic Regression

The logistic regression was fit with an inverse of regularization of strength 0 specifying stronger regularisation, while all the weights associated were set to 1. Dual formulation was used in conjunction with the l2 penalty with a liblinear solver. A constant (bias or intercept) was added to the decision function, and the intercept scalar was set to 1. The maximum number of iterations taken for the solvers to converge were set to 100, while the multiclass option used was ovr, thus fitting binary problem for each label. The number of jobs to use for the computation was set to 1, making use of only 1 CPU core, and the random number generator used by the model was that of Numpy's RandomState. The tolerance for stopping criteria was set to 0.0001, and verbosity was used with a value of 0 for the liblinear solver. The model was built using a cold start

by not making use of the previous call to fit and add more estimators to the ensemble, meaning a whole new forest was fit instead.

3.4.1.7 Bernoulli Naive Bayes

The model was fit with an additive (Laplace/Lidstone) smoothing parameter of 0 for no smoothing, and the threshold for binarising (mapping to booleans) of sample features was set to 0, and was presumed to already consist of binary vectors. The model was set to learn class prior probabilities, which means that the priors were adjusted according to the data.

3.4.1.8 Gaussian Naive Bayes

The model was fit to the data with no previous prior probabilities of the classes, thus the priors were not adjusted according to the data.

3.4.1.9 Neural Network

The model was fit to the data with 3 hidden layer of 100 neurons each using the rectified linear unit activation function for the hidden layer. The L2 penalty parameter used, which is the regularisation term, was set to 0.0001, and the size of minibatches for stochastic optimisers was set to the minimum of the two arguments being the number 200 and the number of samples. The solver used for weight optimisation was the 'adam' solver, which refers to a stochastic gradient-based optimiser. Since the 'adam' solver was used, the exponential decay rate for estimates of first moment vector was set to 0.9, while the second moment vector was set to 0.999. Early stopping was not used to terminate training when validation score is not improving whilst fitting the data to the model, and the value for numerical stability in the 'adam' solver was set to 1e-08. A constant learning rate was used for weight updates in conjunction with the initial learning rate, which controls the step-size in updating the weights, and was set to 0.001. The maximum number of iterations which the solver iterates until convergence was set to 200, and the momentum for gradient descent update was set to 0.9. Nesterov's momentum was used, and the exponent for the inverse scaling learning rate, which is used in updating the effective learning rate, was set to 0.5. A random number generator was not used, and the samples were shuffled in each iteration. The tolerance for optimisation was set to 0.0001 when the loss or score is not improving by at least the tolerance score set for two consecutive iterations in which convergence is considered to be reached and training is stopped. The validation fraction, which is the portion of training data to set aside

as validation set for early stopping, was set to 0.1. The model was built using a cold start by not making use of the previous call to fit as initialisation, meaning the previous solution was erased.

3.4.1.10 Stochastic Gradient Descent

The model was fit to the data using the ordinary least squares fit squared epsilon insensitive loss function, which ignores errors less than epsilon and is linear past that; this is the loss function used in SVR. The penalty used, also referred to regularisation term, was 'l2' which is the standard regularizer for linear SVM models, and the constant used to multiply the regularisation term was 0.0001. The elastic net mixing paramater was set to 0.15, while the fit intercept was estimated, meaning the data was not assumed to be already centered. A total of 5 passes over the training data, also known as epochs, were used, and the training data was shuffled after each epoch. No seed from the pseudo random number generated was used while shuffling the data, and a level 0 verbosity was used. The epsilon threshold in the epsilon-insensitive loss functions was set to 0.1, and any differences between the current prediction and the correct label were ignored if they were less than the threshold. The learning rate schedule used was 'invscaling', and the initial learning rate was set to 0.01. The exponent for inverse scaling learning rate was set to 0.25, and the model was built using a cold start by not making use of the previous call to fit as initialisation. The SGD weights of the model were not averaged.

3.4.2 Regression

15 and 50 day SMAs were used as the features to train the models, while the adjusted close price of the stock was used as the target valued for the model to predict. In-sample testing was carried out using the models predict function, passing the test data set's features in order to predict the output. The mean absolute error, mean squared error, median absolute error, and R^2 (coefficient of determination) score were used as metrics in order to rank the performance of the model's prediction capabilities in in-sample testing. An algorithm was developed for out-of-sample testing. The algorithm iterates for a number of n steps, increasing the index by 1 each step, forecasting the next day's outcome, and calculating the SMAs.

3.4.2.1 Decision Tree

The decision tree model was fit with a mean squared error, which is equal to variance reduction as feature selection criterion, and mae for the mean absolute error. The

maximum allowed depth of the tree was left unrestricted, which was the same as for the maximum leaf nodes. A threshold of 1e-7 was used to terminate the tree growth to determine if a node is a leaf, if the impurity of a node is below the threshold, the node is a leaf. The minimum number of samples required to be at a leaf node was set to 1, while the minimum number of samples required to split an internal node was set to 2. The data fitted to the model was not pre-sorted, and the random number generator used by the model was that of Numpy's RandomState. The model was given the liberty to select the best strategy in order to split the tree at each node.

3.4.2.2 Boosted Decision Tree

The boosted decision tree was fit with with a decision tree regressor as the base estimator from which the boosted ensemble is built. The maximum number of estimators at which the boosting is terminated was set to 50, and the learning rate which shrinks the contribution of each regressor was set to 1.0. The loss function used when updating the weights after each boosting iteration was set to linear, and the random number generator used by the model was that of Numpy's RandomState.

3.4.2.3 Random Forest

The random forest was fit with a mean absolute error, and bootstrap samples were used when building trees. The maximum features to consider when looking for the best split were left to the model to select the best number, and the number of jobs to run in parallel for both the fitting of the model and its predictions. The model was built using a cold start by not making use of the previous call to fit and add more estimators to the ensemble, meaning a whole new forest was fit instead. A threshold of 1e-7 was used to terminate the tree growth to determine if a node is a leaf, if the impurity of a ode is below the threshold, the node is a leaf. The minimum number of samples required to be at a leaf node was set to 1, while the minimum number of samples required to split an internal node was set to 2.

3.4.2.4 Linear Regression

The linear regression was fit with no normalised regressors, note that this makes the hyperparameters learnt more robust and almost independent of the number of samples. The number of jobs to use for the computation was set to 1, making use of only 1 CPU core. The intercept of the model was calculated which centres the data being fit to the model.

3.4.2.5 Neural Network

The model was fit to the data with the same paramaters as the Neural Network classifier.

3.4.2.6 Stochastic Gradient Descent

The model was fit to the data with the same paramaters as the SGD classifier.

3.5 Bayesian Statistics

The last 500 rows of the selected stocks were extracted from the data set and stored into a DataFrame. The log returns were calculated by dividing each day's adjusted close with the adjusted close of the following day, in logarithmic form. The resulting values from the said calculation were then stored in a new column in the DataFrame and all infinite values were dropped from the series.

3.5.1 No-U-Turn Sampler (NUTS)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam sodales tortor tortor. Nunc eget cursus dolor, id efficitur arcu. Maecenas posuere dictum nisi, non pulvinar nunc aliquam eu. Sed egestas, leo nec molestie hendrerit, nulla purus mattis lectus, eu rhoncus ipsum nulla vitae lectus. Integer mollis ligula ut risus maximus, ut ultrices tellus aliquet. Ut semper laoreet enim facilisis vulputate. Fusce ullamcorper a nisi iaculis pellentesque. Duis vel magna quis justo cursus fermentum vitae ac libero. Maecenas eget arcu et neque egestas scelerisque. Sed hendrerit at augue id interdum. Phasellus blandit tempus nunc ac feugiat. Vivamus egestas augue nec erat vestibulum elementum. Sed ac metus eu nunc consectetur mattis id ut lorem.

3.5.2 Metropolis-Hastings

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam sodales tortor tortor. Nunc eget cursus dolor, id efficitur arcu. Maecenas posuere dictum nisi, non pulvinar nunc aliquam eu. Sed egestas, leo nec molestie hendrerit, nulla purus mattis lectus, eu rhoncus ipsum nulla vitae lectus. Integer mollis ligula ut risus maximus, ut ultrices tellus aliquet. Ut semper laoreet enim facilisis vulputate. Fusce ullamcorper a nisi iaculis pellentesque. Duis vel magna quis justo cursus fermentum vitae ac libero. Maecenas eget arcu et neque egestas scelerisque. Sed hendrerit at augue id interdum. Phasellus

blandit tempus nunc ac feugiat. Vivamus egestas augue nec erat vestibulum elementum. Sed ac metus eu nunc consectetur mattis id ut lorem.

3.6 Strategy

The automated algorithmic strategies were run over the period of 01/05/2014. An ordered dictionary was used to store a series of data frames container the price data of the stocks selected. The data was tidied where the columns 'open', 'high', 'low', 'close', 'ex-dividend', and 'split_ratio' were dropped from each data frame and the columns 'ticker', 'adj_open', 'adj_high', 'adj_low', and 'adj_close' were renamed to 'sid', 'open', 'high', 'low', and 'close' respectively. The ordered dictionary was converted into a panel and passed to the trading algorithm. An ordered dictionary was used to store a boolean value for each ticker to determine whether the stock has already been invested in or not. All boolean values were set to false.

3.6.1 Classification

An ordered dictionary was used to store the day's open and close price for each stock as the backtester simulated each trading day. The algorithm was only allowed to run once there was enough price data, the amount chosen was 6. The 2, 3, 4, 5, and 6 day SMAs were calculated and each stored in an array. The six arrays were converted into an array of tuples, where the i-th tuple contains the i-th element from each of the argument sequences or iterables. An array was created to store a 1 if the cloase price increased from the open price, while a 0 if it decreased. The independent variables, the SMAs, and the dependant variables, the binary outcome for the particular stock, were passed to the machine learning algorithm to predict the next day's close price. An order of 10% of the current portfolio value was placed on a stock if the algorithm predicted an increase in the following day's price. The boolean value to determine whether the stock has already been invested in was set to true. The stock was shorted 10% of the current portfolio value if the algorithm predicted a decrease in the following day's price. The boolean value to determine whether the stock has already been invested in was set to false.

3.6.2 Regression

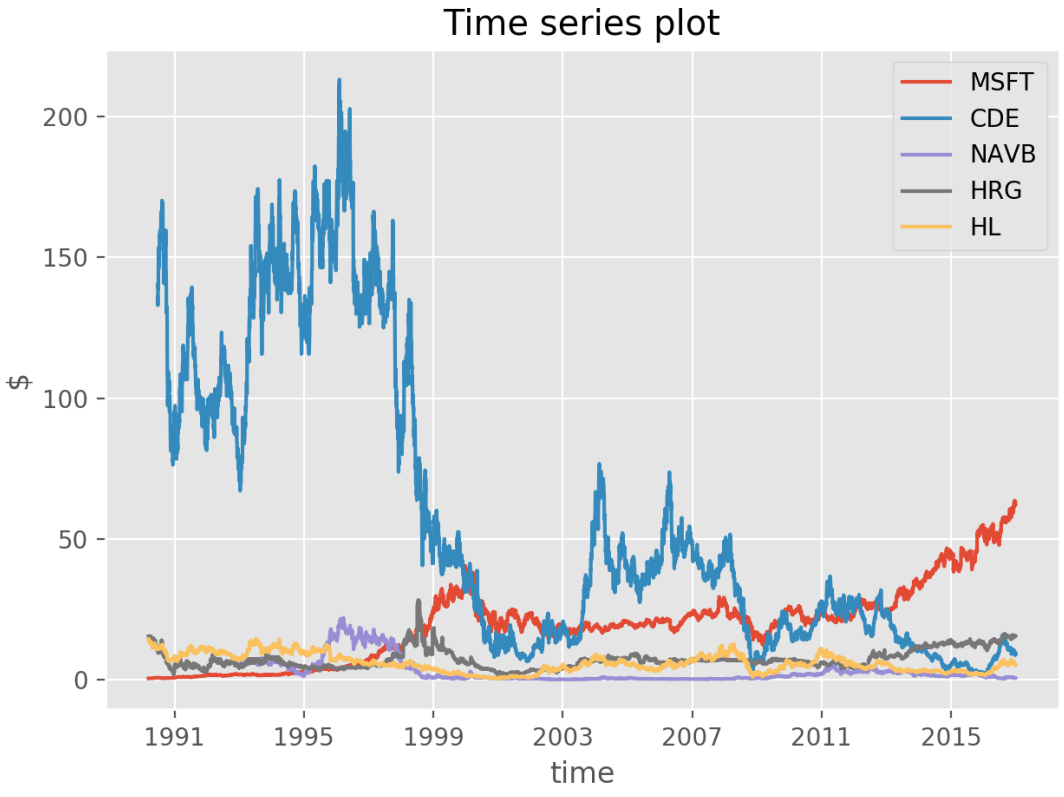
An ordered dictionary was used to store the day's close price for each stock as the backtester simulated each trading day. The algorithm was only allowed to run once there was

enough price data, the amount chosen was 50. The 15 and 50 day SMAs were calculated and each stored in an array. The two arrays were converted into an array of tuples, where the i-th tuple contains the i-th element from each of the argument sequences or iterables. The independent variables, the SMAs, and the dependant variables, the close prices for the particular stock, were passed to the machine learning algorithm to predict the next day's close price. The next day's 15 and 50 day SMAs were calculated using the predicted price and the following day's price predicted again. This process was iterated over 100 times in order to predict the close price of the following 100 days. An order was placed on a stock if the difference of the last predicted price and the current day's predicted price was higher than 10. The amount of the stock was determined based on the difference, the higher the difference, the larger the percentage of the allocation. The boolean value to determine whether the stock has already been invested in was set to true. The stock was shorted if the difference of the last predicted price and the current day's predicted price was higher than -10. The amount of the stock was determined based on the difference, the higher the difference, the larger the percentage of the allocation. The boolean value to determine whether the stock has already been invested in was set to false.

Chapter 4

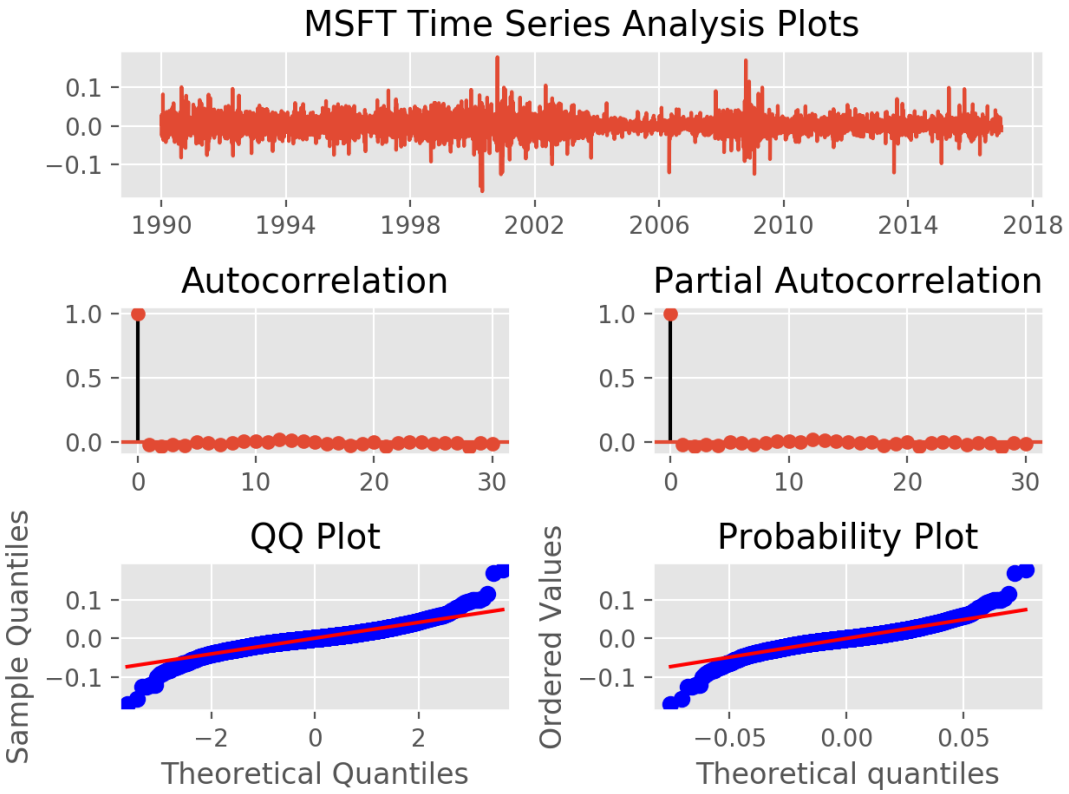
Research Findings

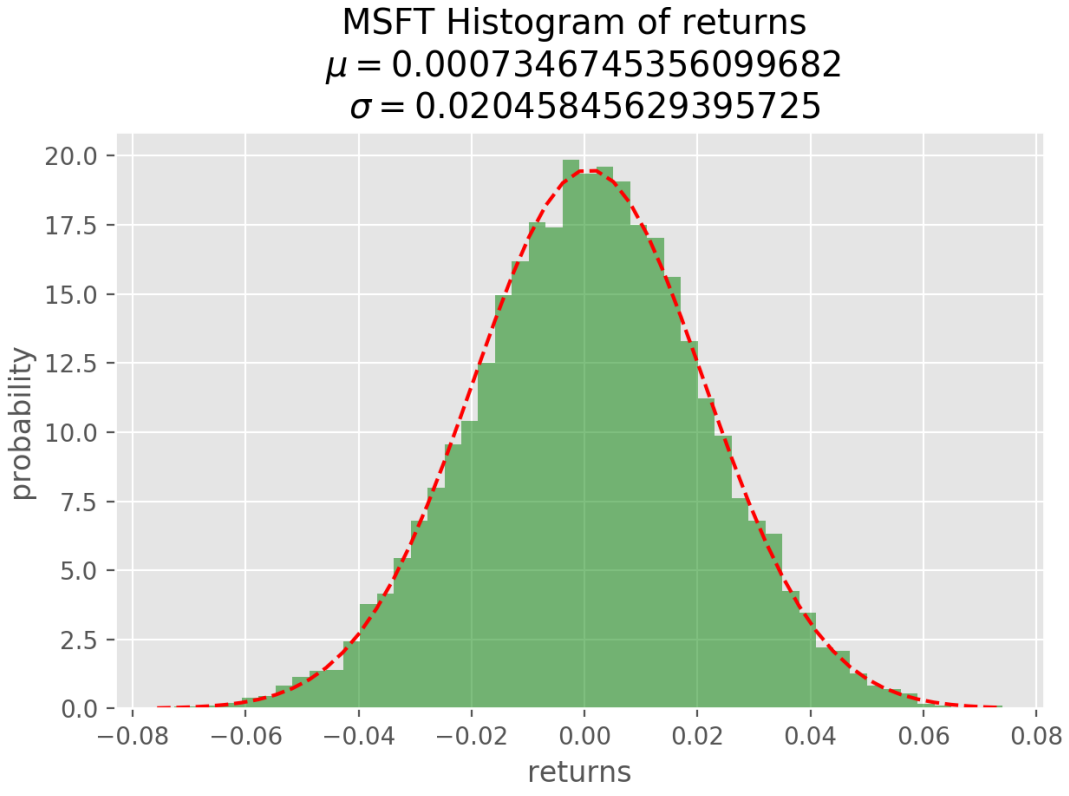
For the purpose of bencharking the performance of the algorithms, a total of five stocks from the basket of uncorrelated stocks were selected. These were MSFT, CDE, NAVB, HRG, and HL.

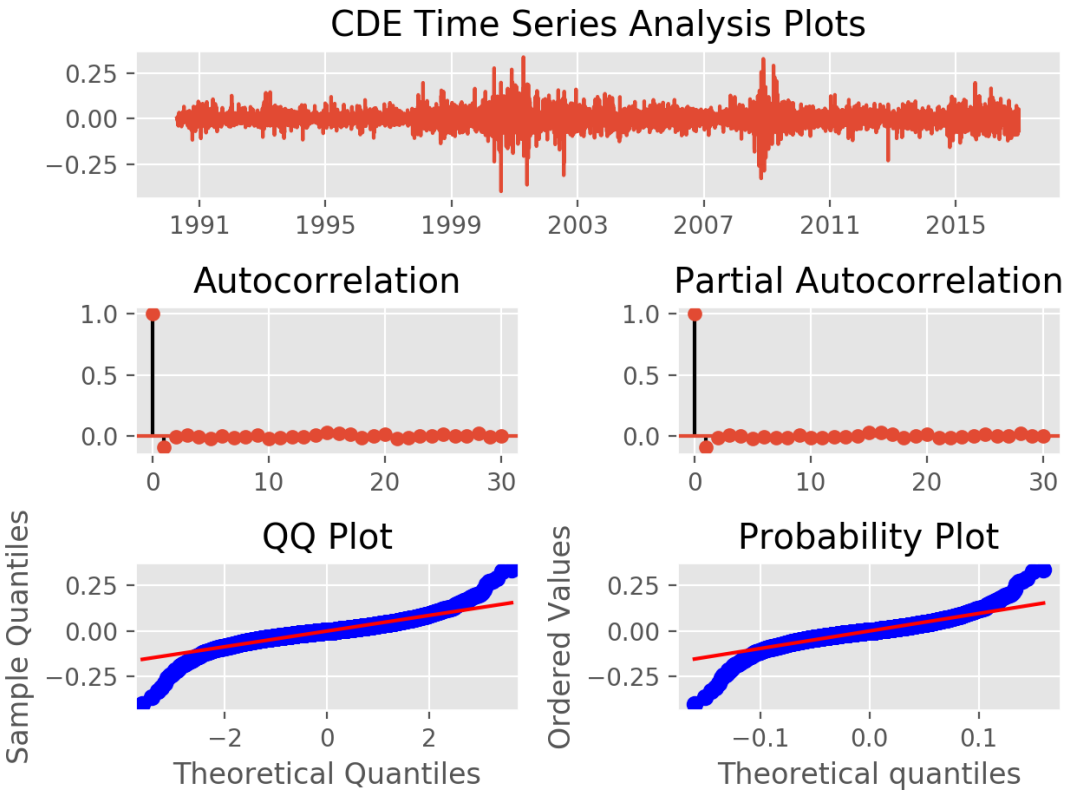


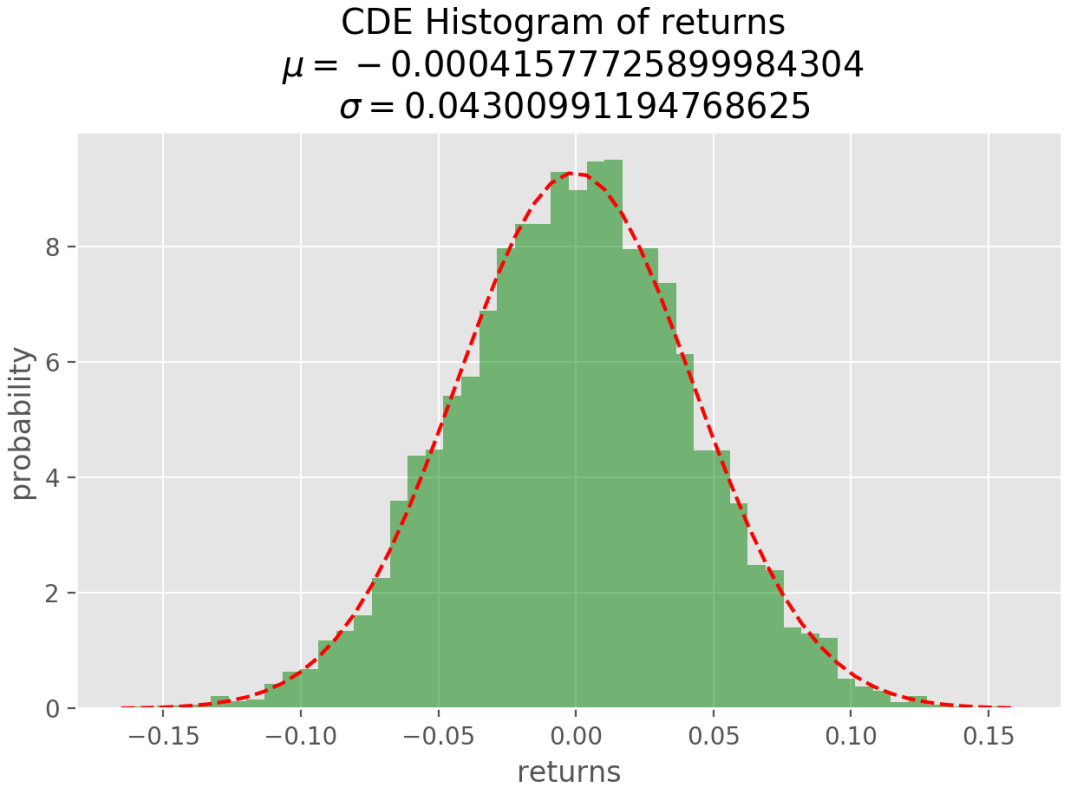
4.1 Time Series Analysis

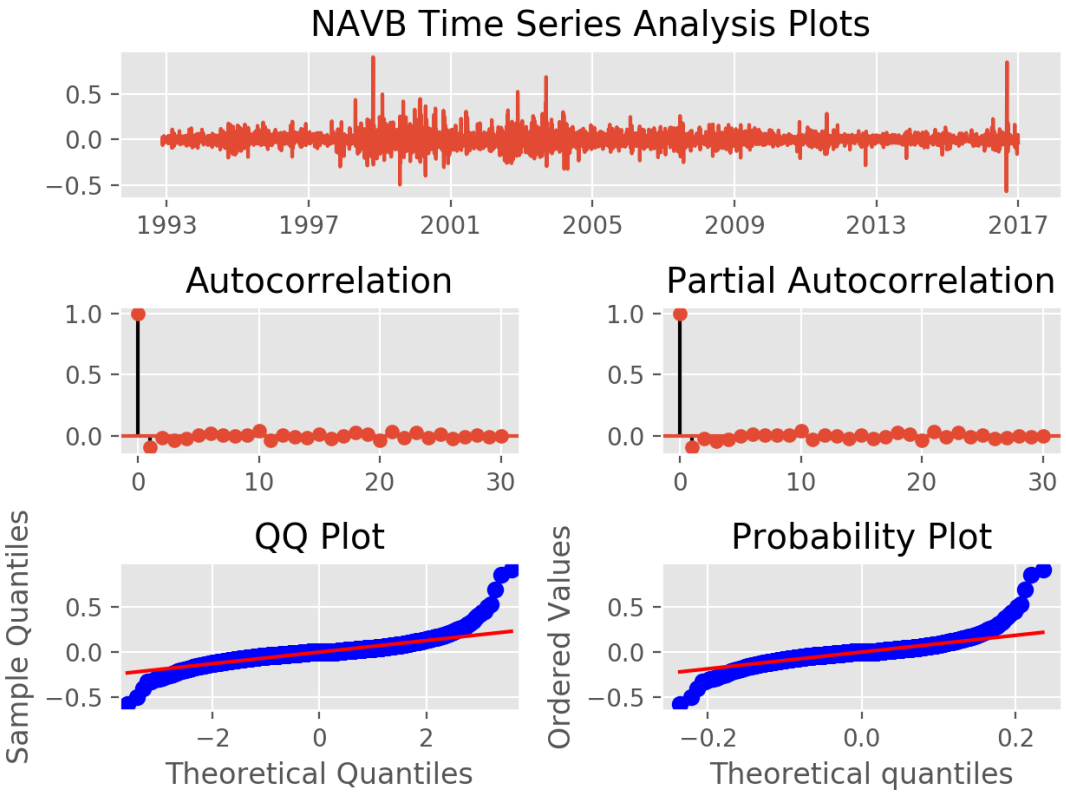
4.1.1 Random Walk

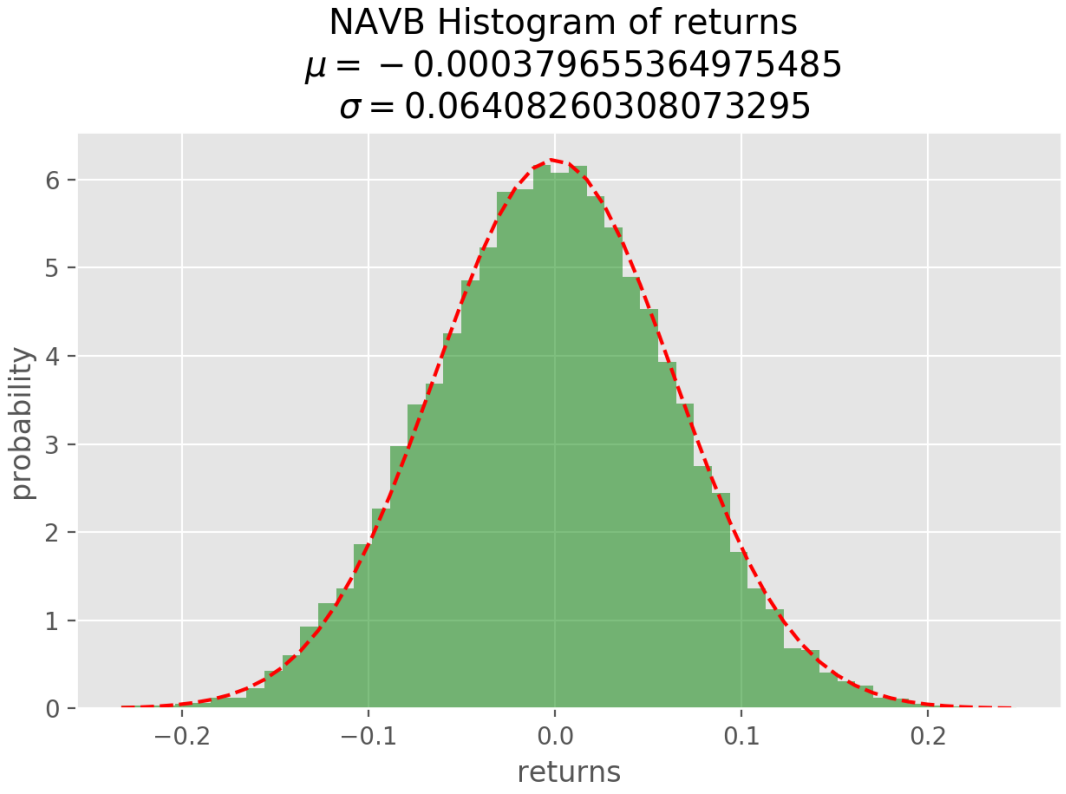


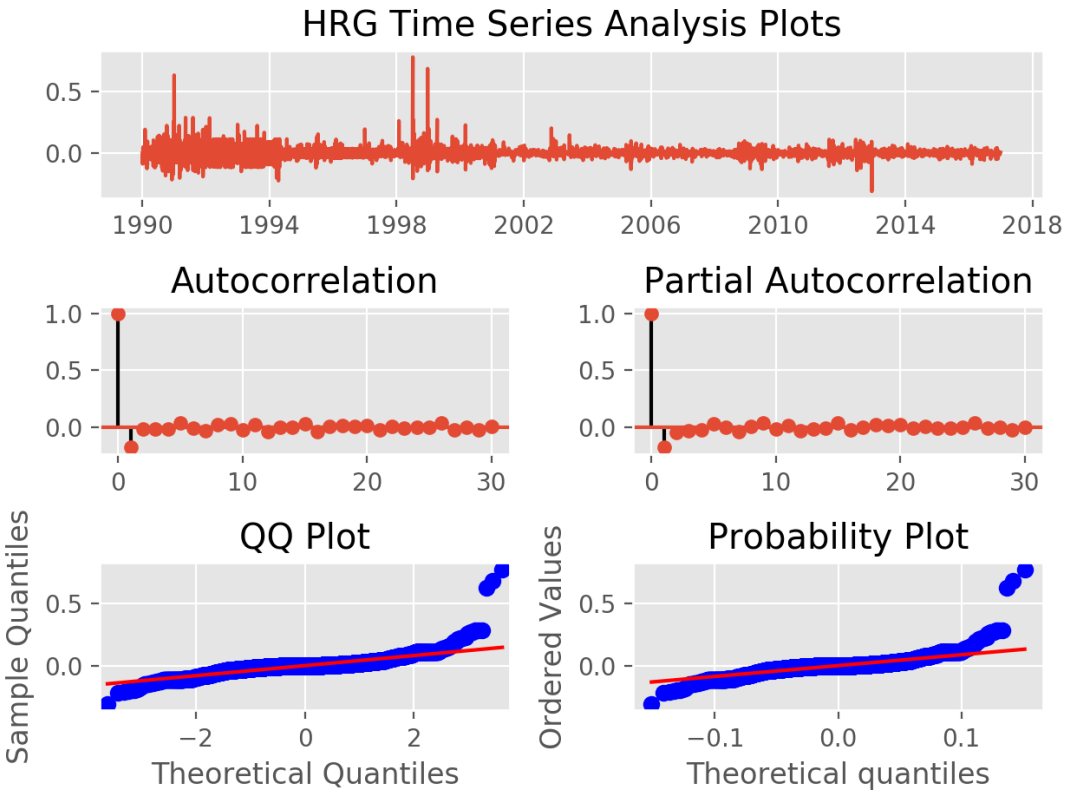


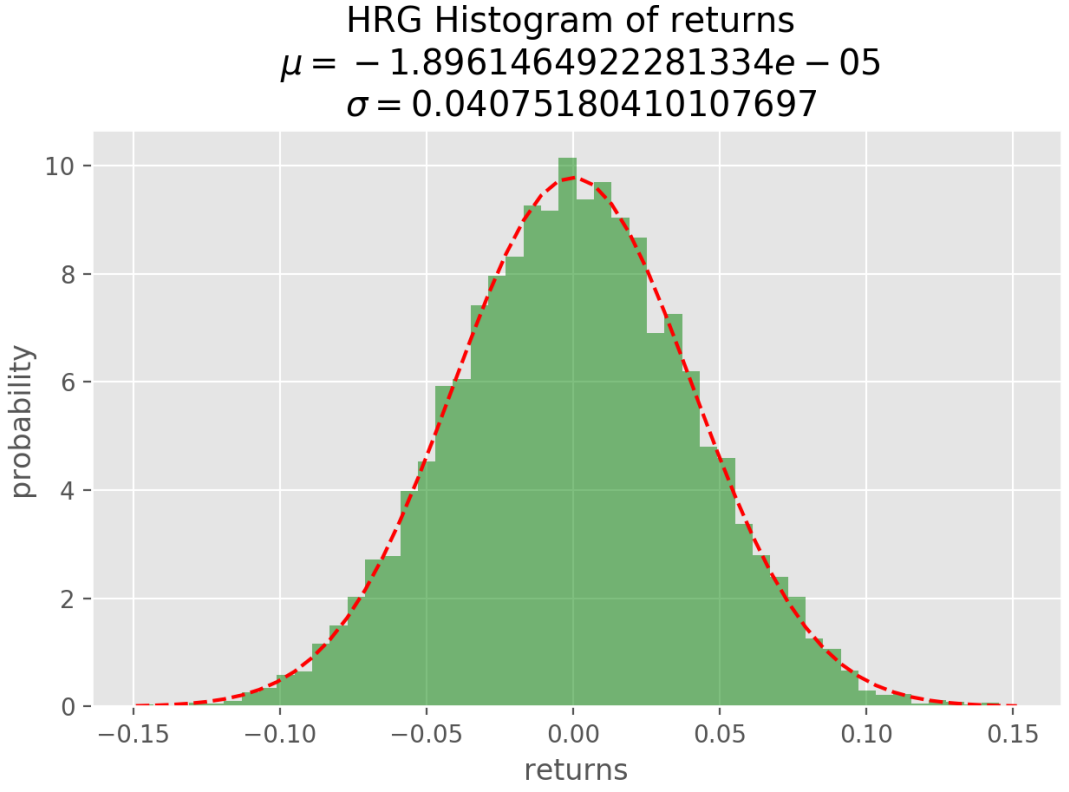

CDE Time Series Analysis Plots
0.25
0.00
−0.25
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

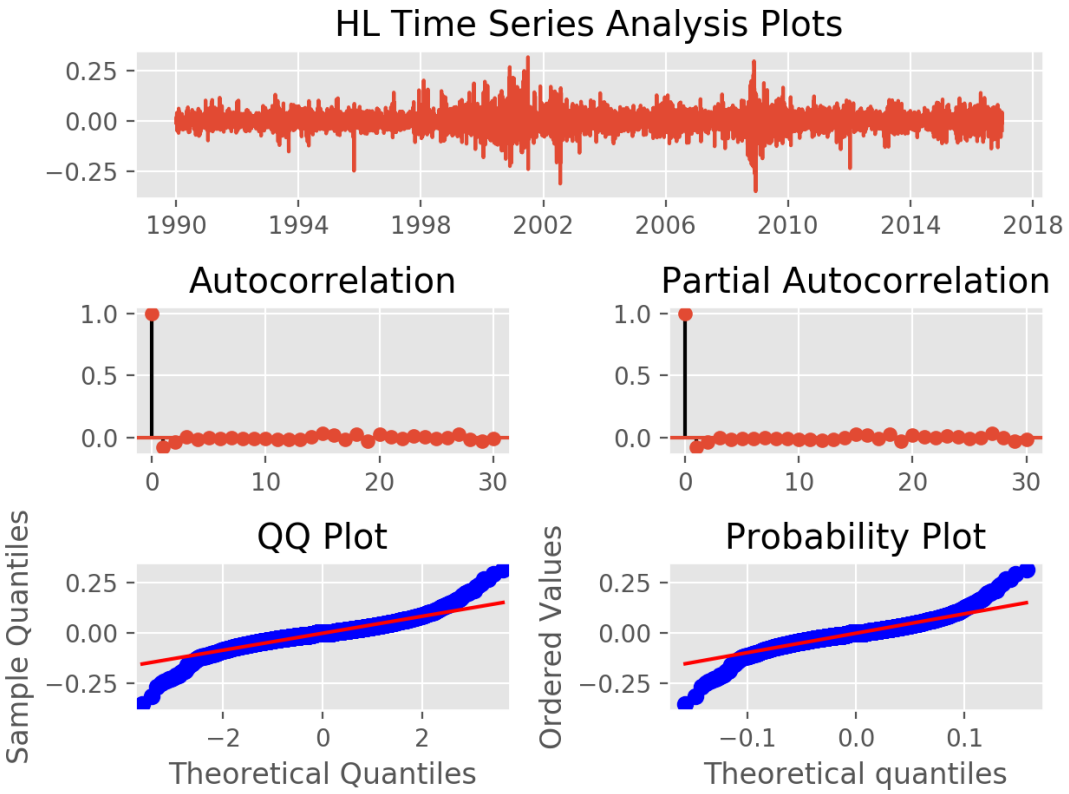

CDE Histogram of returns
$\mu = -0.00041577725899984304$
$\sigma = 0.04300991194768625$
probability
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

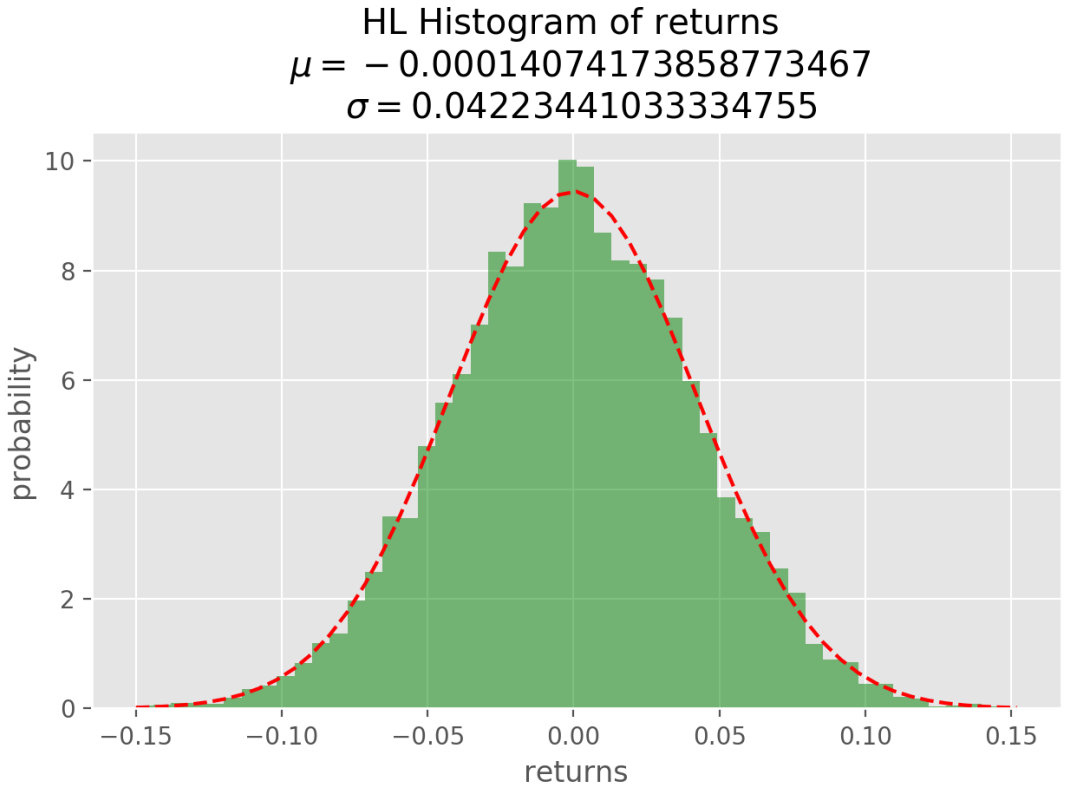

NAVB Time Series Analysis Plots
0.5
0.0
−0.5
1993
1997
2001
2005
2009
2013
2017
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.5
0.0
−0.5
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.5
0.0
−0.5
−0.2
0.0
0.2
Theoretical quantiles


NAVB Histogram of returns
$\mu = -0.0003796553649754 85$
$\sigma = 0.06408260308073295$
probability
6
5
4
3
2
1
0
−0.2
−0.1
0.0
0.1
0.2
returns


HRG Time Series Analysis Plots
0.5
0.0
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles

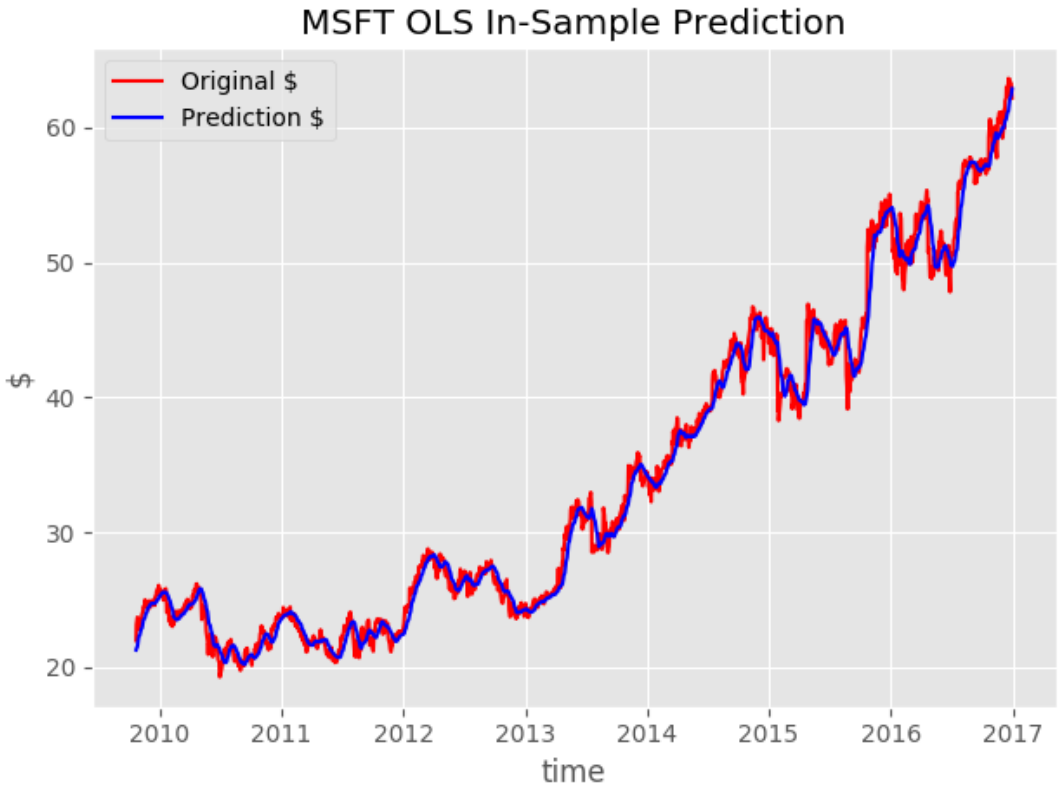

HRG Histogram of returns
$\mu = -1.8961464922281334e-05$
$\sigma = 0.04075180410107697$
probability
returns
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

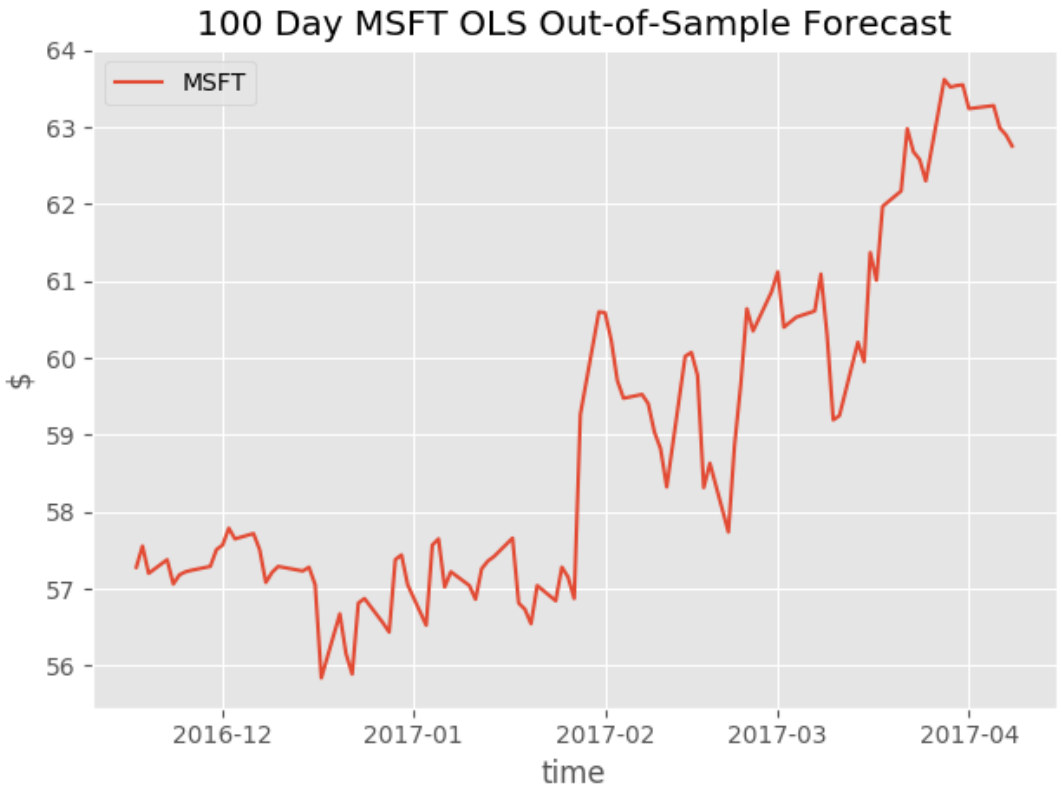

HL Time Series Analysis Plots
0.25
0.00
−0.25
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles


HL Histogram of returns
$\mu = -0.00014074173858773467$
$\sigma = 0.04223441033334755$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

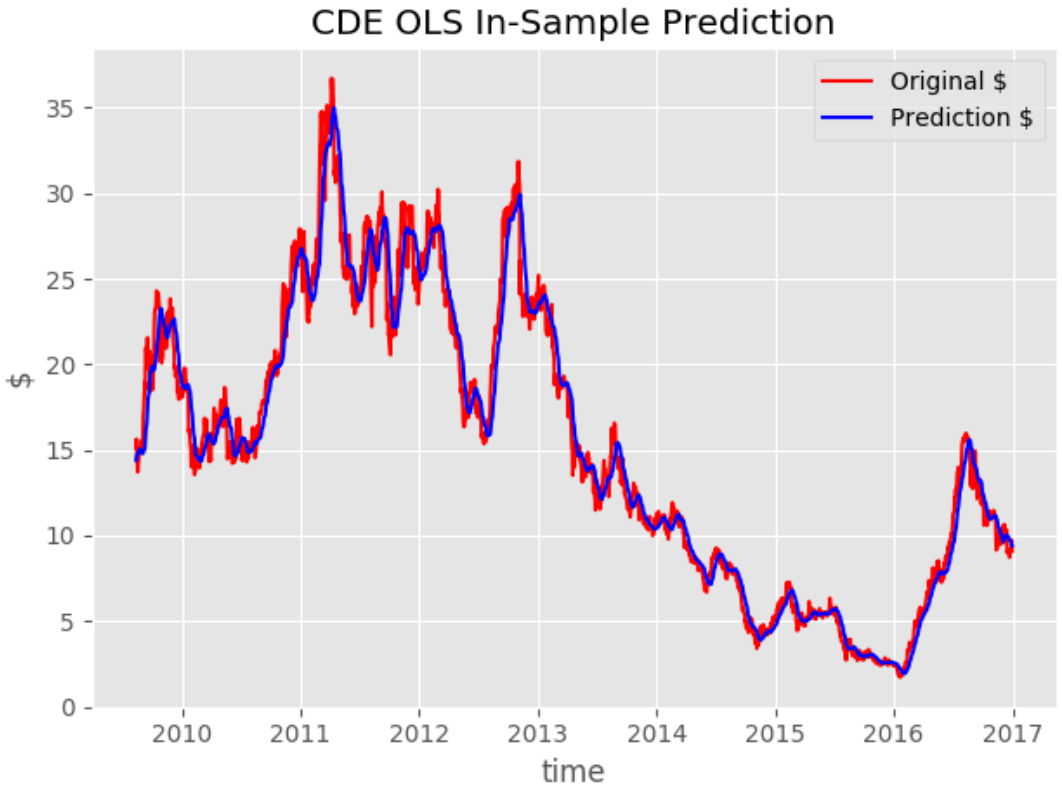
4.1.2 Ordinary Least Squares (OLS)

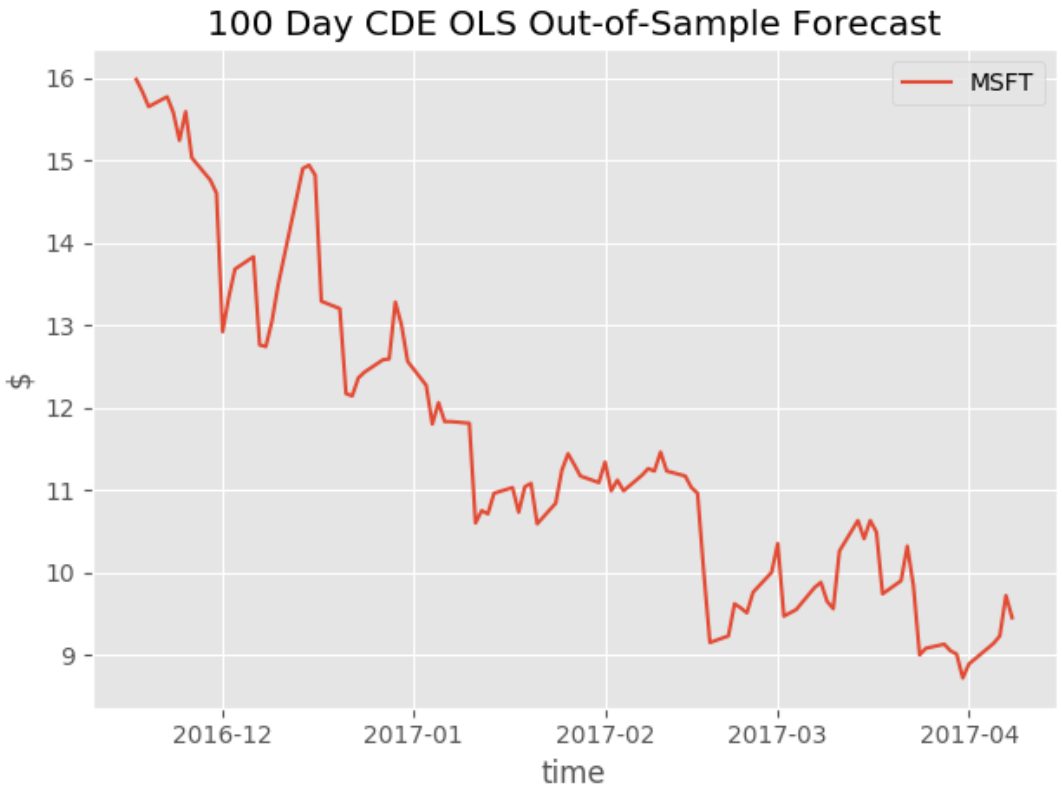
MSFT scored a mean absolute error regression loss of 0.810, and a coefficient of determination of 0.991.



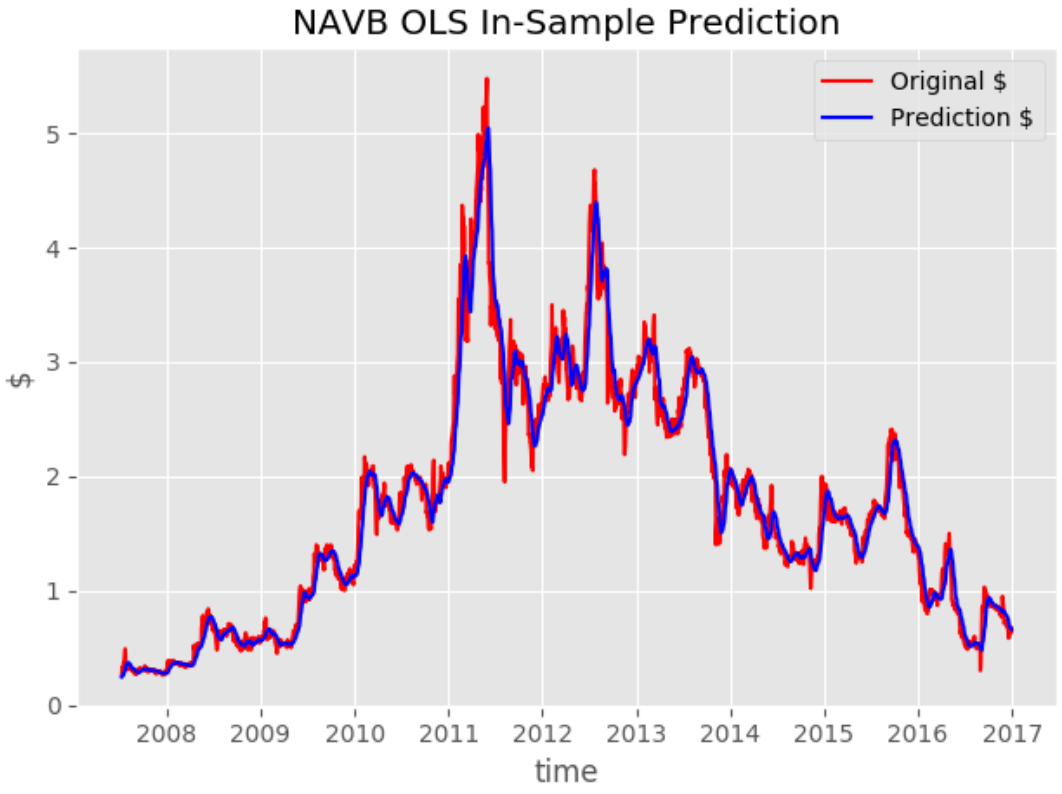


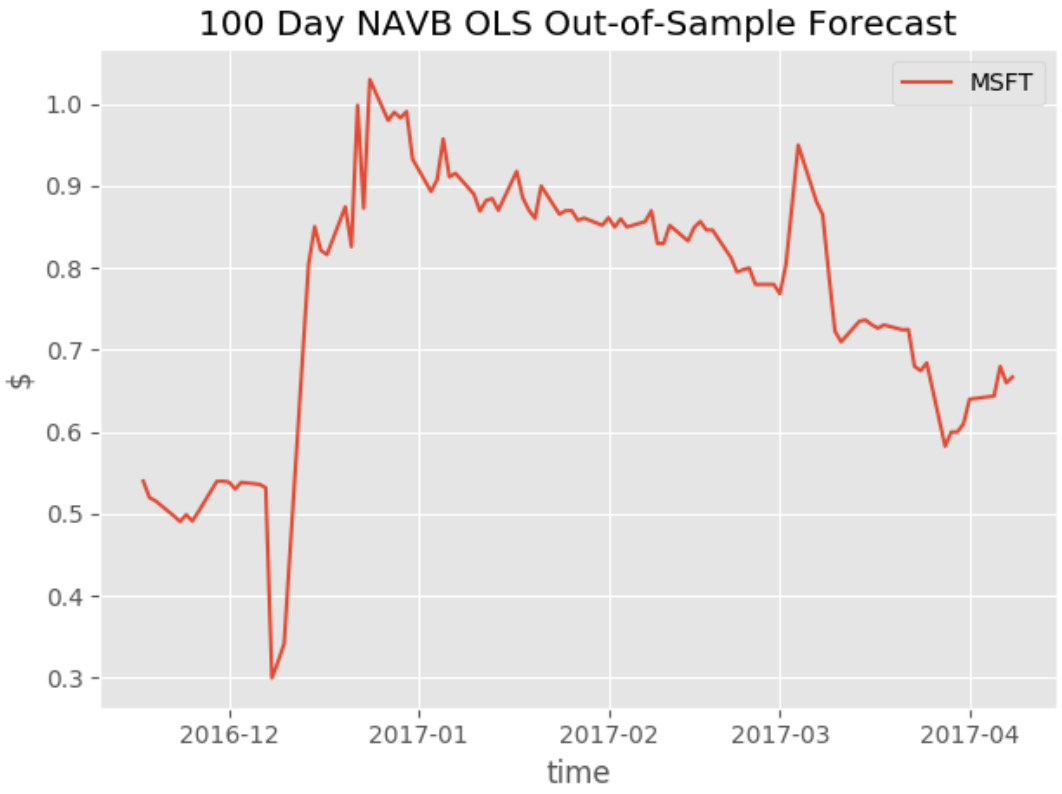
CDE scored a mean absolute error regression loss of 0.985, and a coefficient of determination of 0.973.



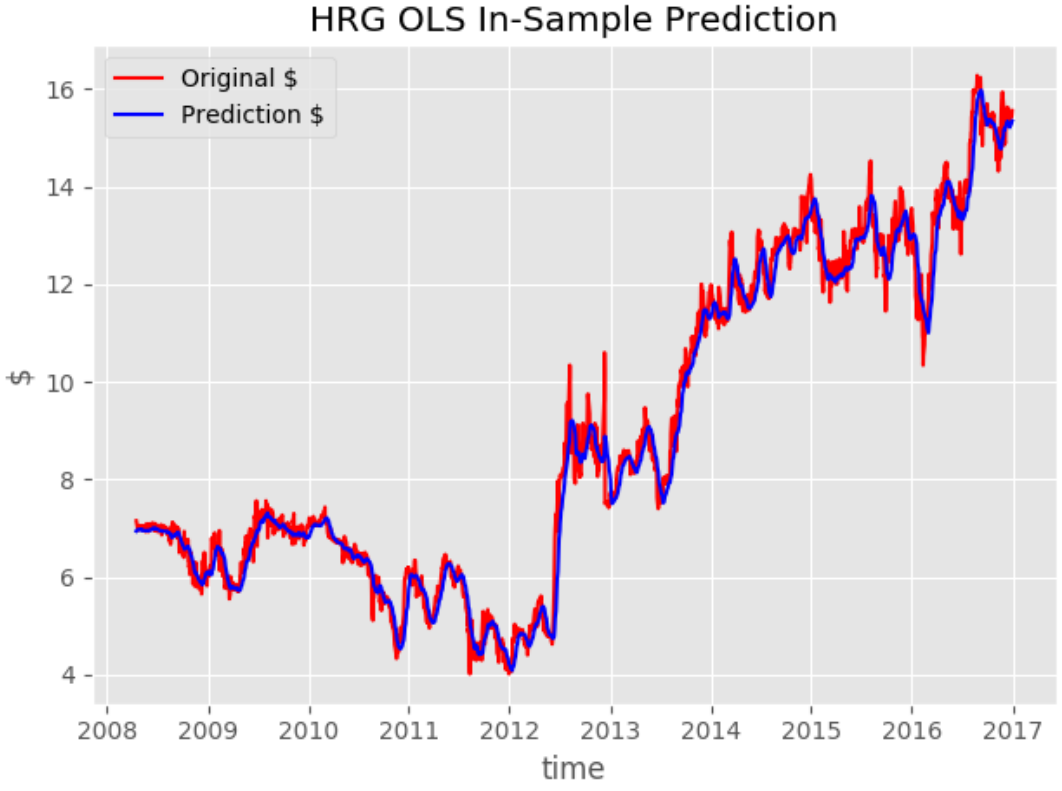


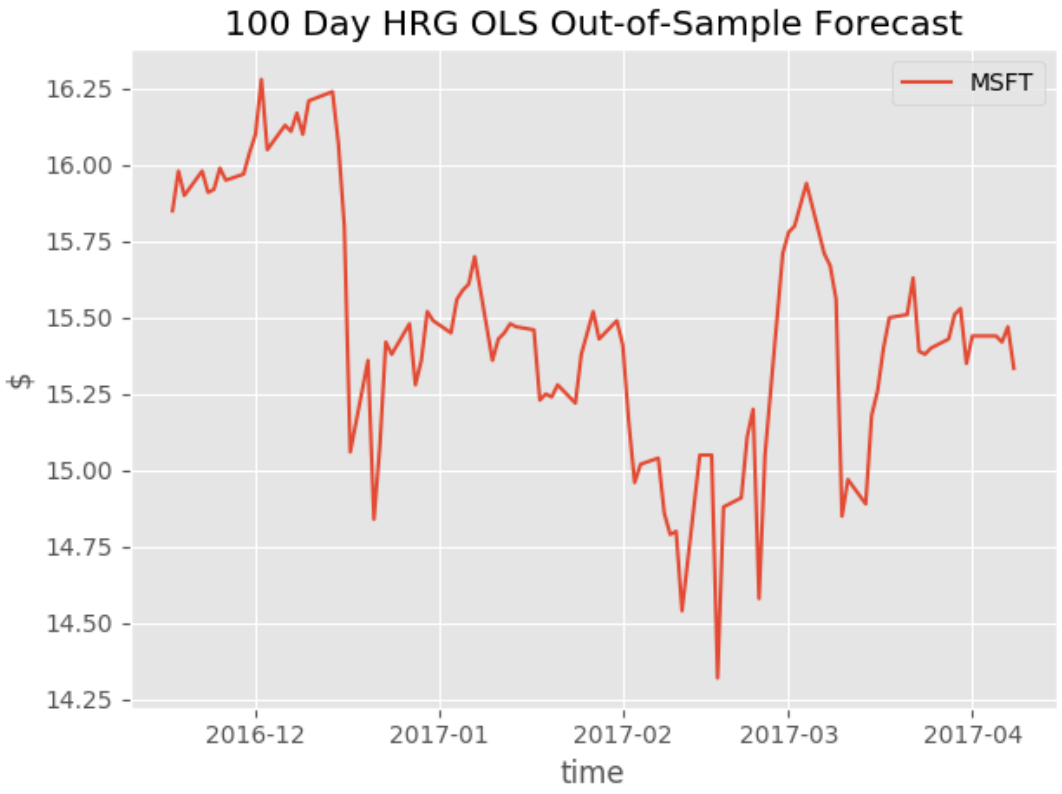
NAVB scored a mean absolute error regression loss of 0.124, and a coefficient of determination of 0.966.



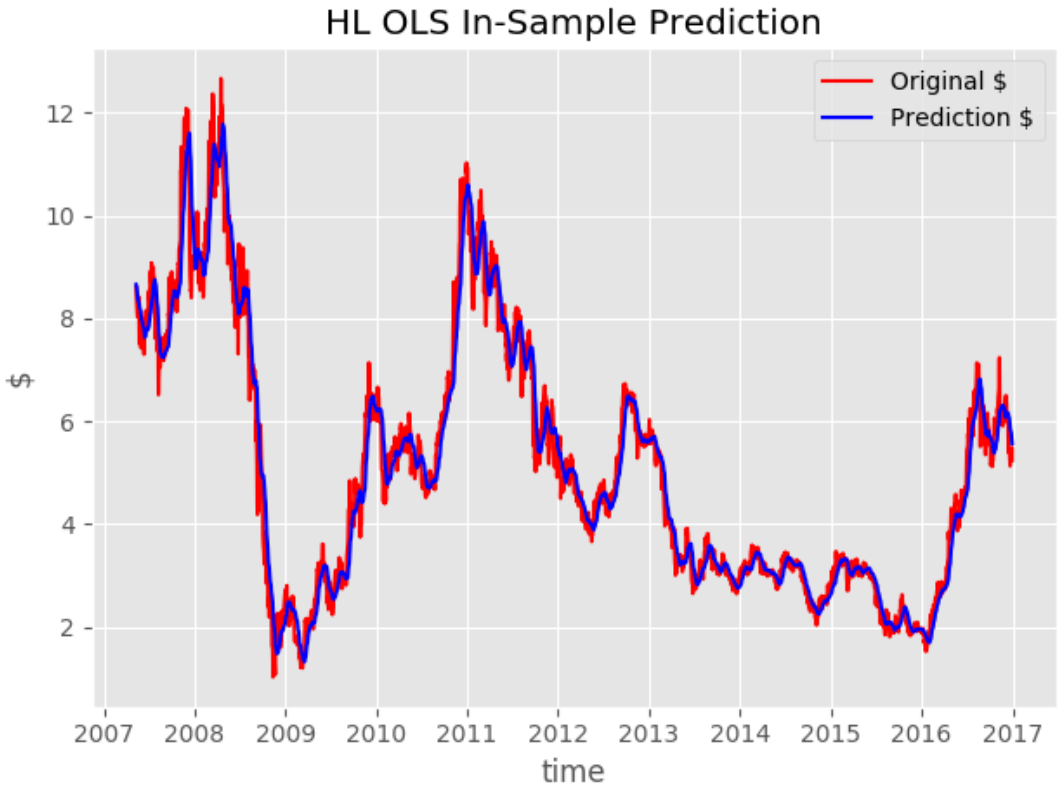


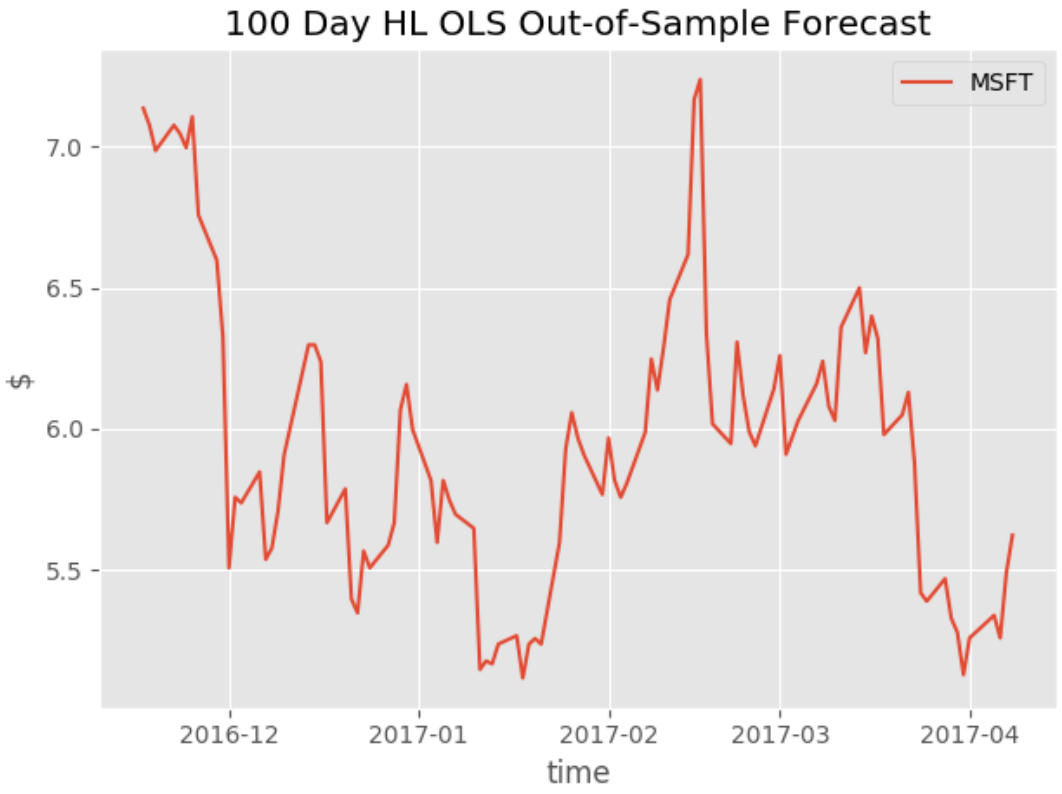
HRG scored a mean absolute error regression loss of 0.291, and a coefficient of determination of 0.986.



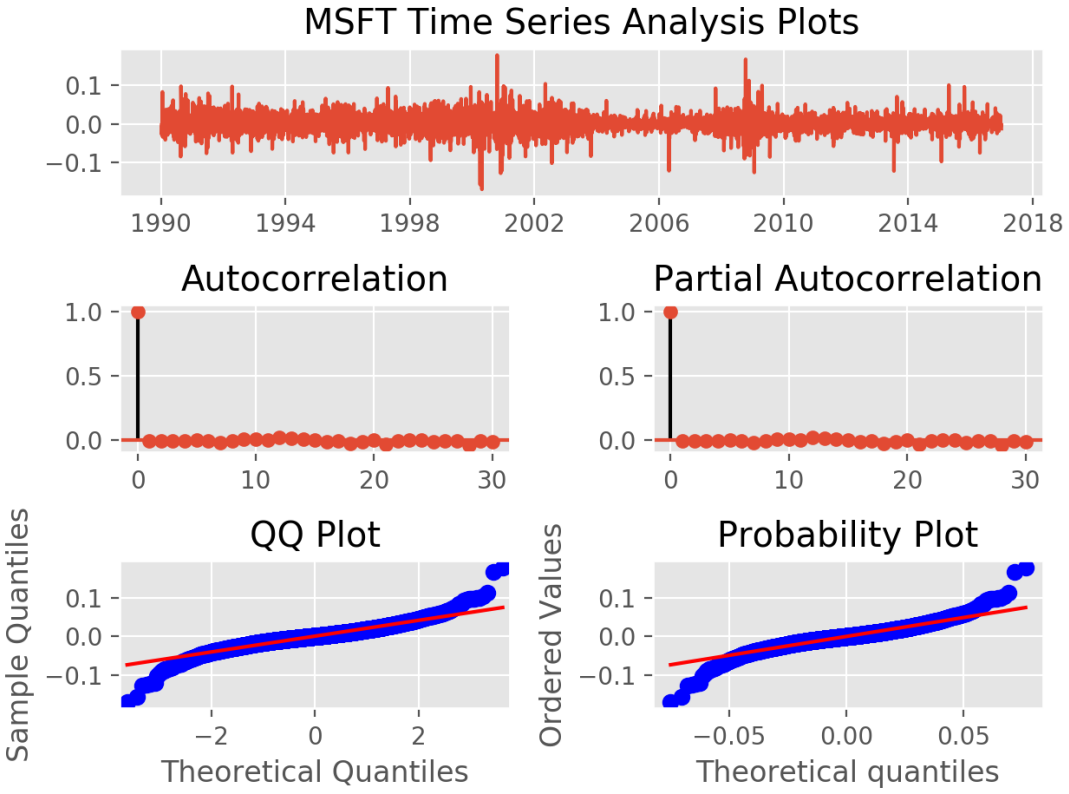


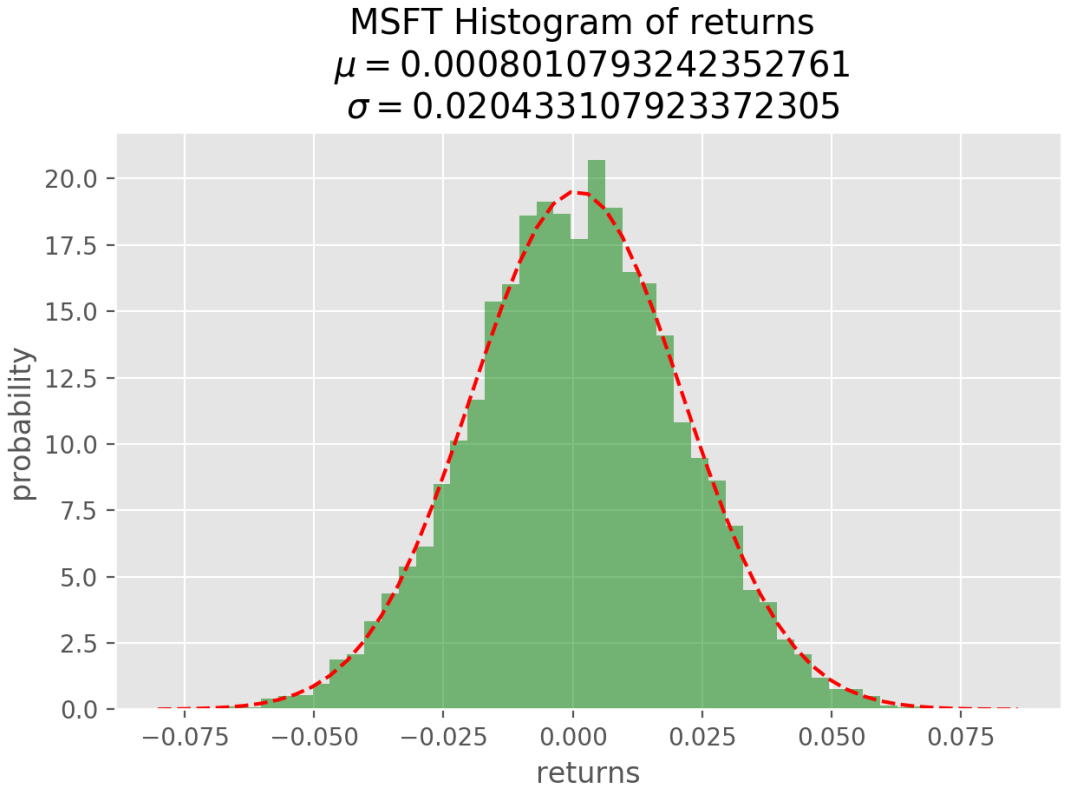
HL scored a mean absolute error regression loss of 0.336, and a coefficient of determination of 0.963.

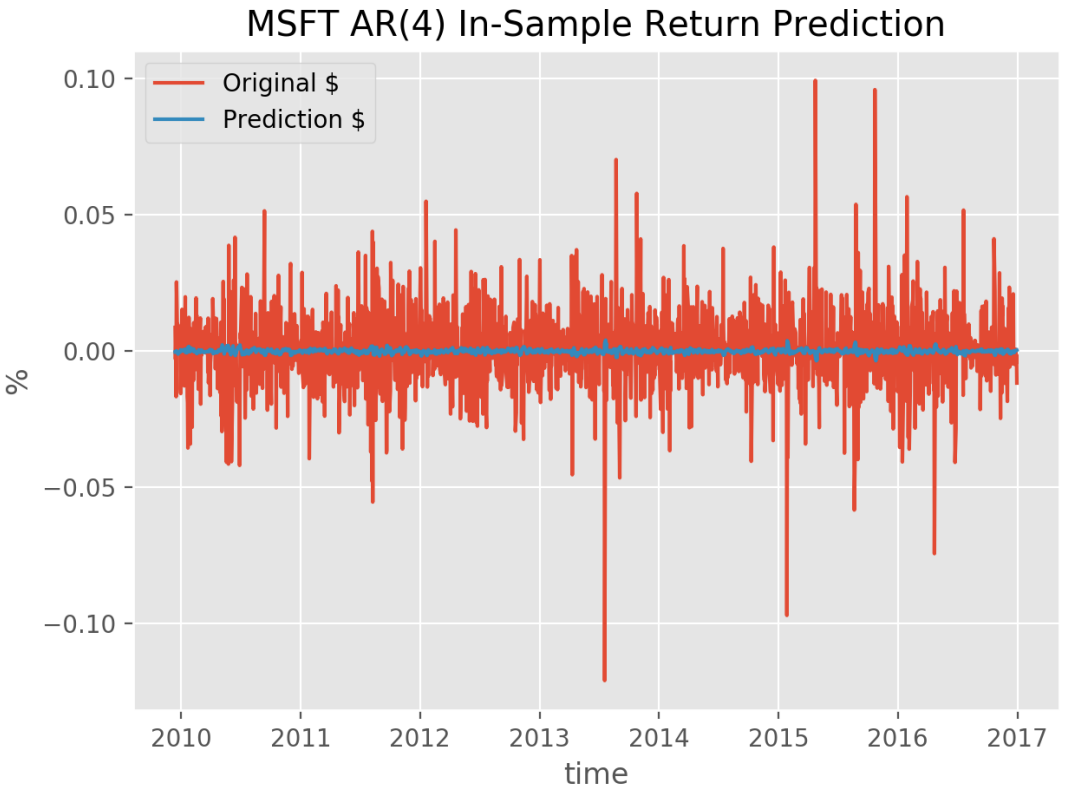


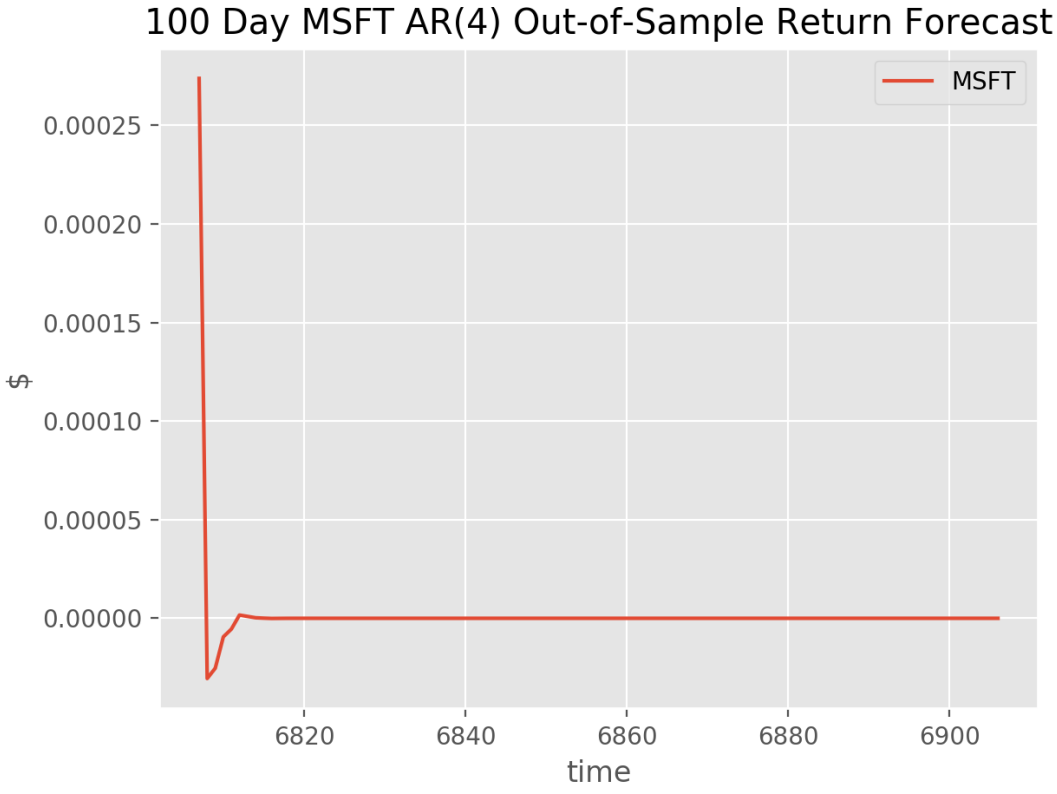

100 Day HL OLS Out-of-Sample Forecast
MSFT
7.0
6.5
6.0
5.5
$
2016-12
2017-01
2017-02
2017-03
2017-04
time

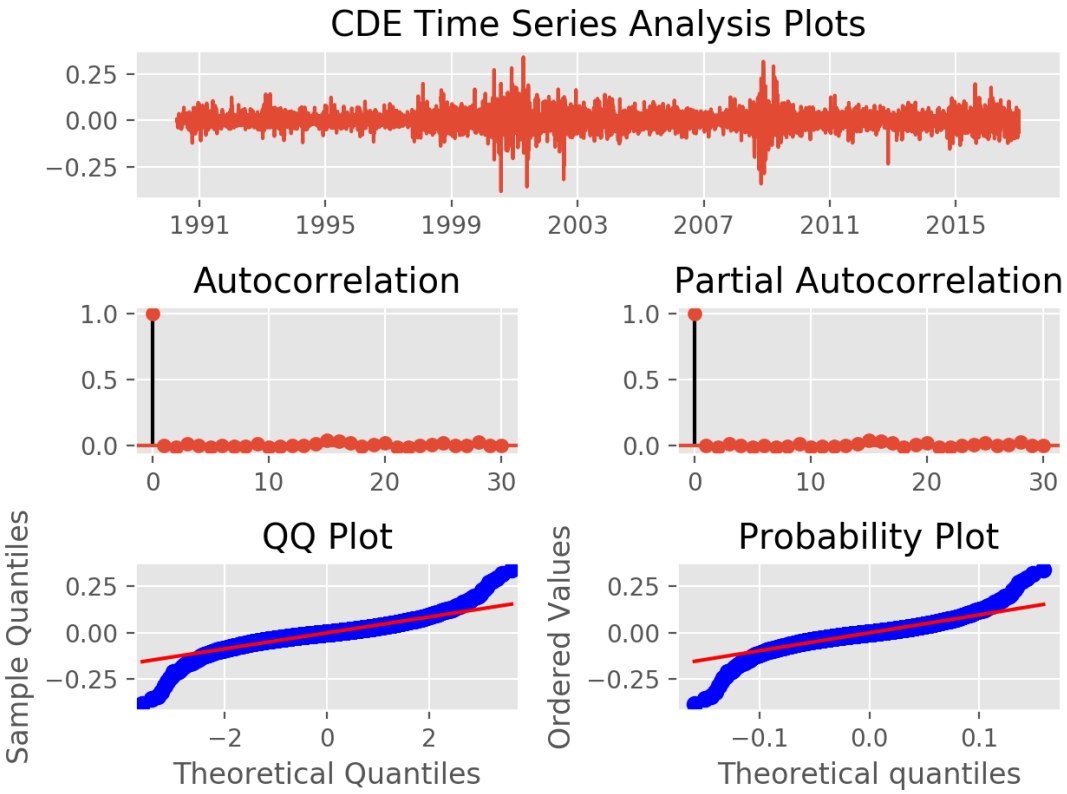
4.1.3 Auto Regressive (AR)

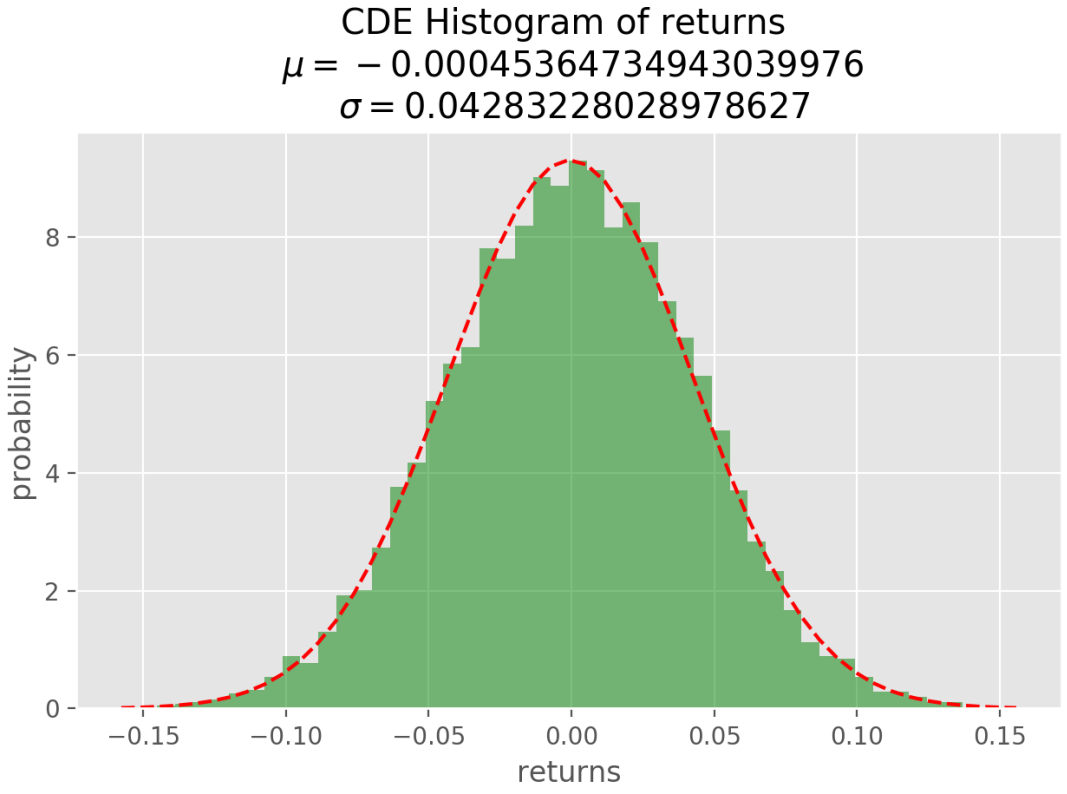


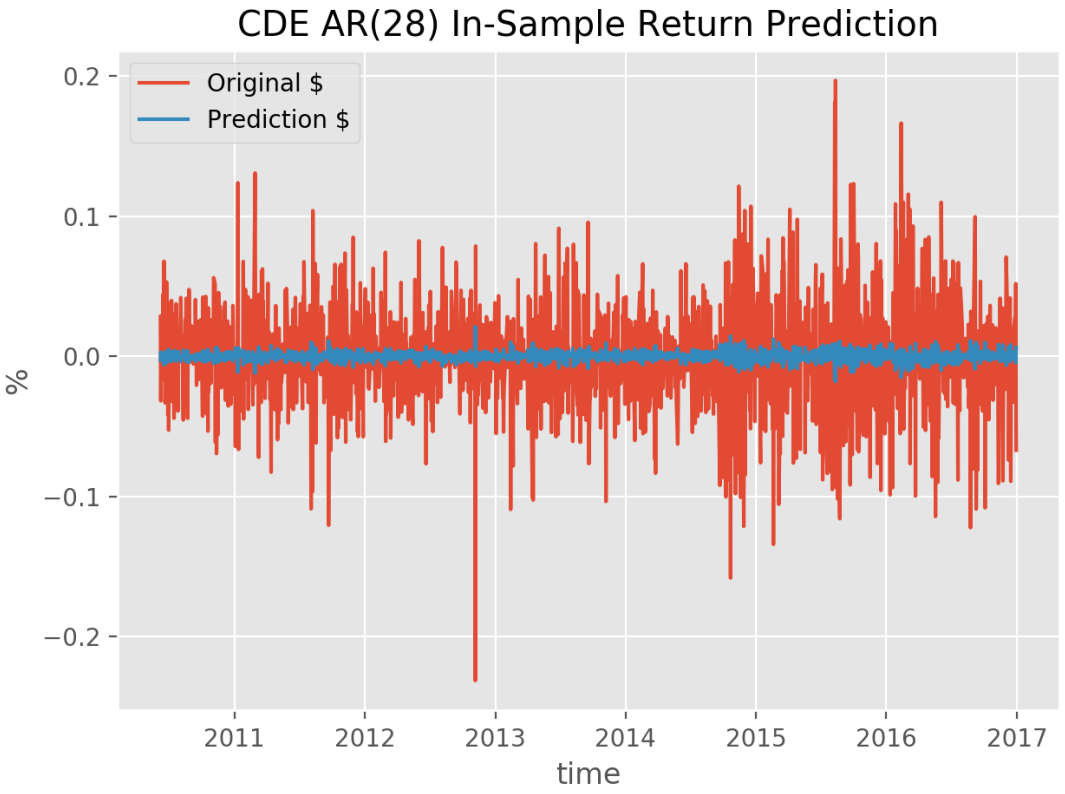


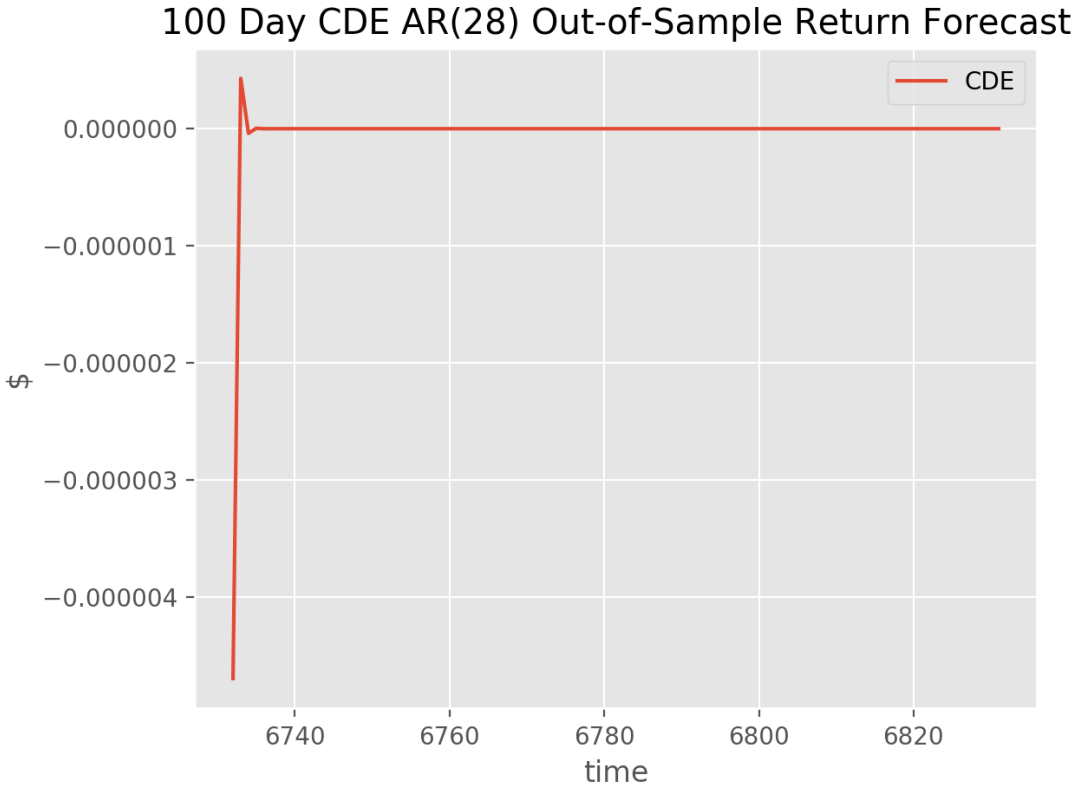

MSFT AR(4) In-Sample Return Prediction
Original $
Prediction $
0.10
0.05
0.00
−0.05
−0.10
%
2010
2011
2012
2013
2014
2015
2016
2017
time

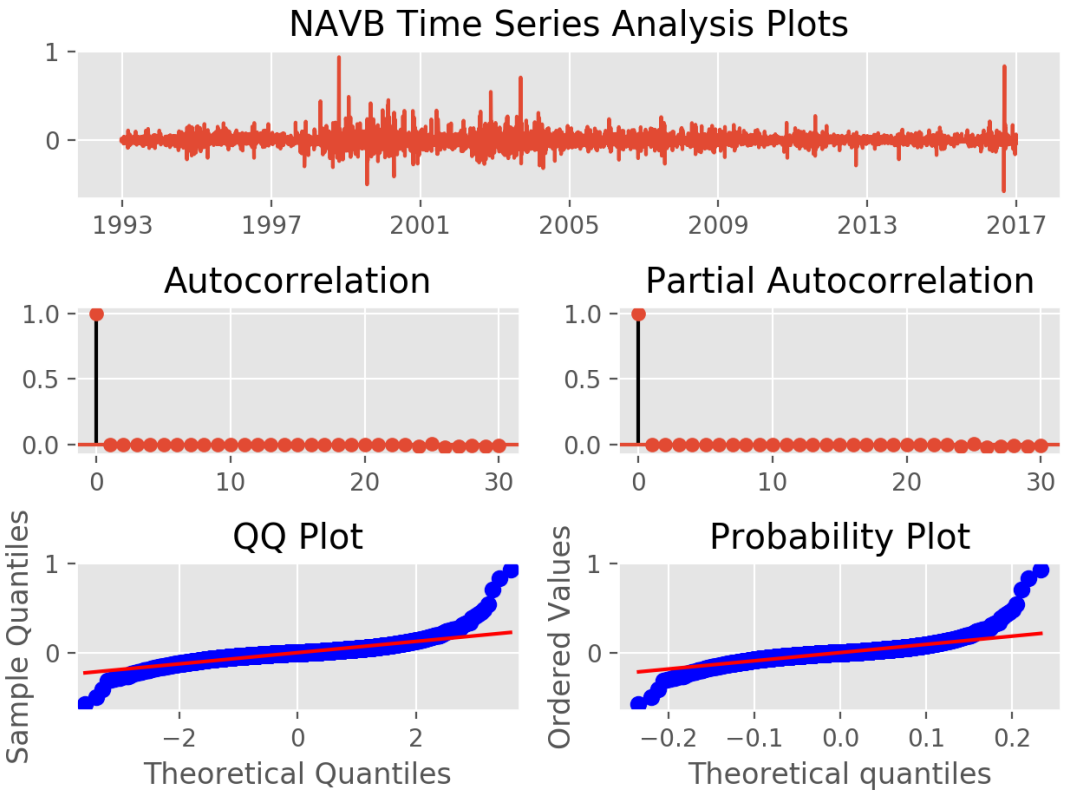

100 Day MSFT AR(4) Out-of-Sample Return Forecast
MSFT
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
$
6820
6840
6860
6880
6900
time

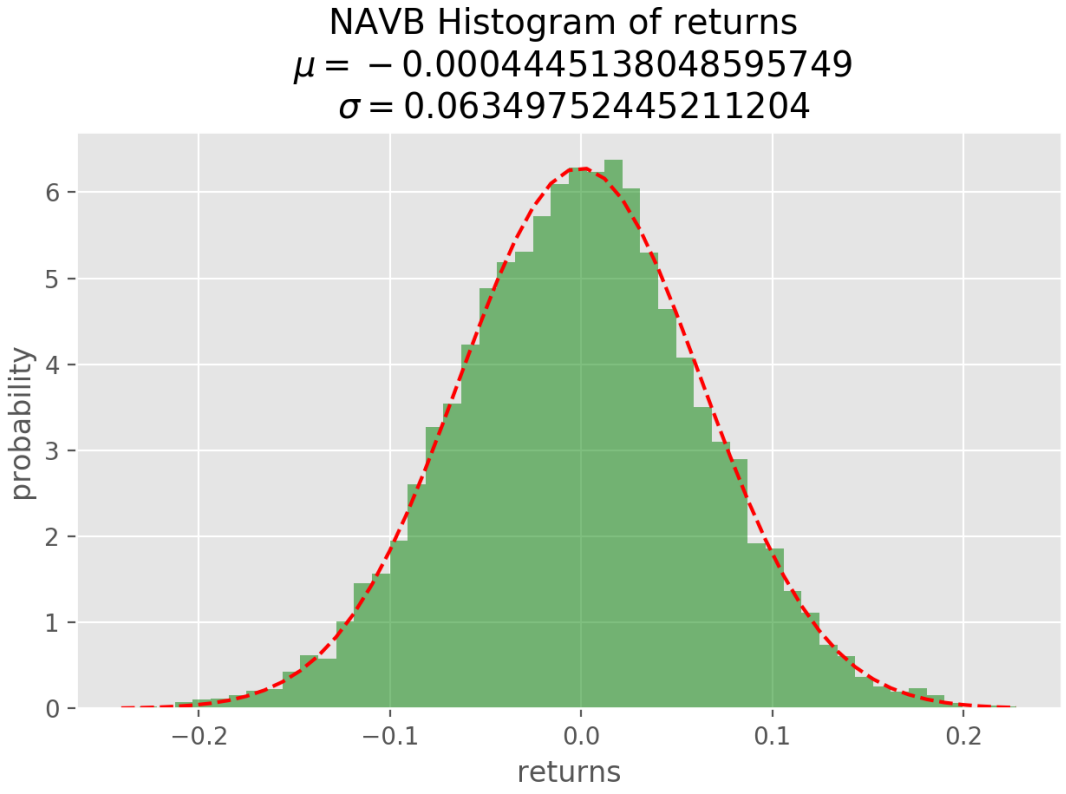

CDE Time Series Analysis Plots
0.25
0.00
−0.25
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles

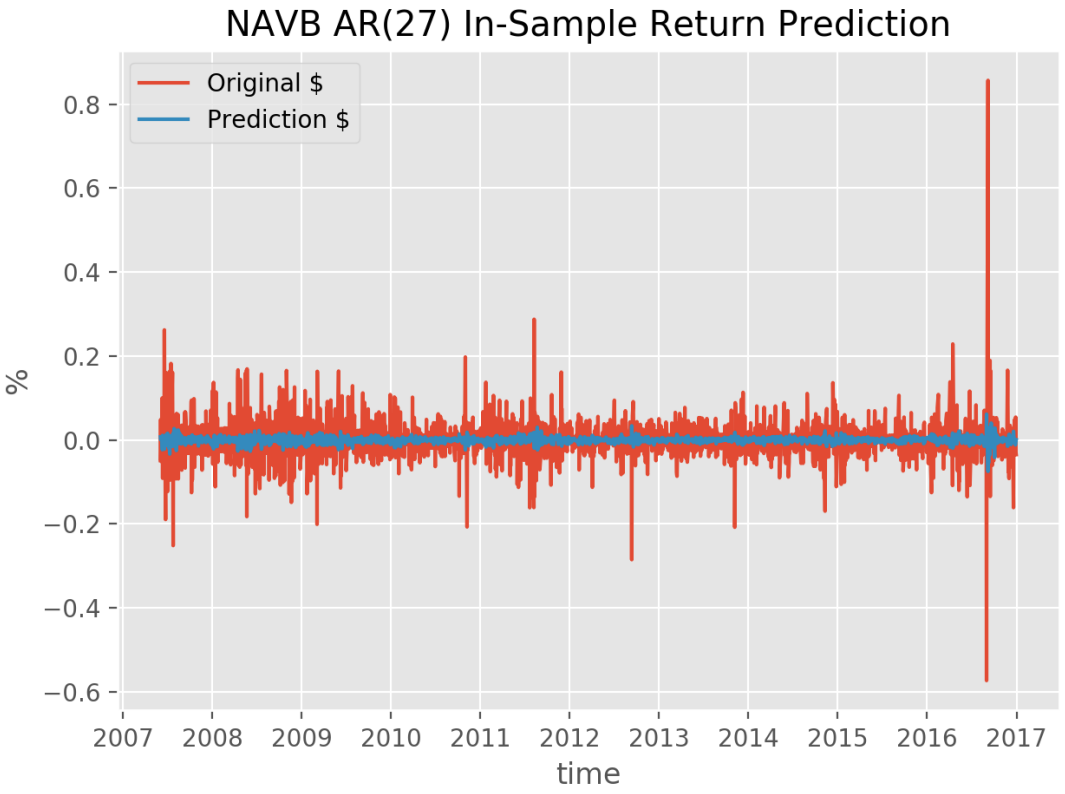

CDE Histogram of returns
$\mu = -0.0004536473494303997 6$
$\sigma = 0.04283228028978627$
probability
0
2
4
6
8
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

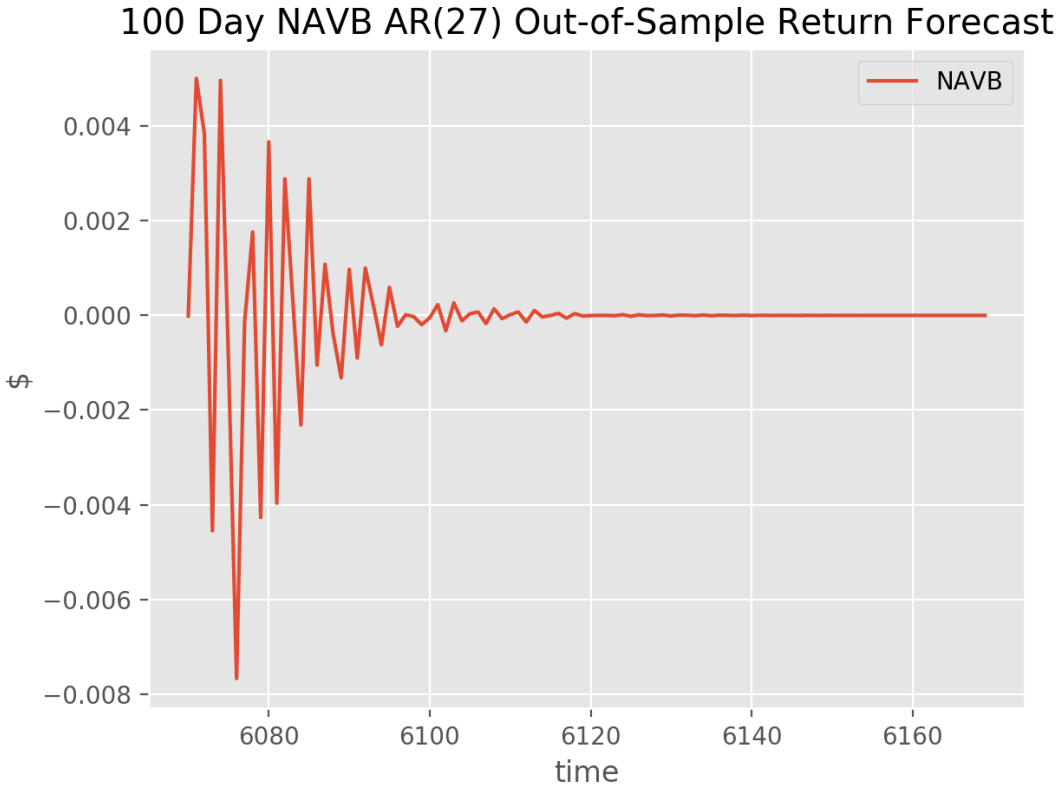

CDE AR(28) In-Sample Return Prediction
Original $
Prediction $
0.2
0.1
0.0
−0.1
−0.2
%
2011
2012
2013
2014
2015
2016
2017
time

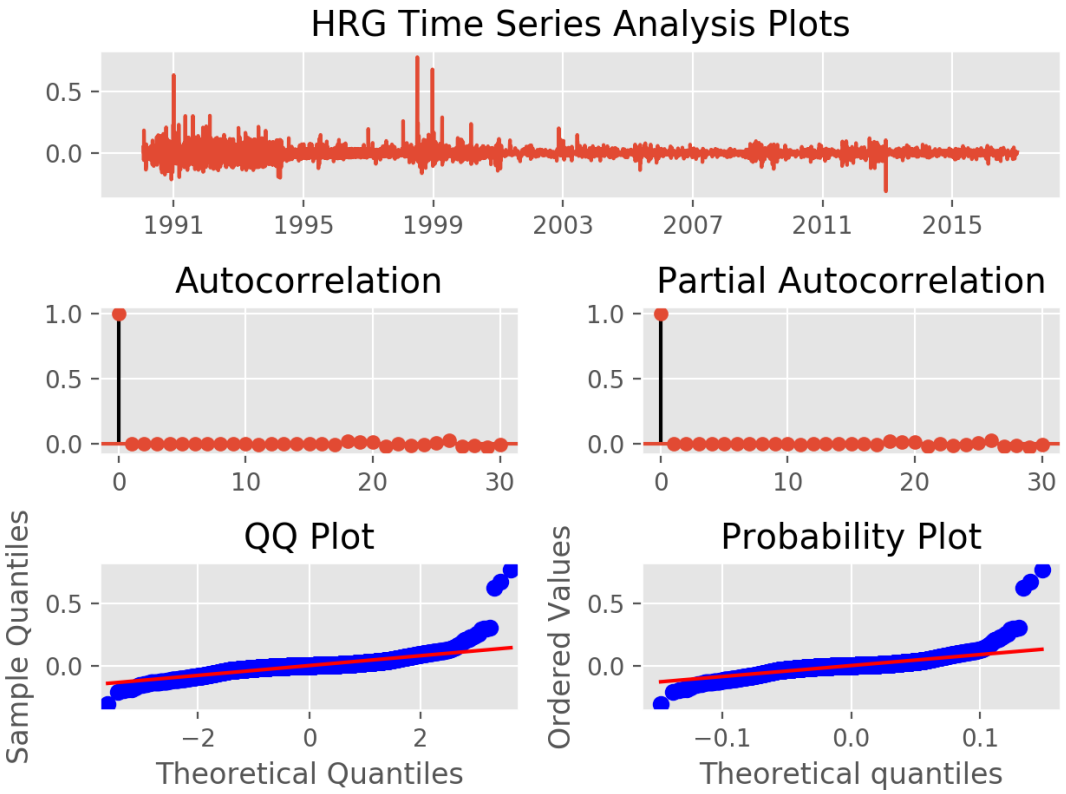

100 Day CDE AR(28) Out-of-Sample Return Forecast
CDE
0.000000
−0.000001
−0.000002
−0.000003
−0.000004
$
6740
6760
6780
6800
6820
time

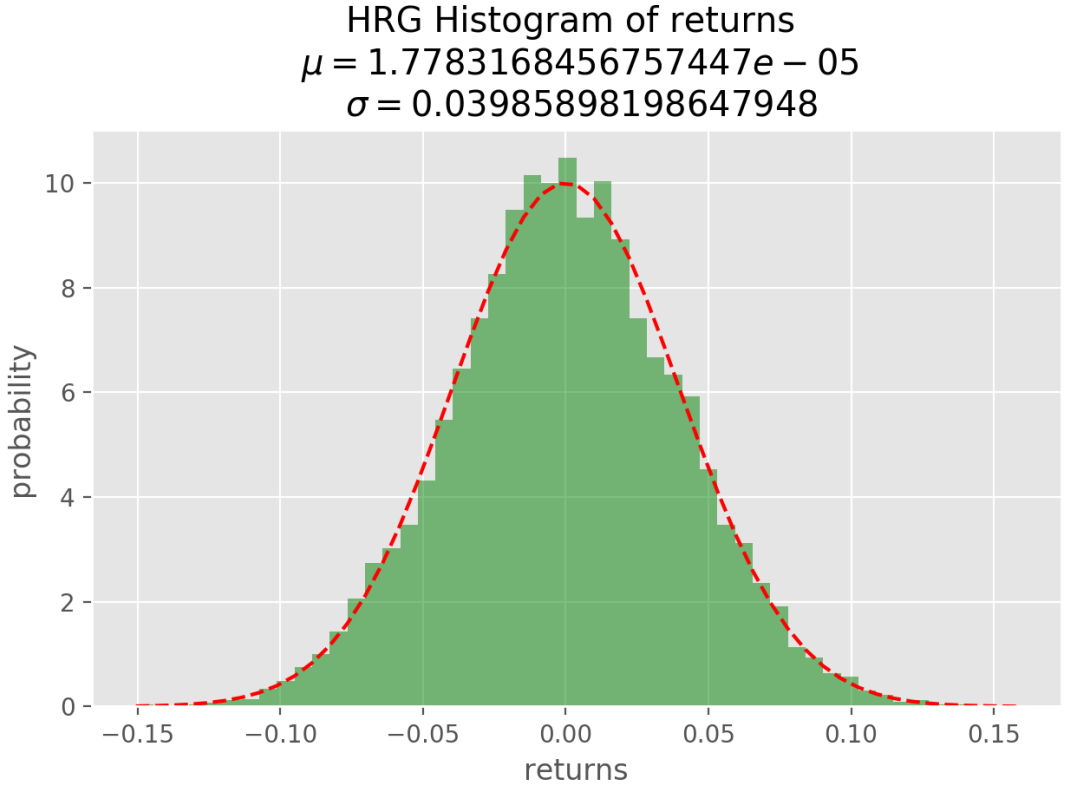

NAVB Time Series Analysis Plots
1
0
1993
1997
2001
2005
2009
2013
2017
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
1
0
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
1
0
−0.2
−0.1
0.0
0.1
0.2
Theoretical quantiles

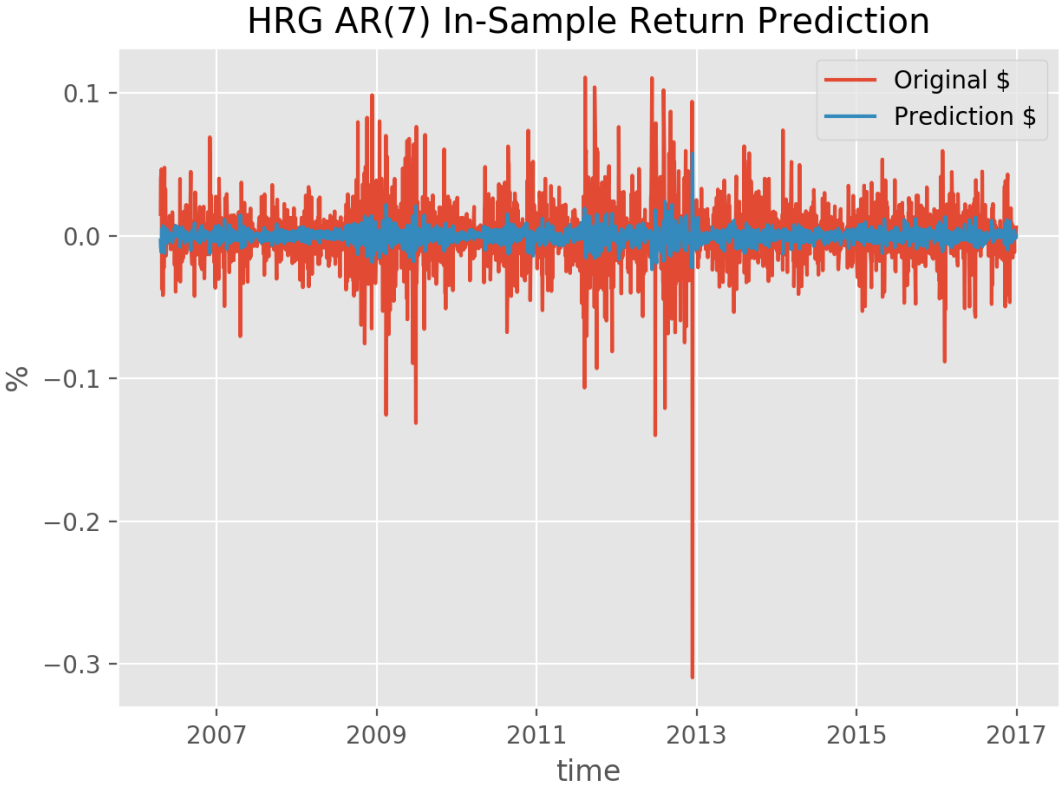

NAVB Histogram of returns
$\mu = -0.00044451380485957 49$
$\sigma = 0.06349752445211204$
probability
6
5
4
3
2
1
0
−0.2
−0.1
0.0
0.1
0.2
returns

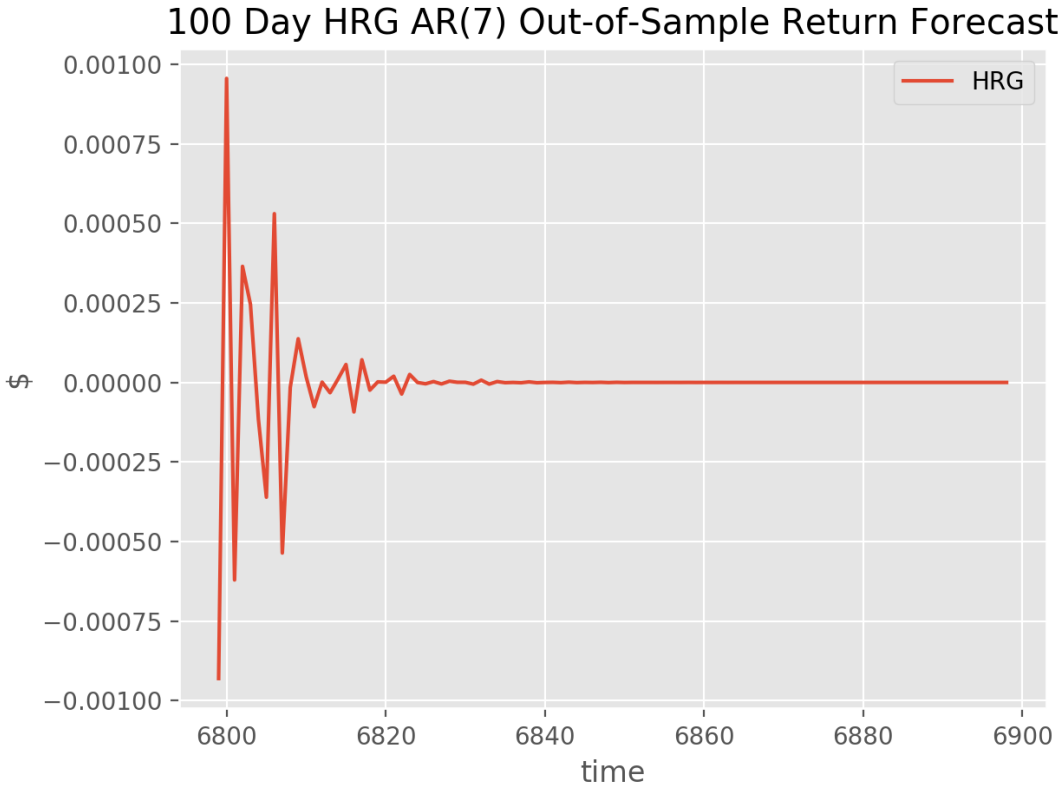

NAVB AR(27) In-Sample Return Prediction
Original $
Prediction $
%
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

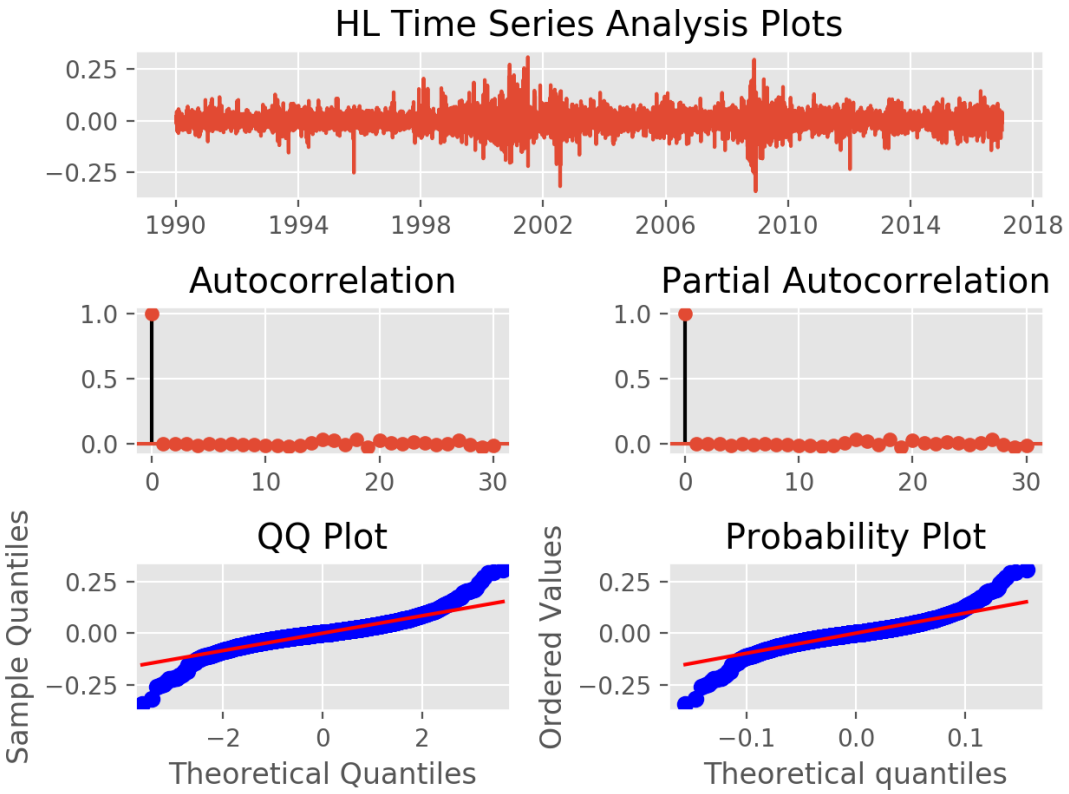

100 Day NAVB AR(27) Out-of-Sample Return Forecast
NAVB
$
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
6080
6100
6120
6140
6160
time

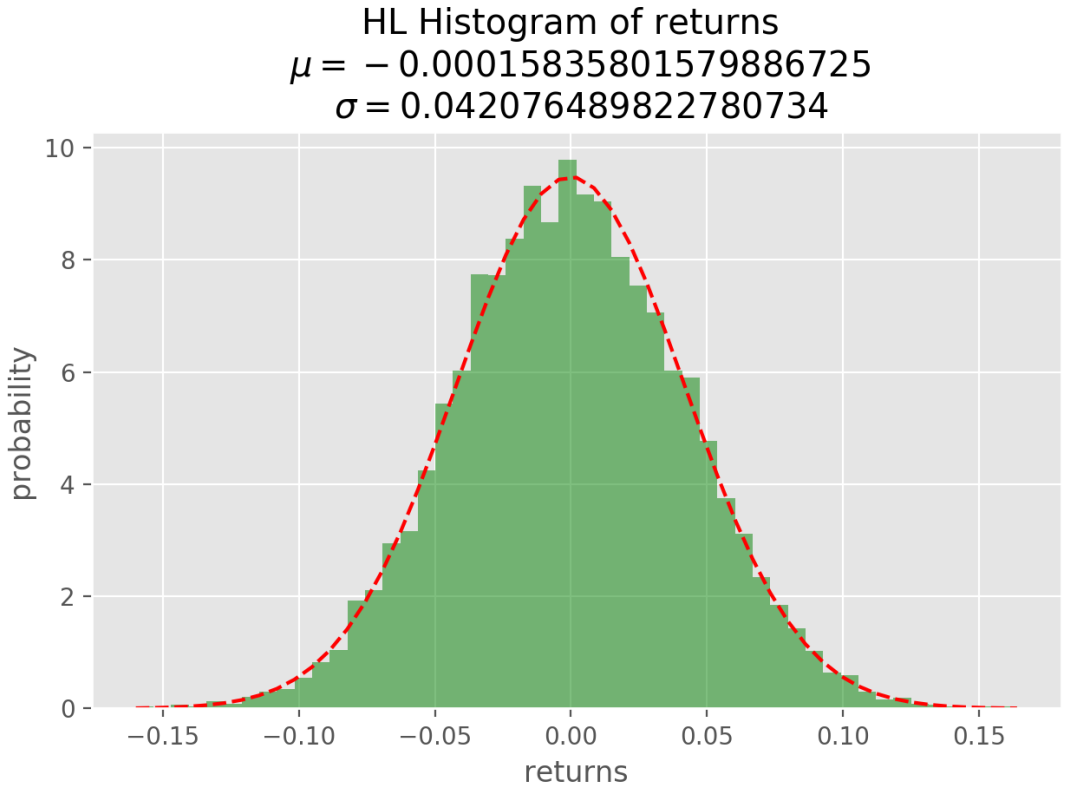

HRG Time Series Analysis Plots
0.5
0.0
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.5
0.0
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.5
0.0
−0.1
0.0
0.1
Theoretical quantiles

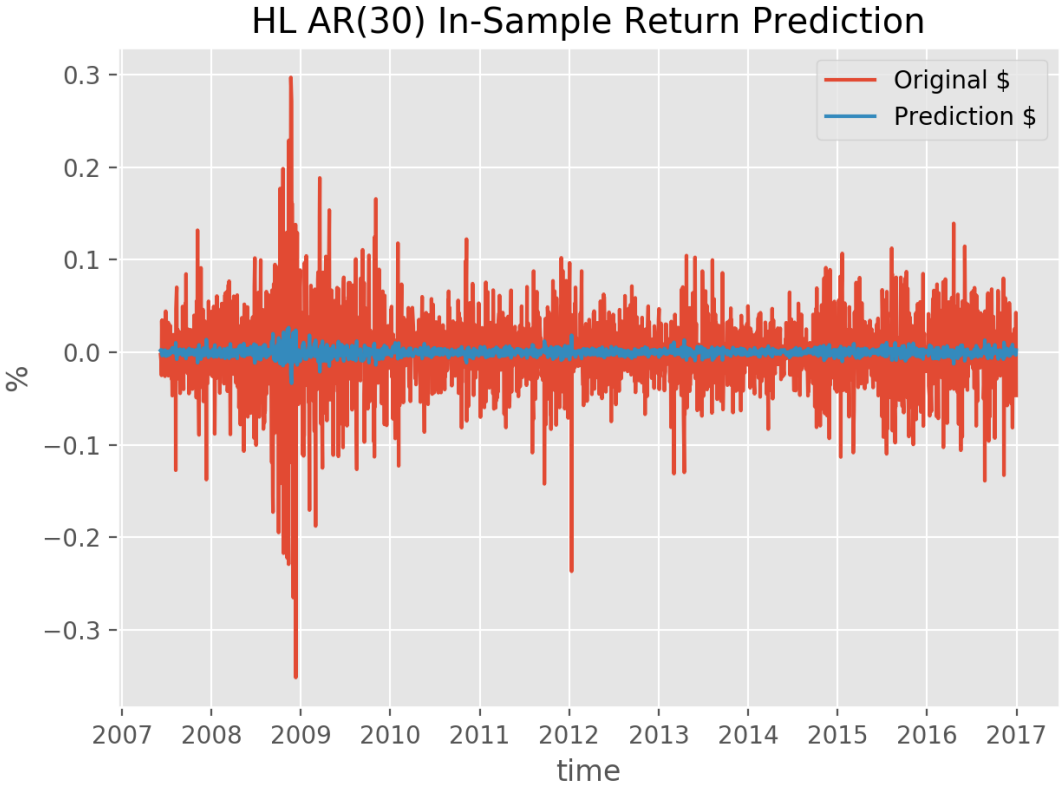

HRG Histogram of returns
$\mu = 1.7783168456757447e-05$
$\sigma = 0.03985898198647948$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

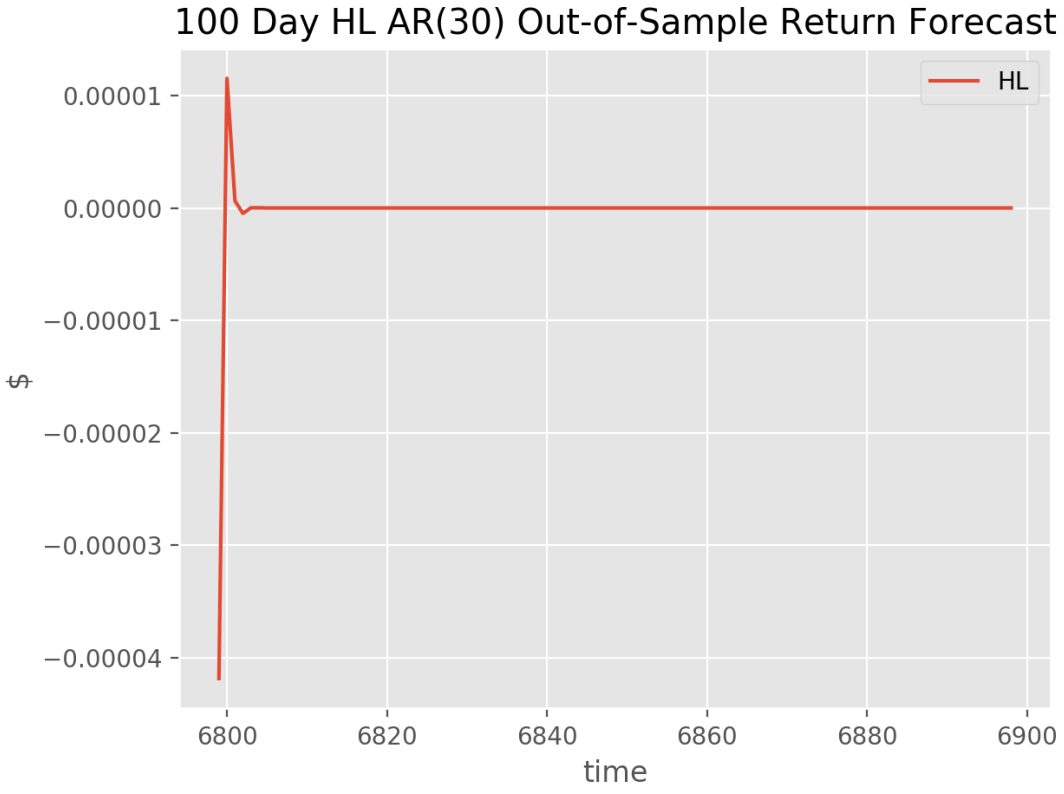

HRG AR(7) In-Sample Return Prediction
Original $
Prediction $
0.1
0.0
−0.1
−0.2
−0.3
%
2007
2009
2011
2013
2015
2017
time


100 Day HRG AR(7) Out-of-Sample Return Forecast
HRG
0.00100
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100
$
6800
6820
6840
6860
6880
6900
time

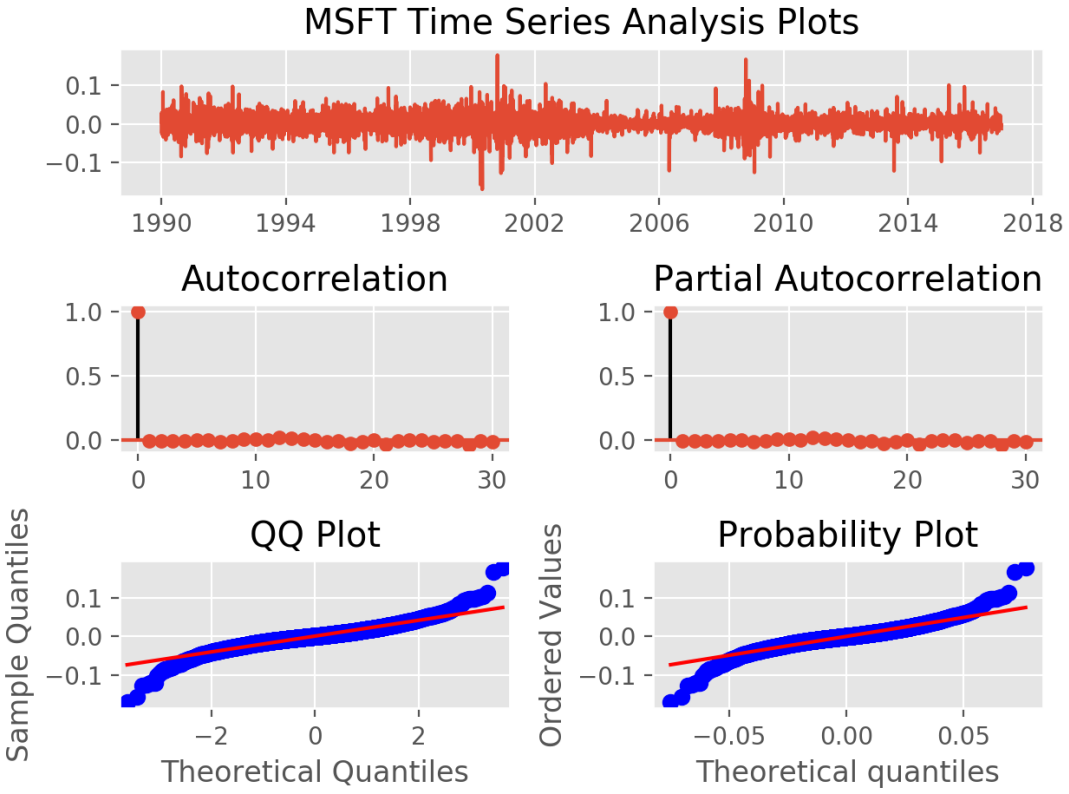

HL Time Series Analysis Plots
0.25
0.00
−0.25
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

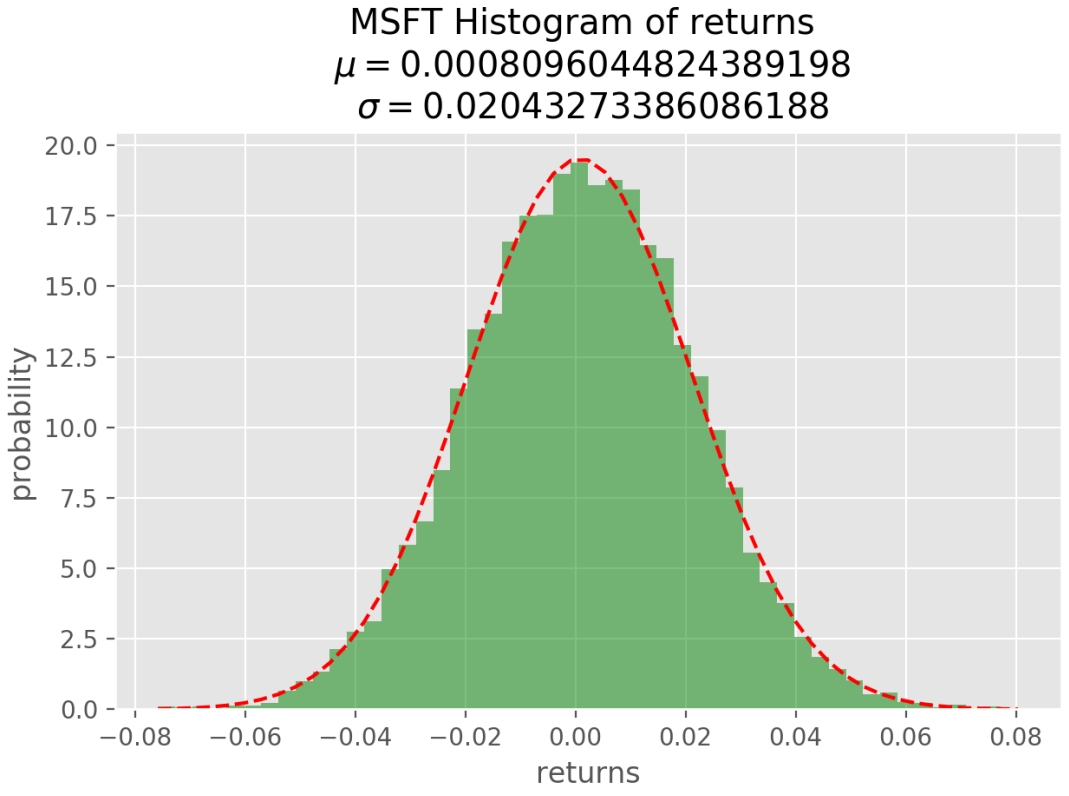

HL Histogram of returns
$\mu = -0.00015835801579886725$
$\sigma = 0.042076489822780734$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

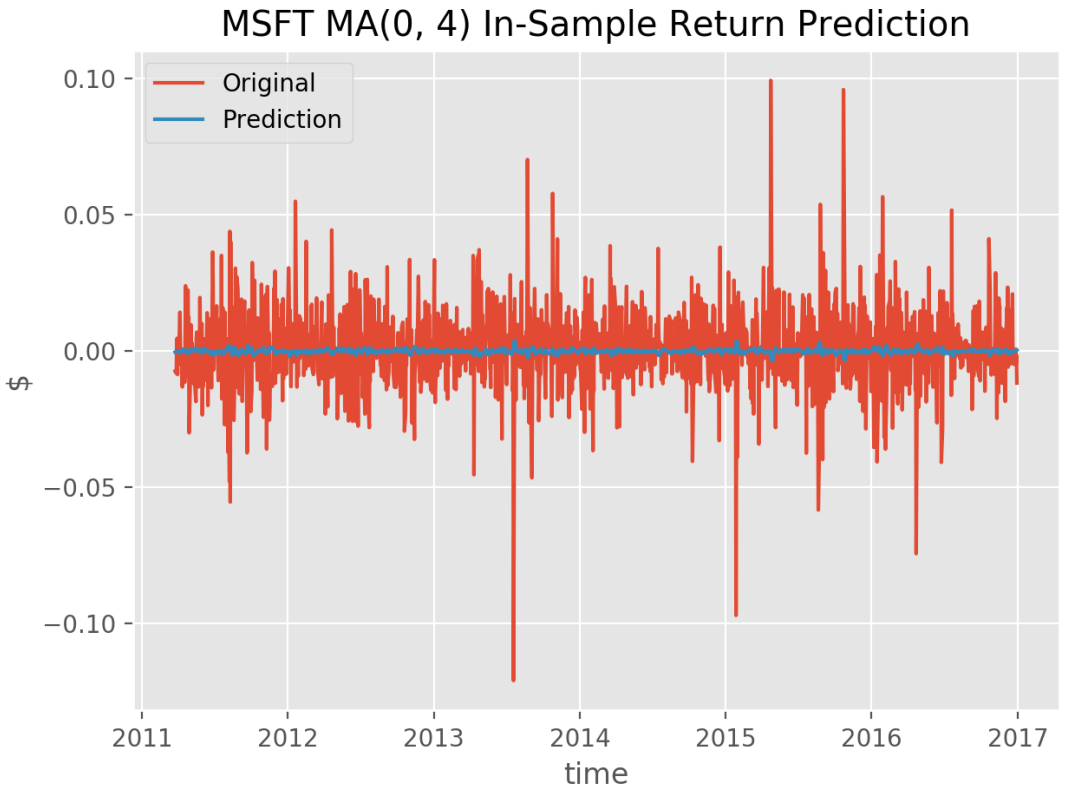

HL AR(30) In-Sample Return Prediction
Original $
Prediction $
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
%
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

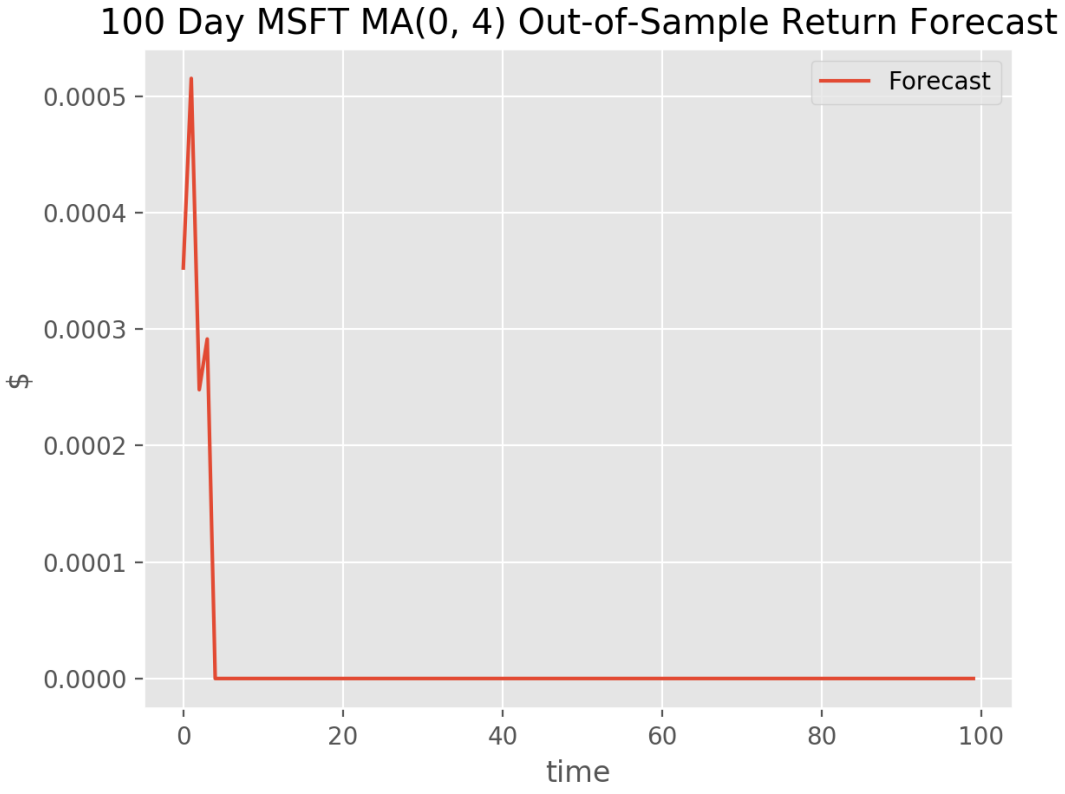

100 Day HL AR(30) Out-of-Sample Return Forecast
HL
0.00001
0.00000
−0.00001
−0.00002
−0.00003
−0.00004
$
6800
6820
6840
6860
6880
6900
time

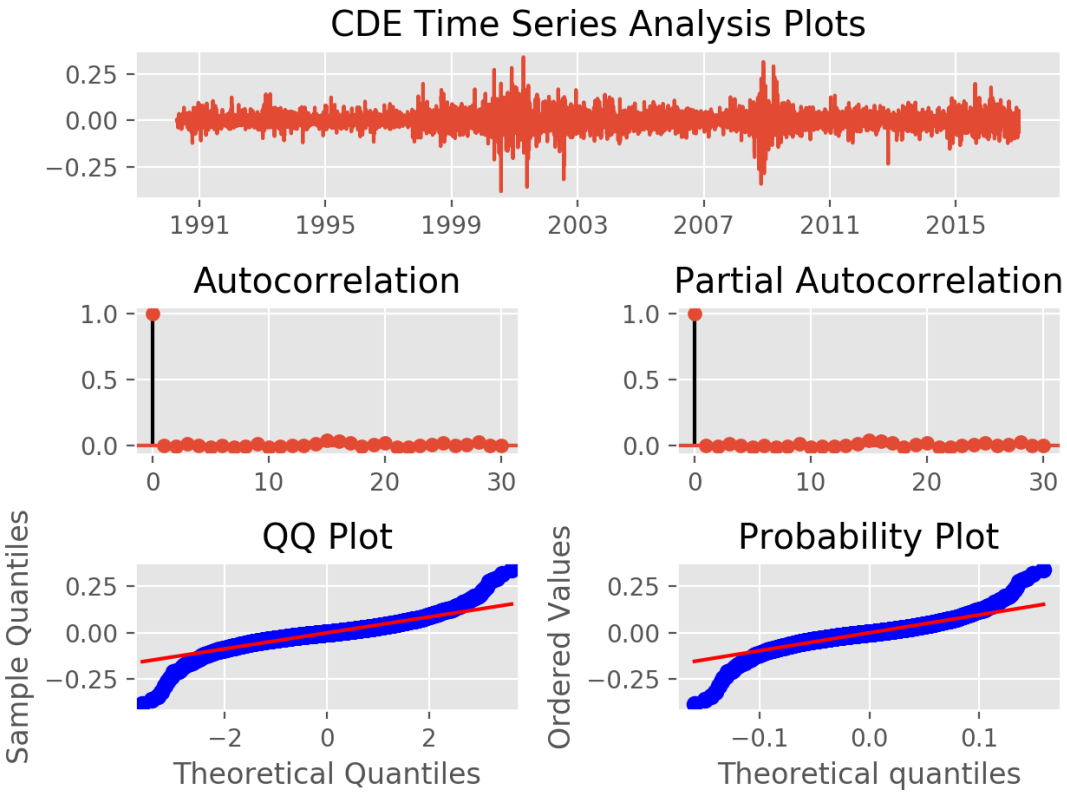
4.1.4 Moving Average (MA)

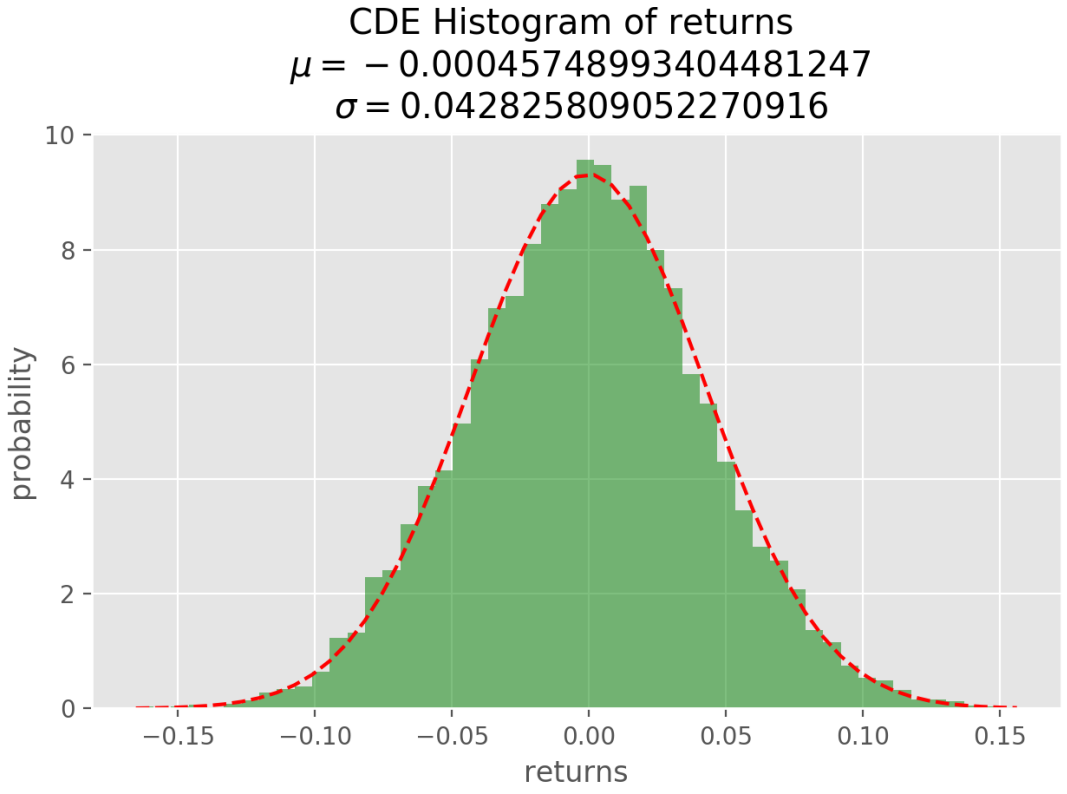


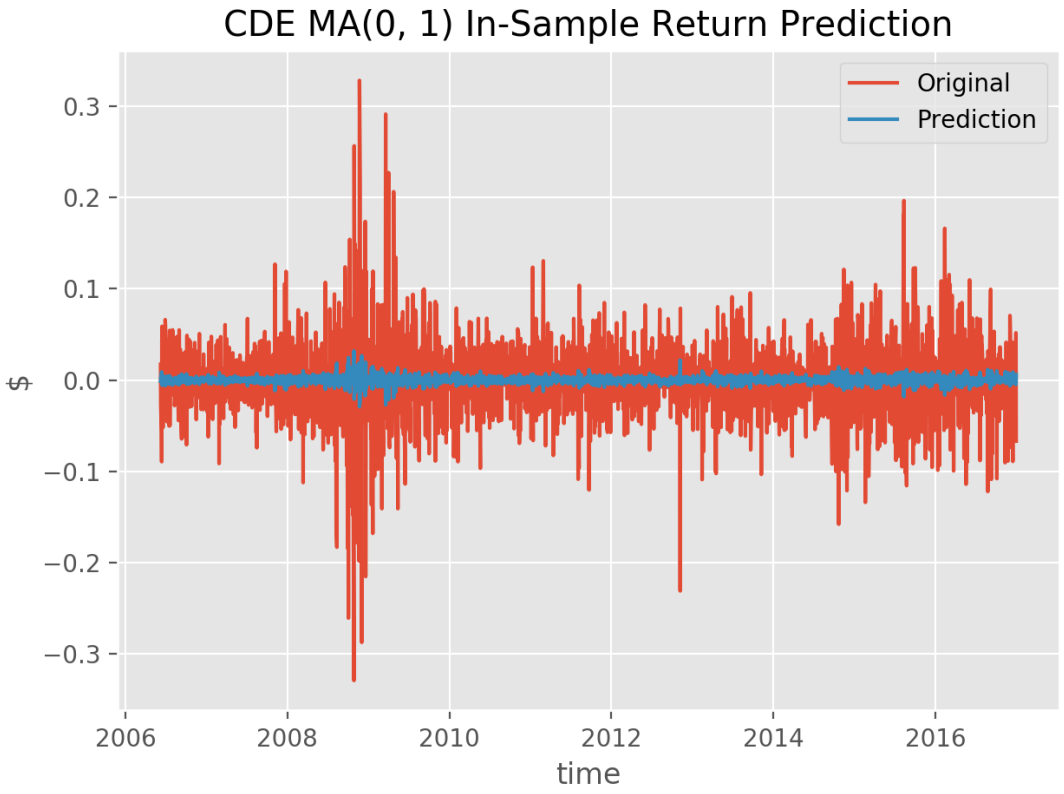


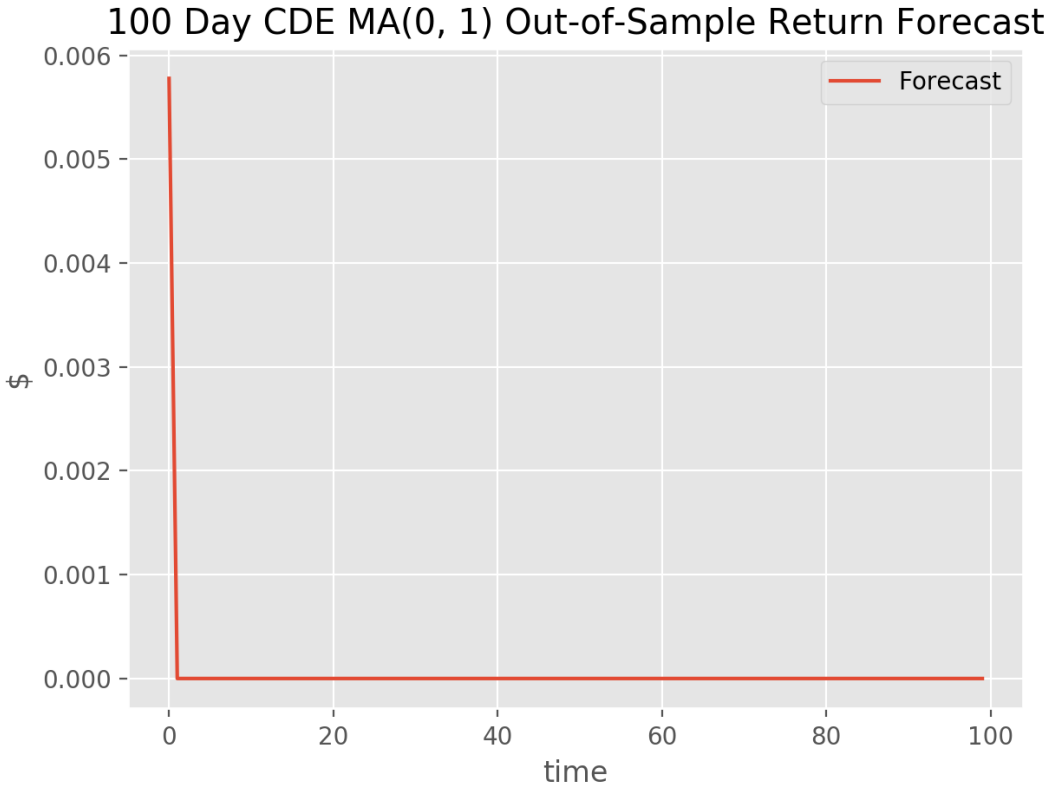

MSFT MA(0, 4) In-Sample Return Prediction
Original
Prediction
0.10
0.05
0.00
−0.05
−0.10
$
2011
2012
2013
2014
2015
2016
2017
time

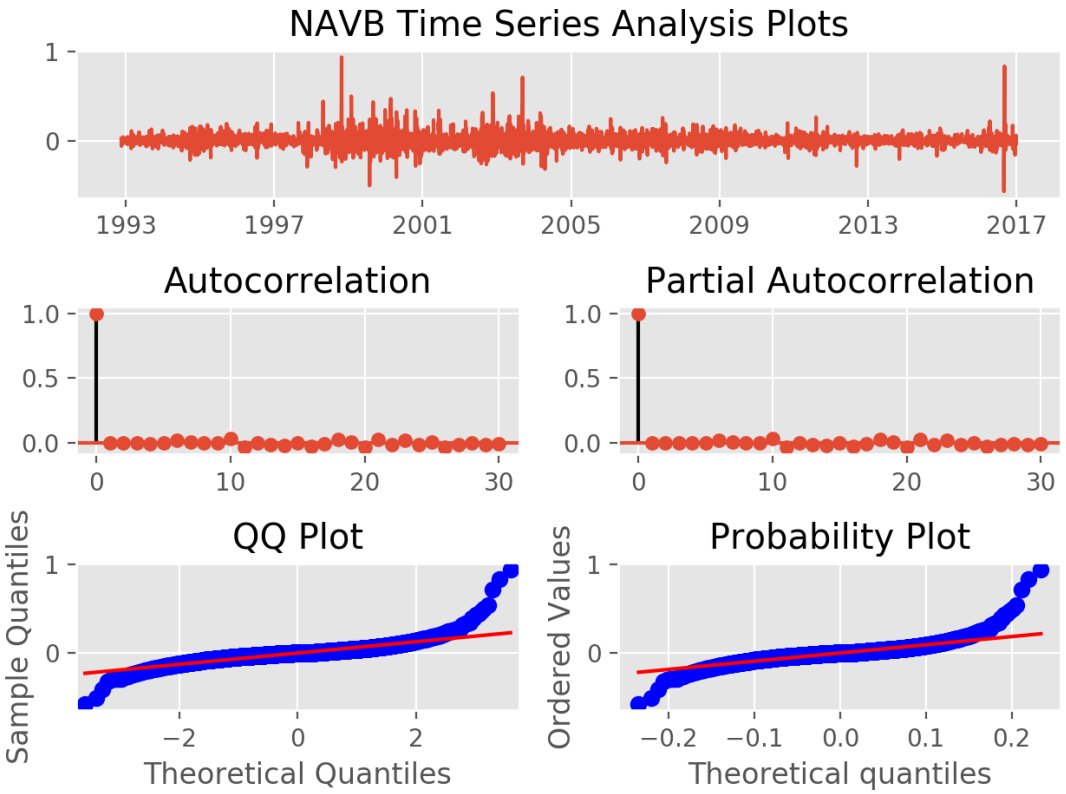

100 Day MSFT MA(0, 4) Out-of-Sample Return Forecast
Forecast
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
$
0
20
40
60
80
100
time

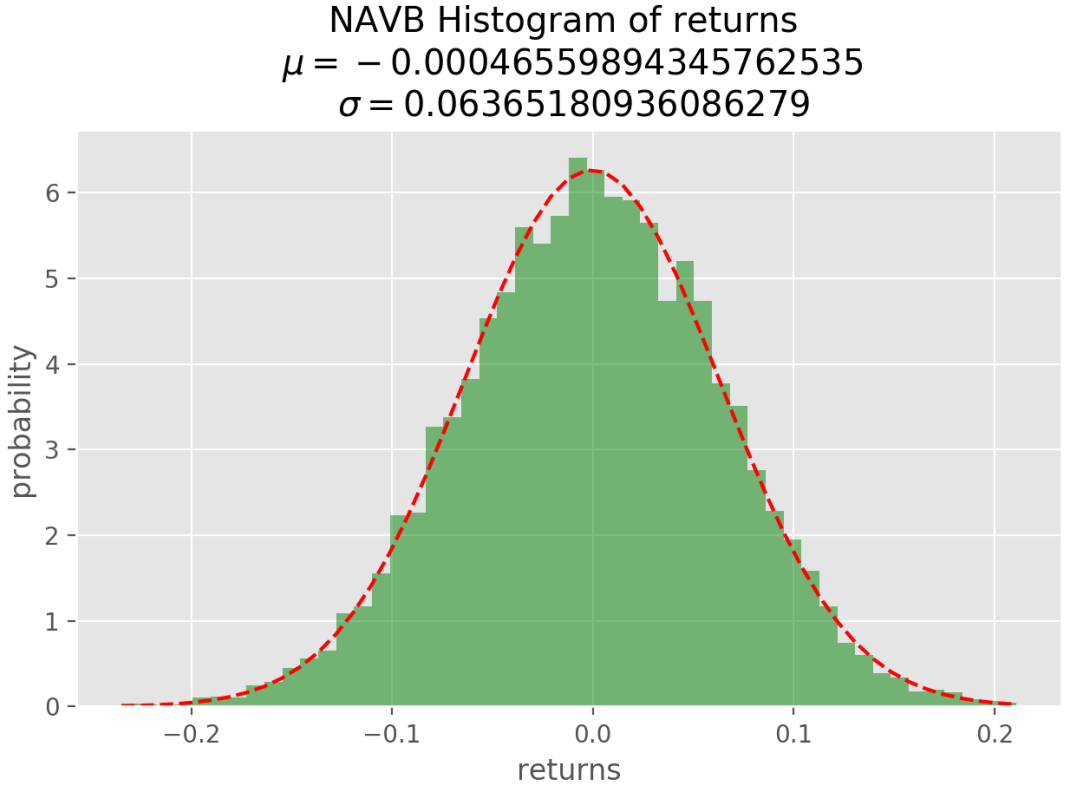

CDE Time Series Analysis Plots
0.25
0.00
−0.25
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

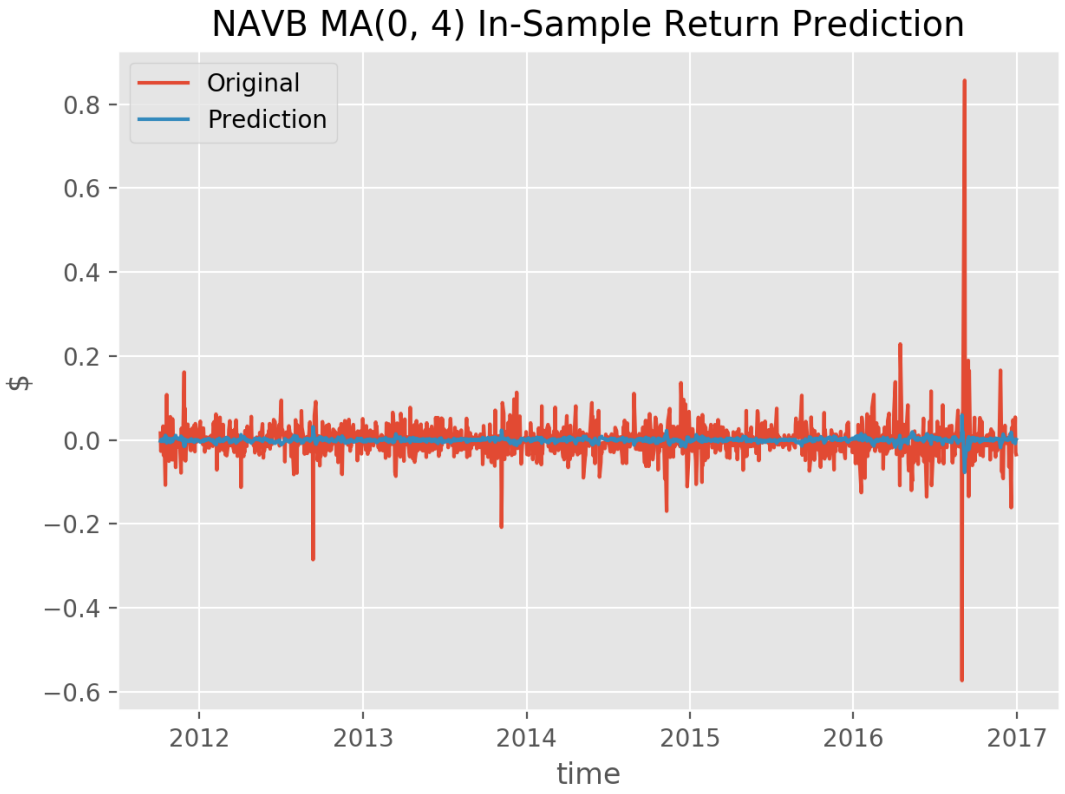

CDE Histogram of returns
$\mu = -0.0004574899340448124 7$
$\sigma = 0.042825809052270916$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

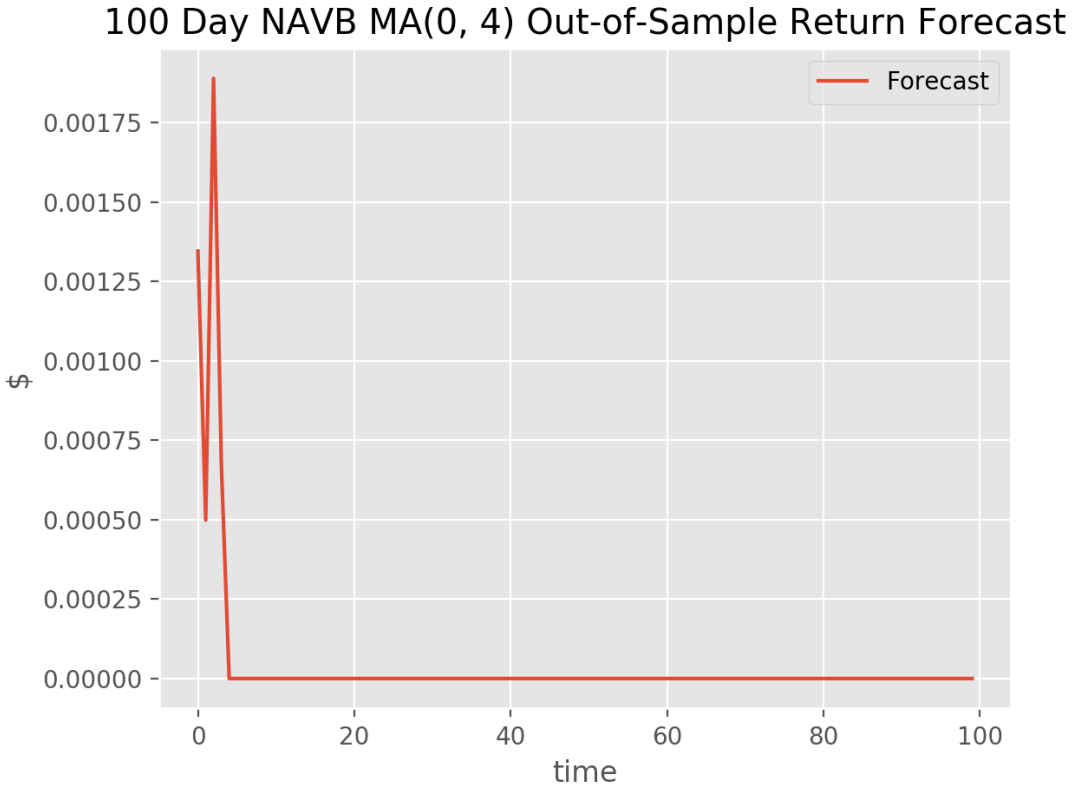

CDE MA(0, 1) In-Sample Return Prediction
Original
Prediction
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
$
2006
2008
2010
2012
2014
2016
time

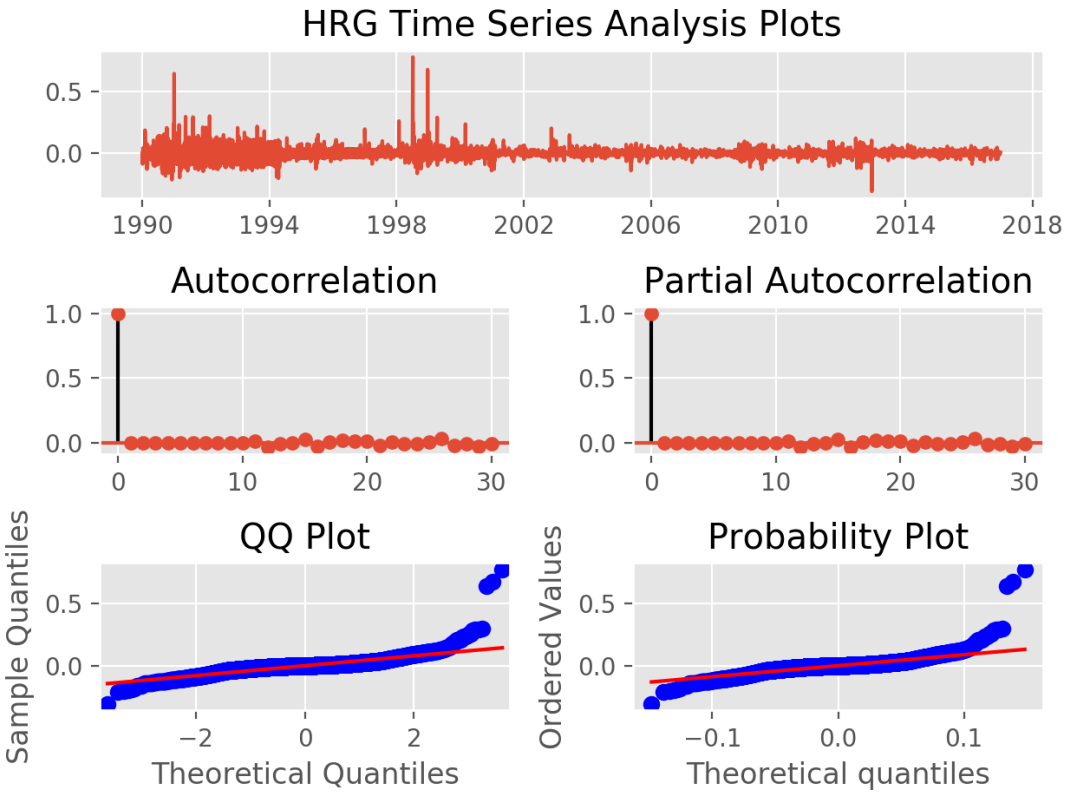

100 Day CDE MA(0, 1) Out-of-Sample Return Forecast
Forecast
0.006
0.005
0.004
0.003
0.002
0.001
0.000
$
0
20
40
60
80
100
time

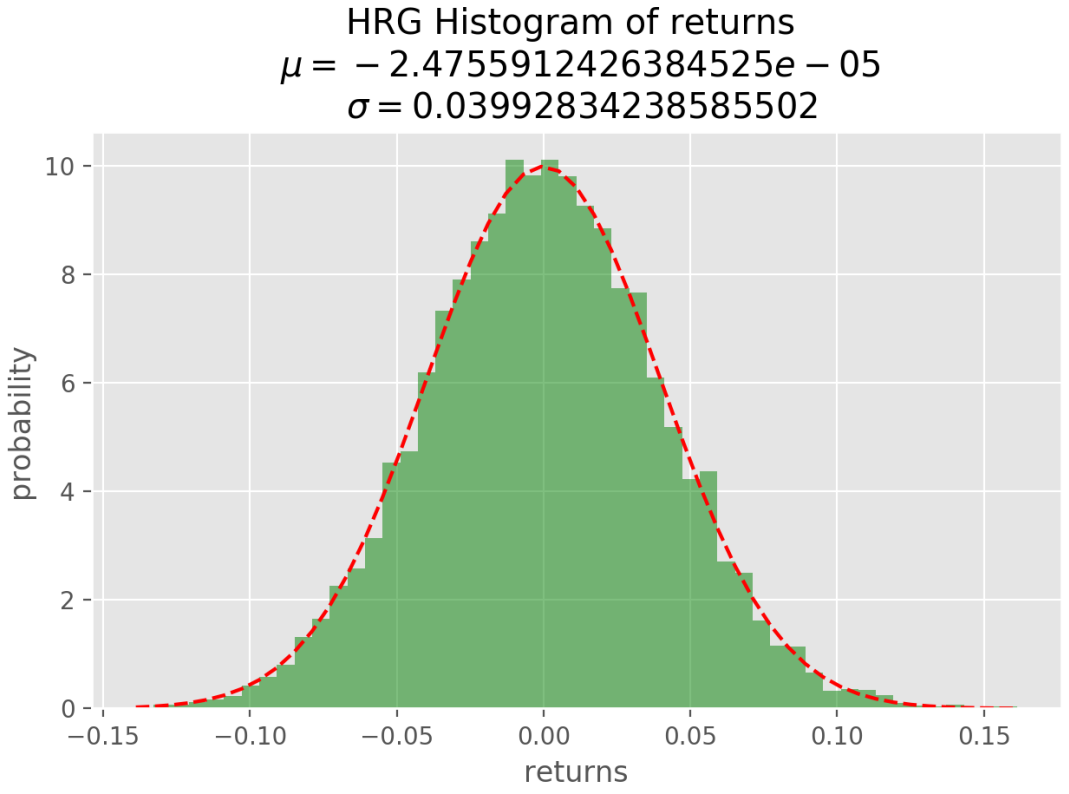

NAVB Time Series Analysis Plots
1
0
1993
1997
2001
2005
2009
2013
2017
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
1
0
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
1
0
−0.2
−0.1
0.0
0.1
0.2
Theoretical quantiles

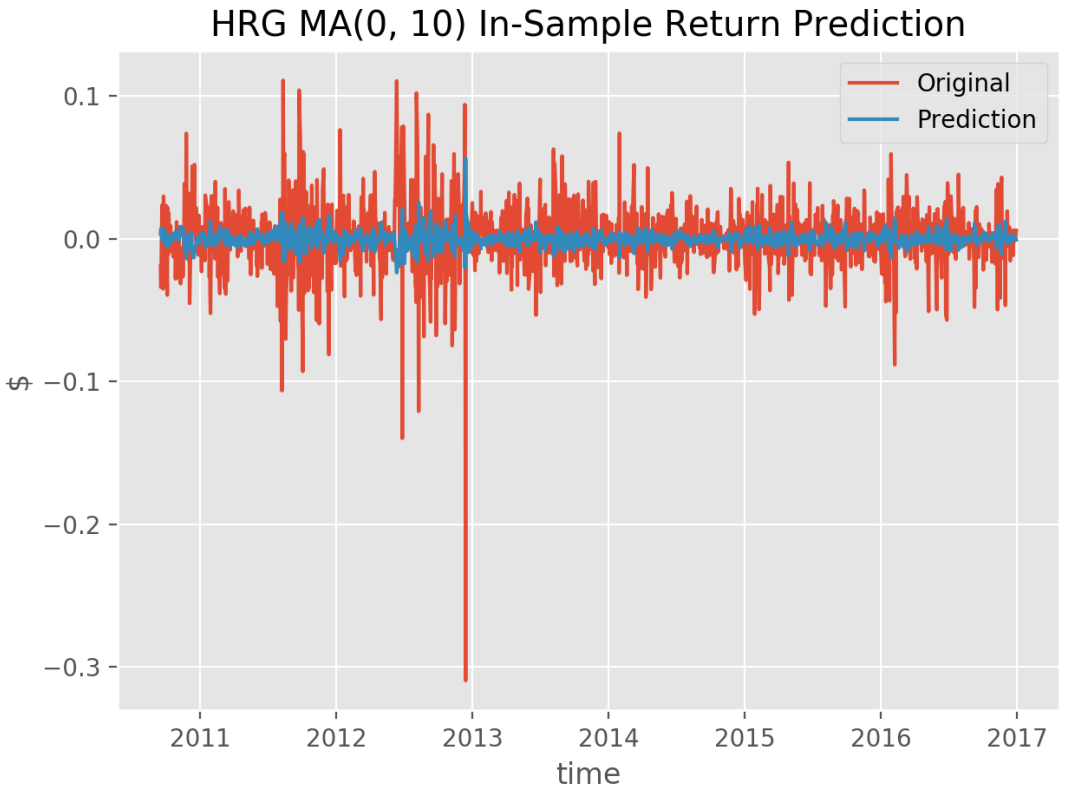

NAVB Histogram of returns
$\mu = -0.00046559894345762535$
$\sigma = 0.06365180936086279$
probability
6
5
4
3
2
1
0
−0.2
−0.1
0.0
0.1
0.2
returns

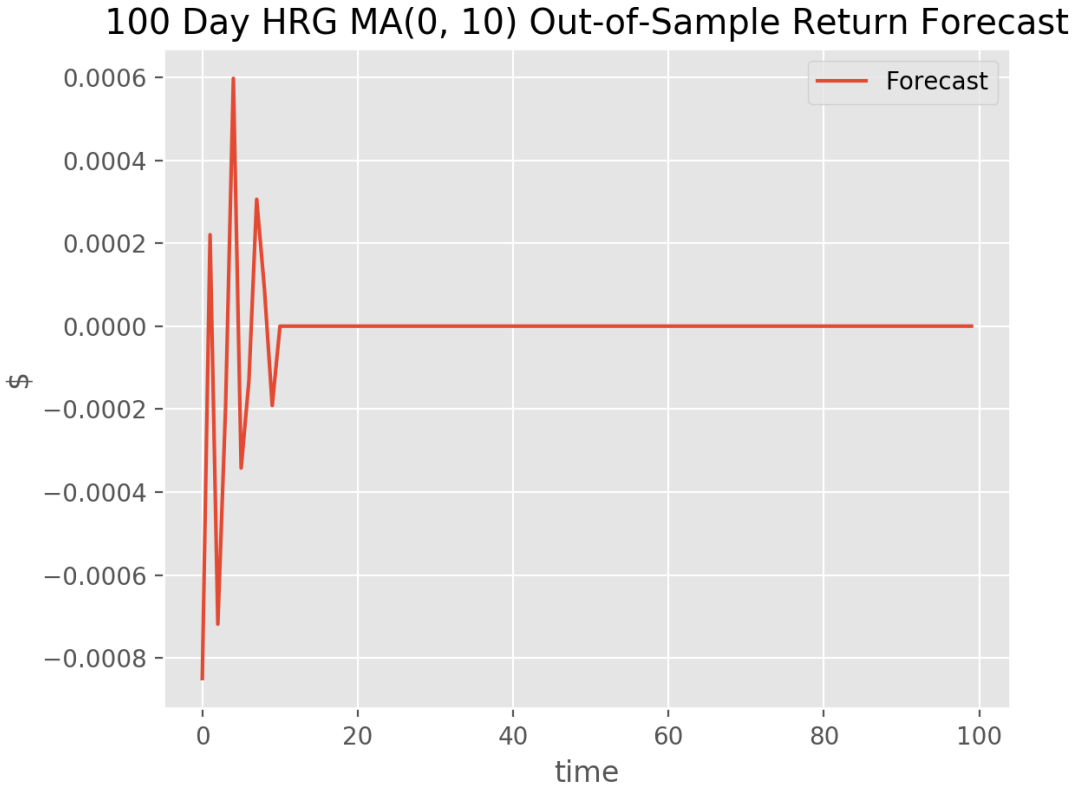

NAVB MA(0, 4) In-Sample Return Prediction
Original
Prediction
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
$
2012
2013
2014
2015
2016
2017
time

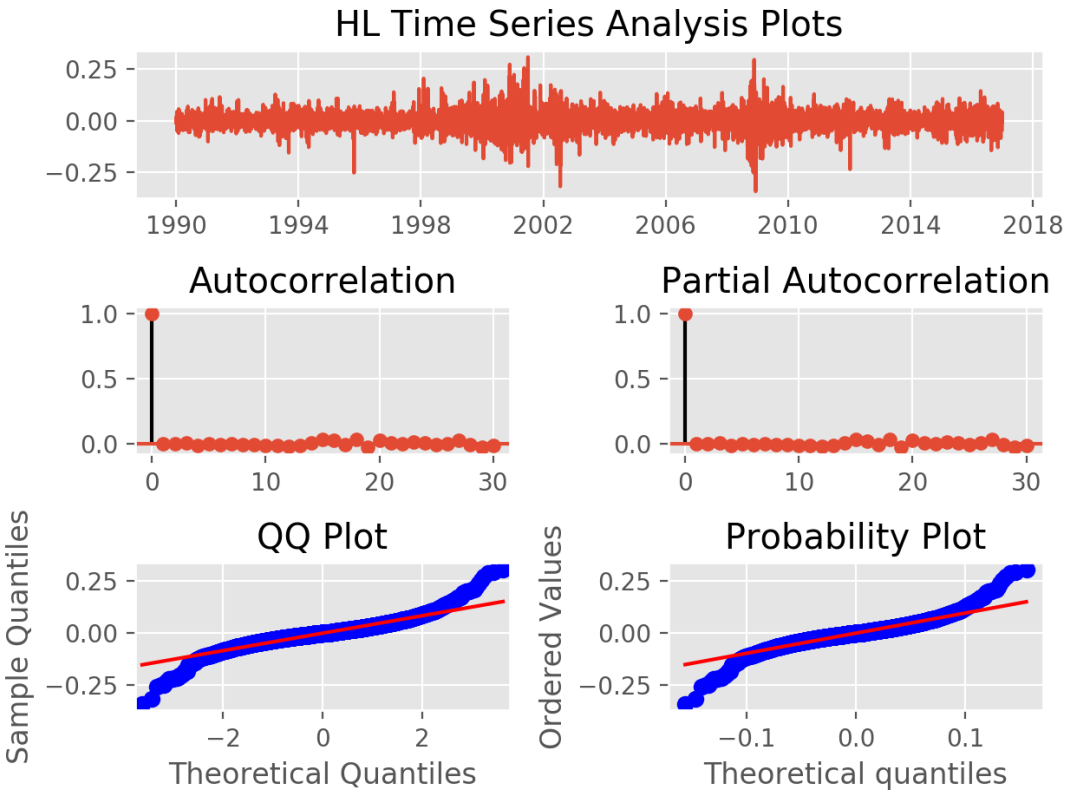

100 Day NAVB MA(0, 4) Out-of-Sample Return Forecast
Forecast
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
$
0
20
40
60
80
100
time

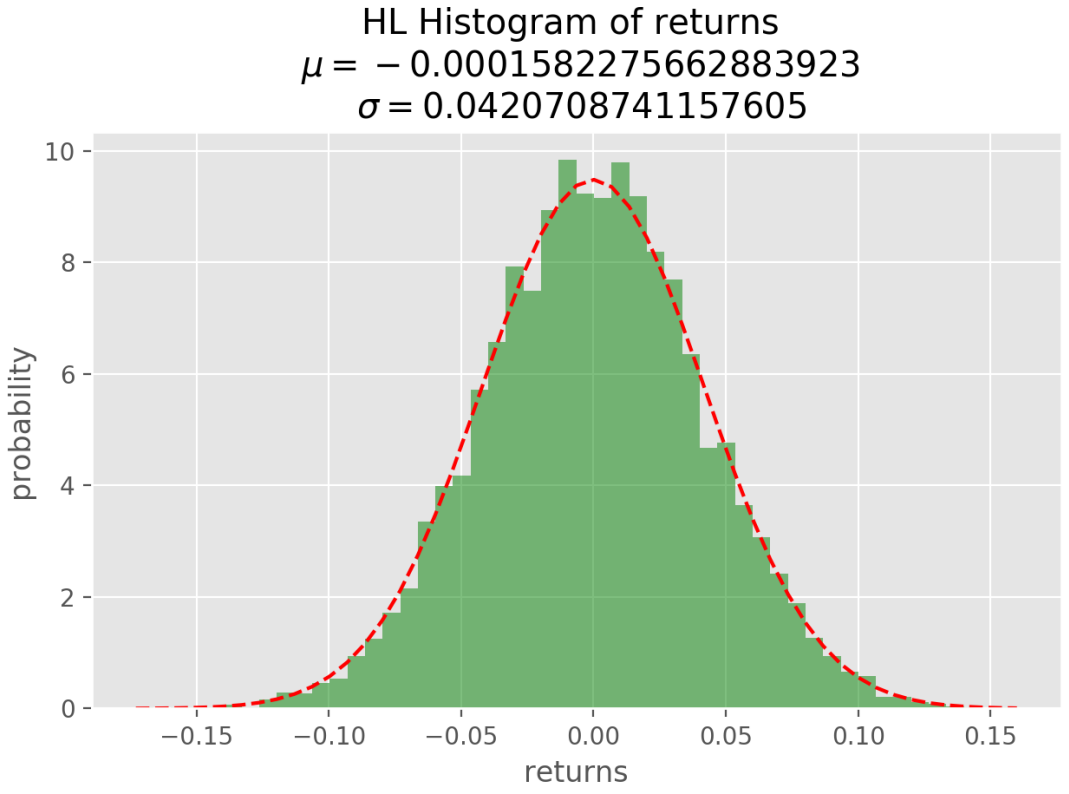

HRG Time Series Analysis Plots
0.5
0.0
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles

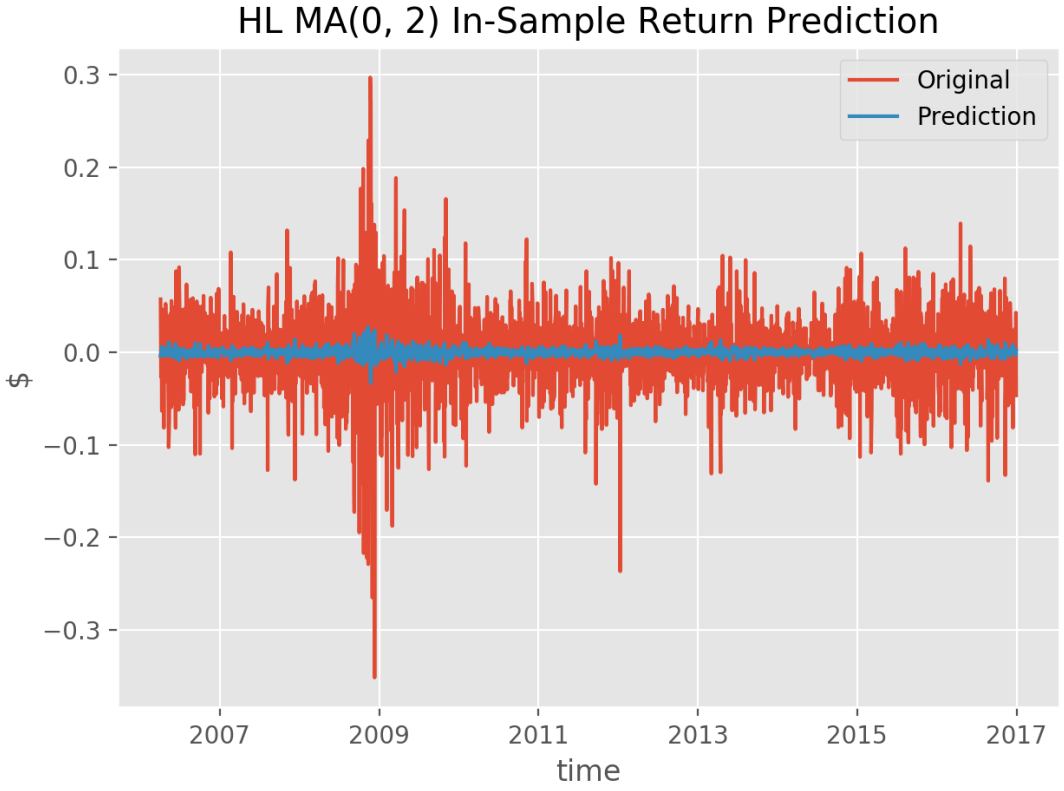

HRG Histogram of returns
$\mu = -2.4755912426384525e-05$
$\sigma = 0.03992834238585502$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

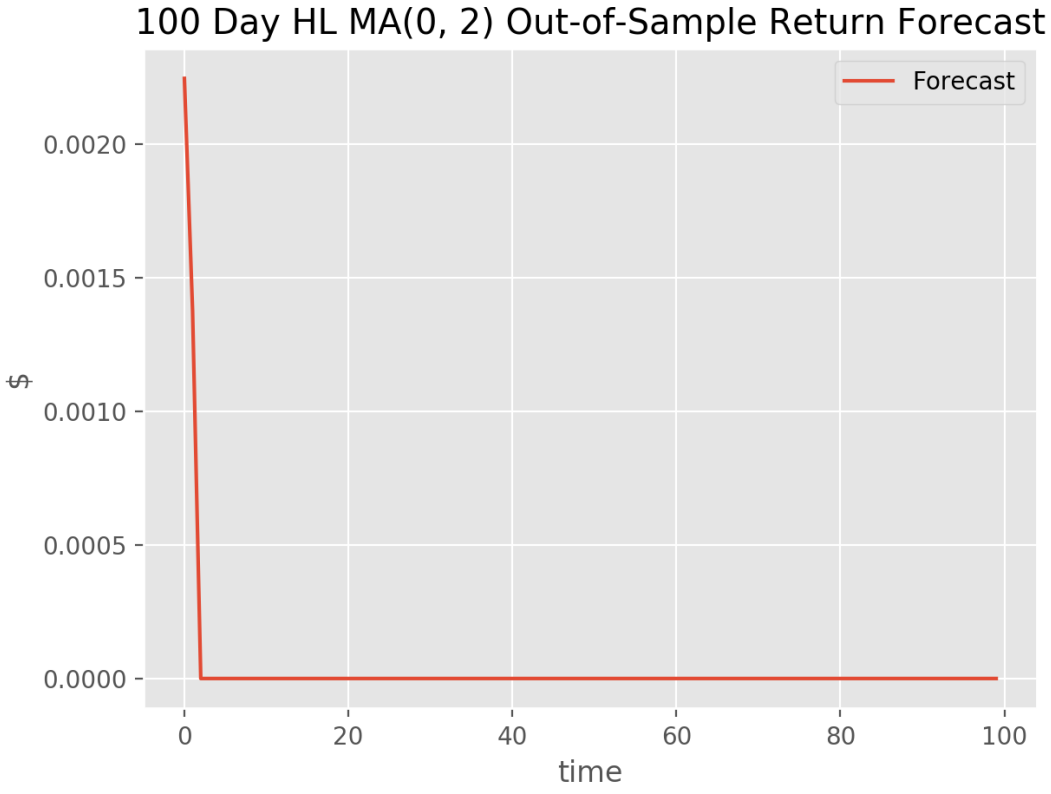

HRG MA(0, 10) In-Sample Return Prediction
Original
Prediction
0.1
0.0
−0.1
−0.2
−0.3
$
2011
2012
2013
2014
2015
2016
2017
time


100 Day HRG MA(0, 10) Out-of-Sample Return Forecast
Forecast
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
$
0
20
40
60
80
100
time

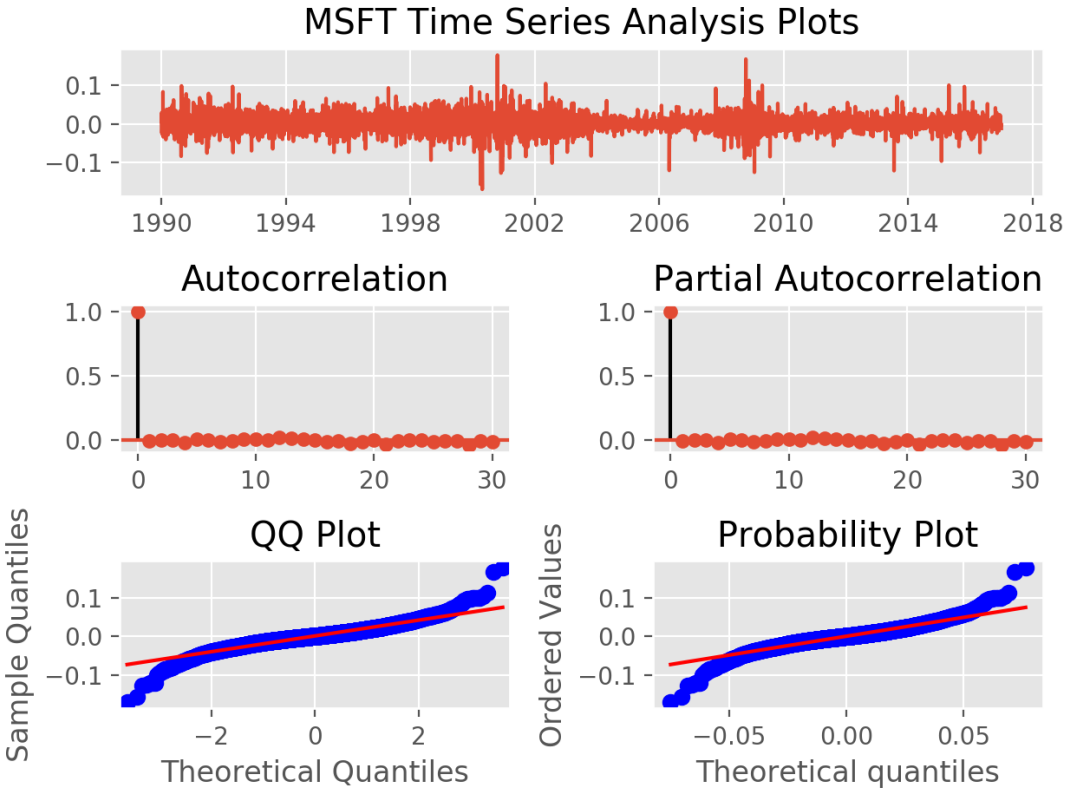

HL Time Series Analysis Plots
0.25
0.00
−0.25
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

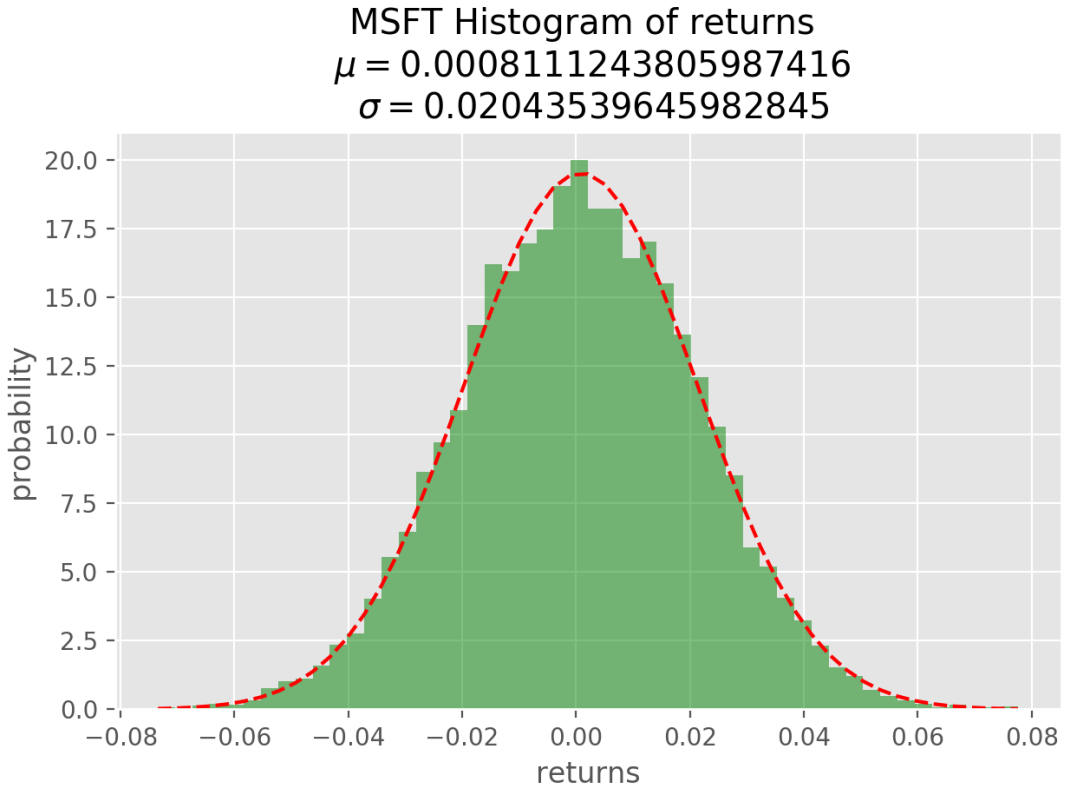

HL Histogram of returns
$\mu = -0.0001582275662883923$
$\sigma = 0.0420708741157605$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

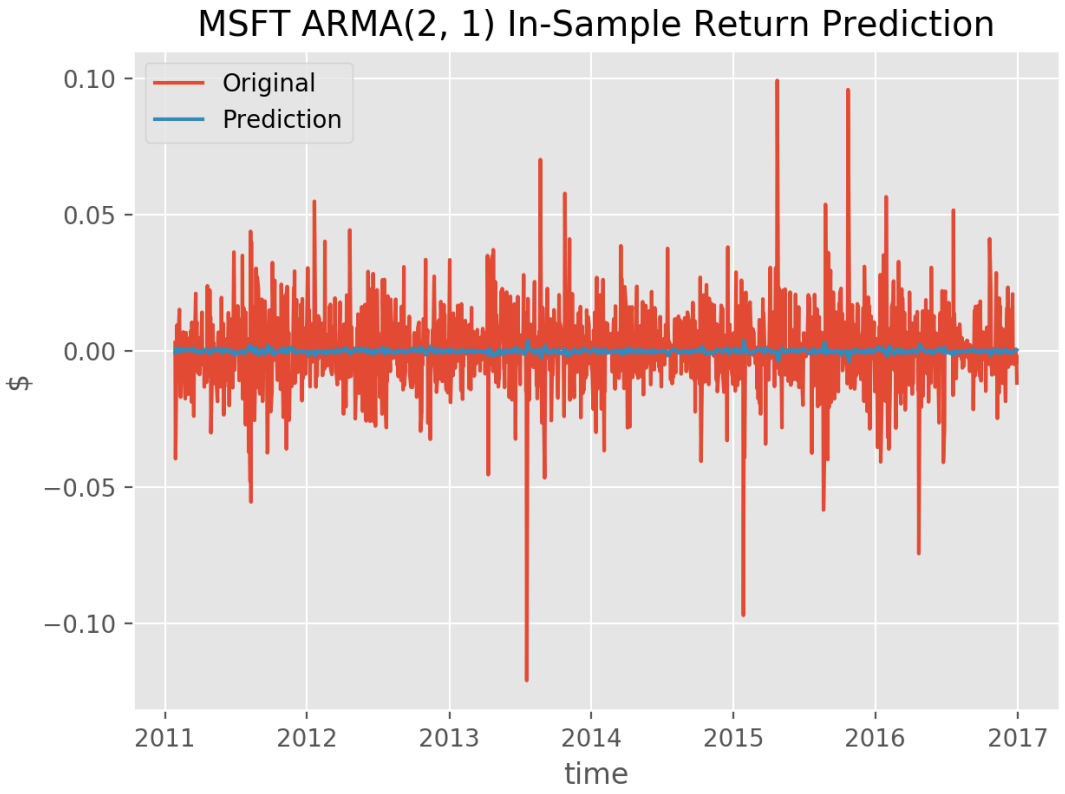

HL MA(0, 2) In-Sample Return Prediction
Original
Prediction
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
$
2007
2009
2011
2013
2015
2017
time


100 Day HL MA(0, 2) Out-of-Sample Return Forecast
Forecast
0.0020
0.0015
0.0010
0.0005
0.0000
$
0
20
40
60
80
100
time

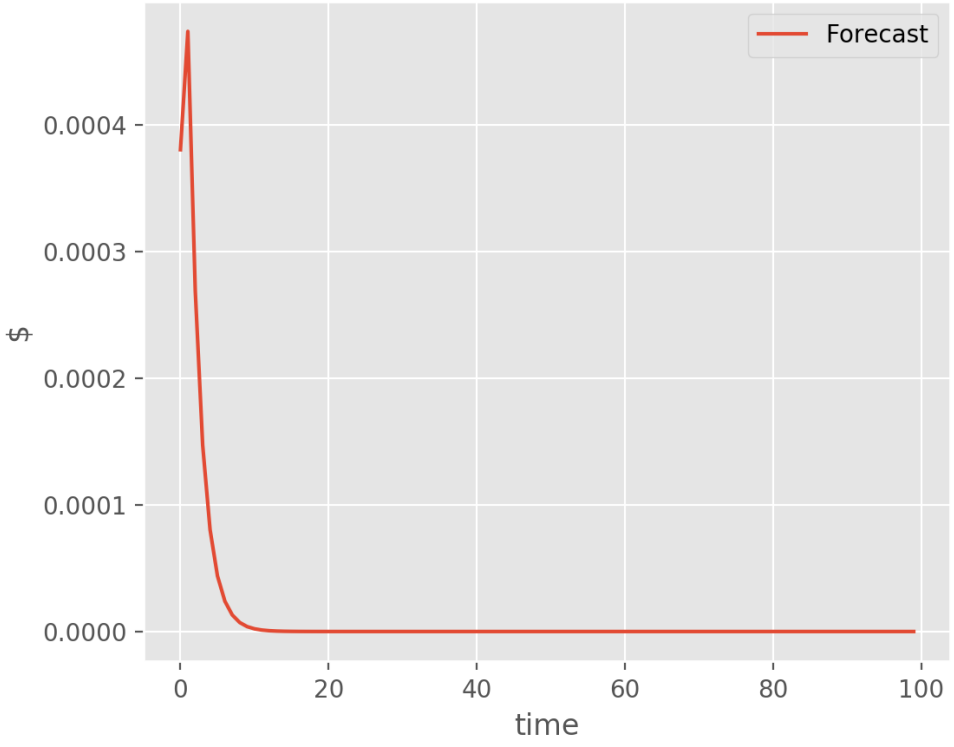
4.1.5 Auto Regressive Moving Average (ARMA)

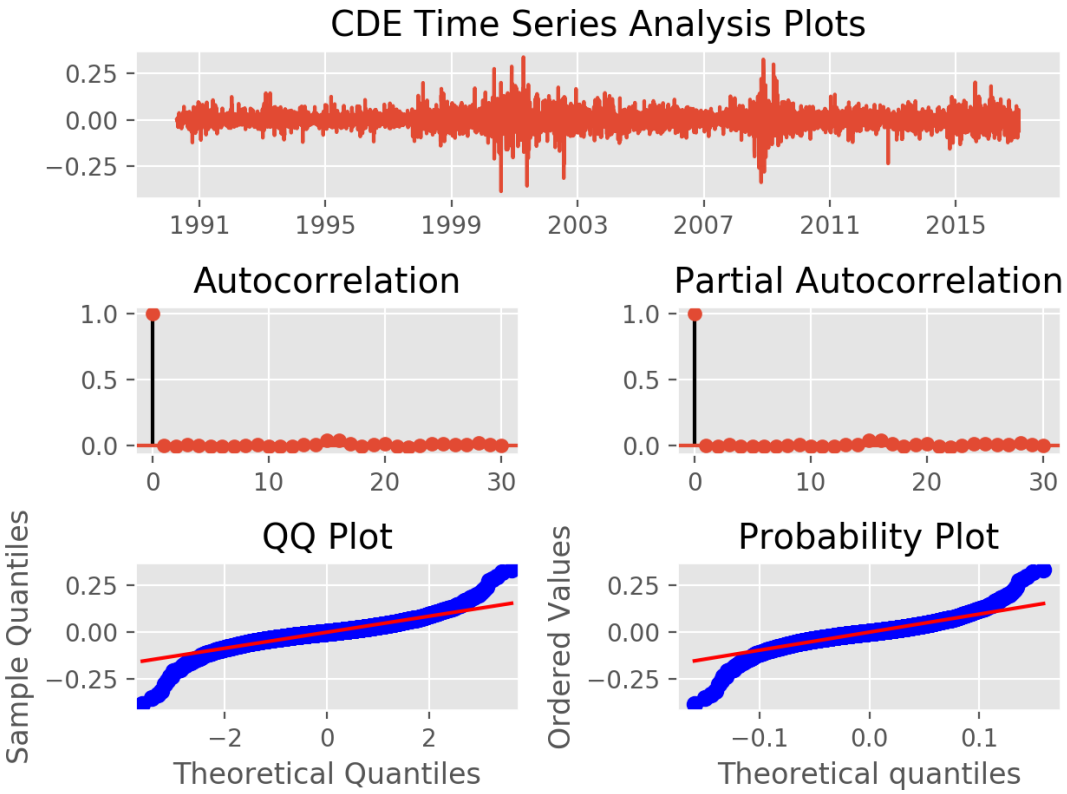


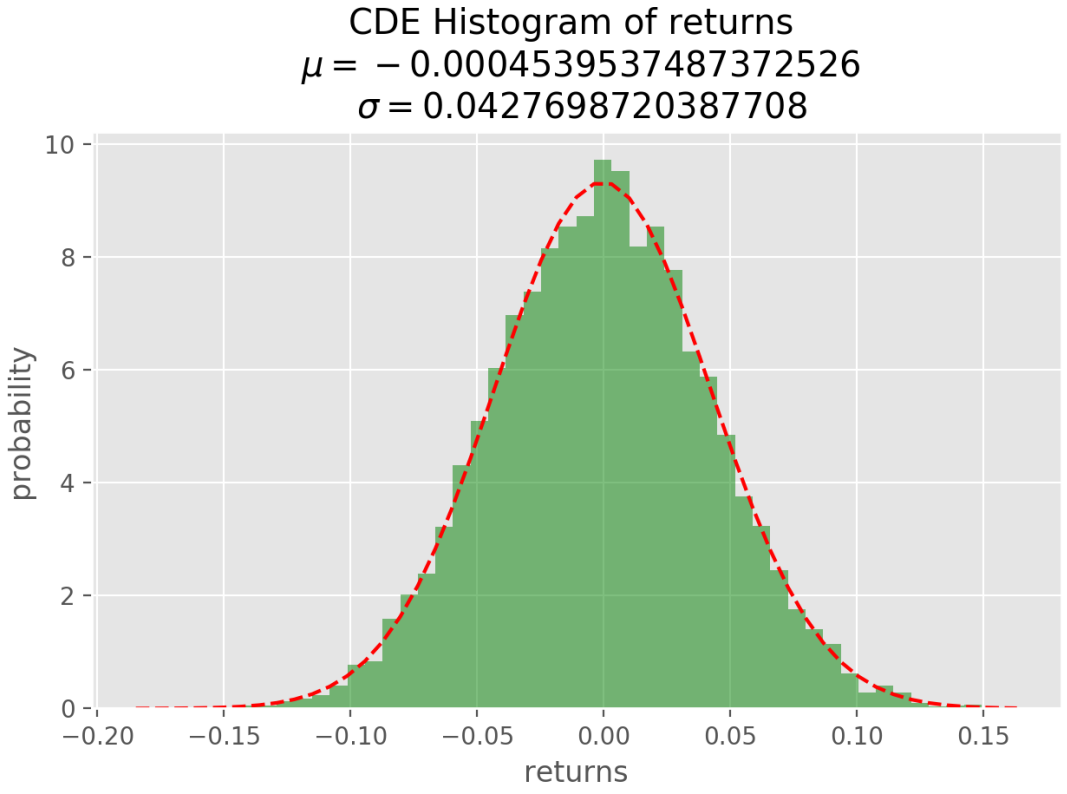


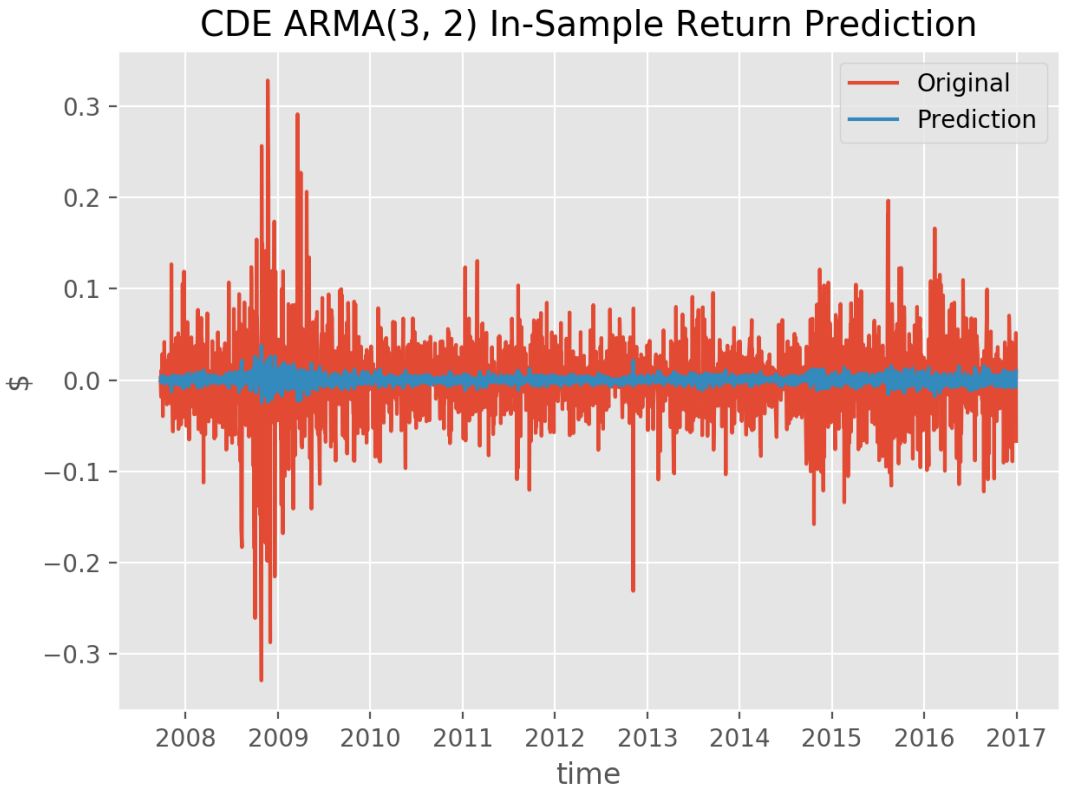


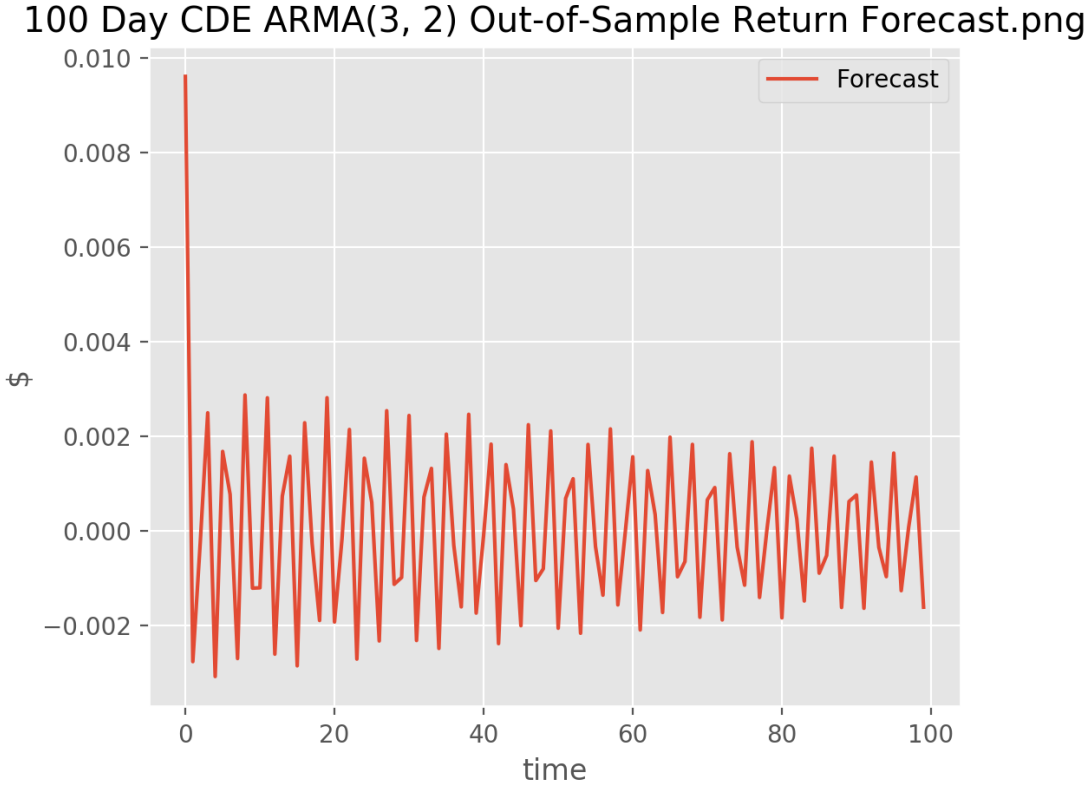
100 Day MSFT ARMA(2, 1) Out-of-Sample Return Forecast.png

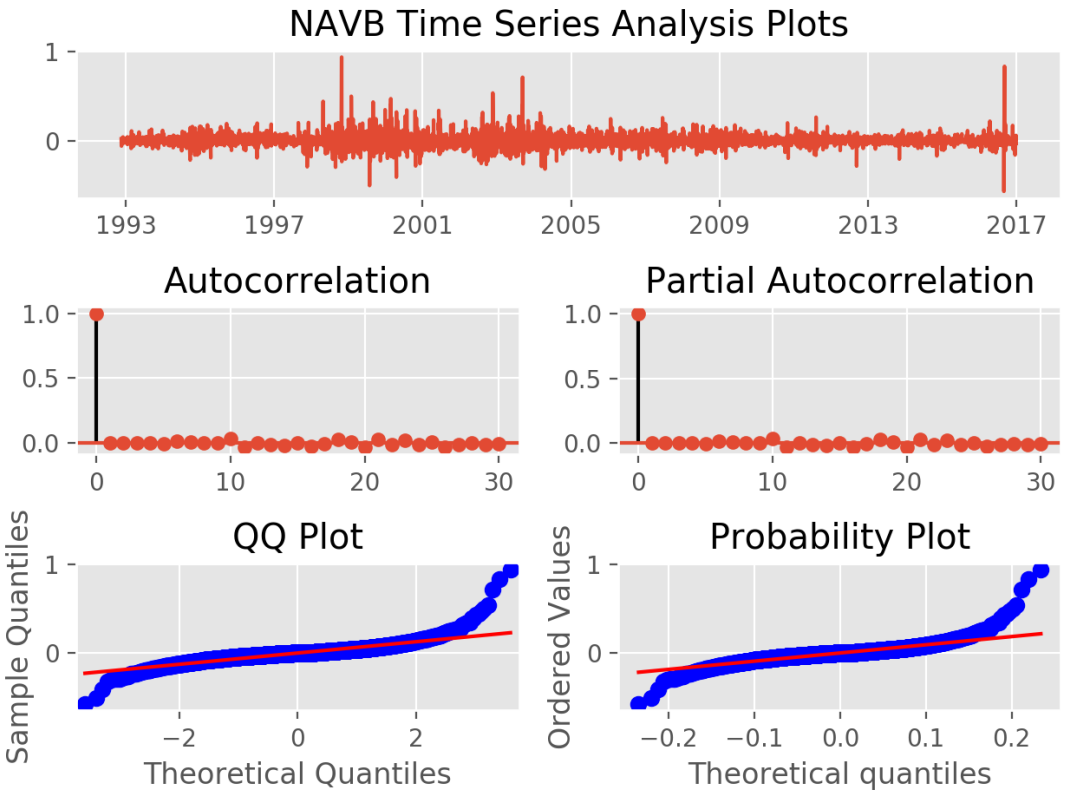


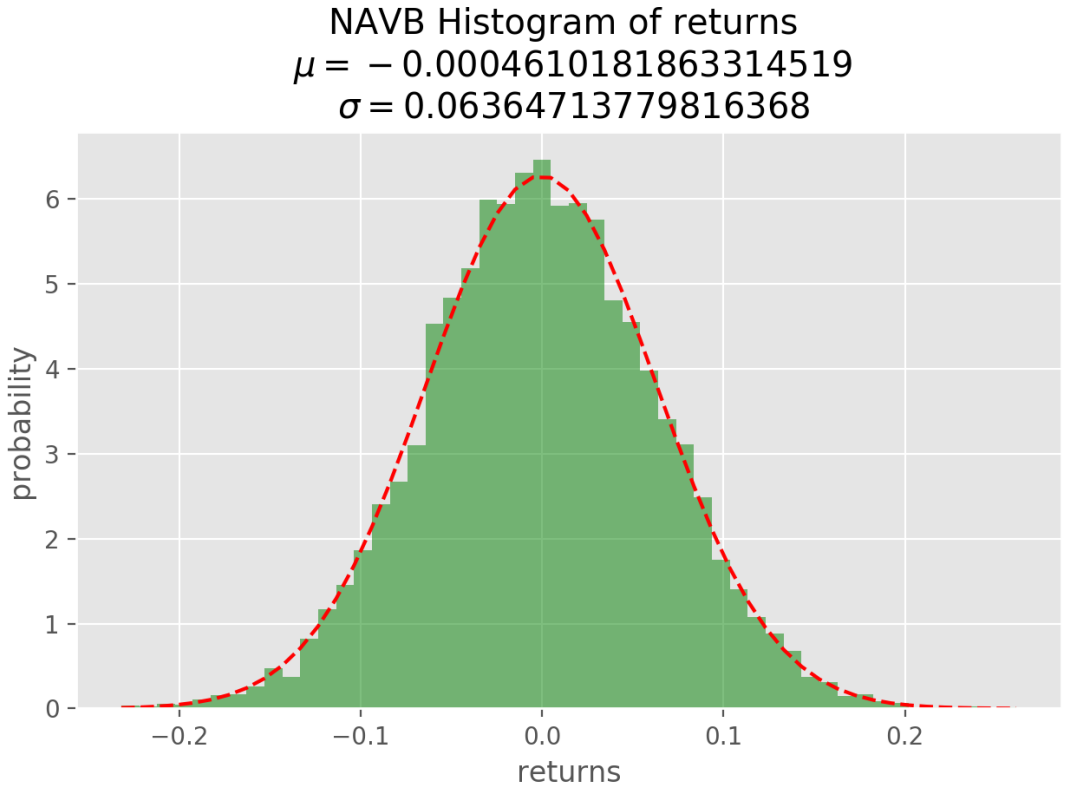

CDE Time Series Analysis Plots
0.25
0.00
−0.25
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

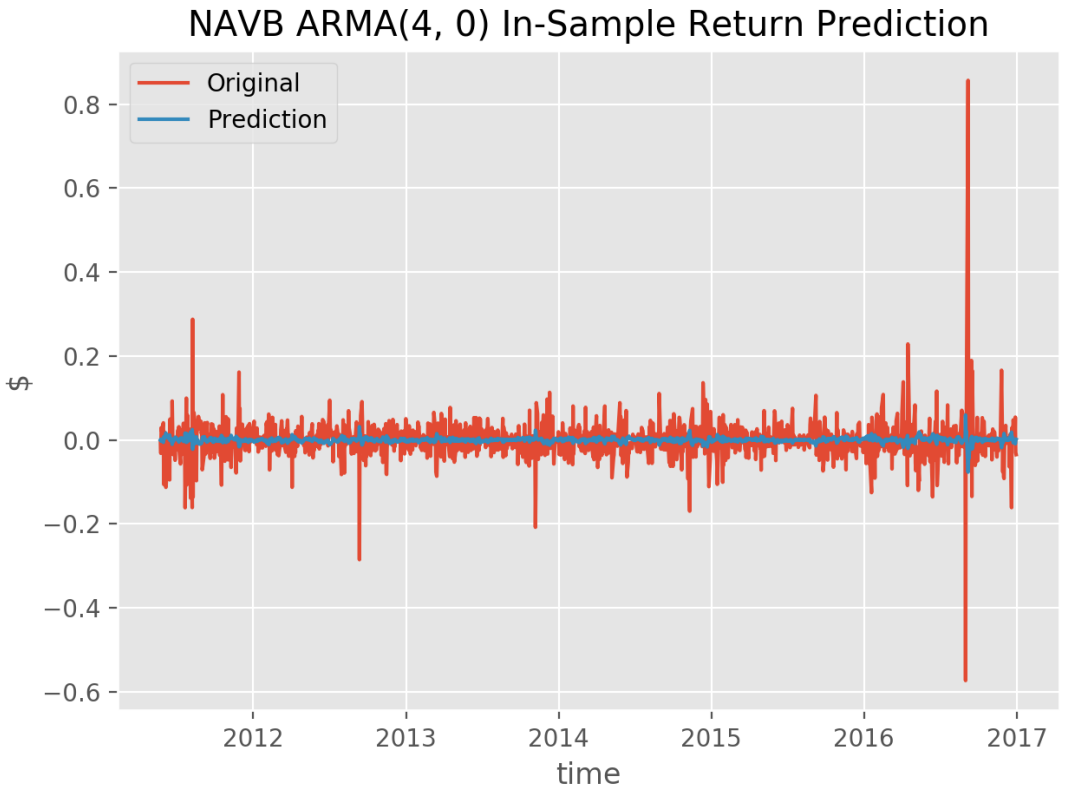

CDE Histogram of returns
$\mu = -0.0004539537487372526$
$\sigma = 0.0427698720387708$
probability
10
8
6
4
2
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns


CDE ARMA(3, 2) In-Sample Return Prediction
Original
Prediction
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
$
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

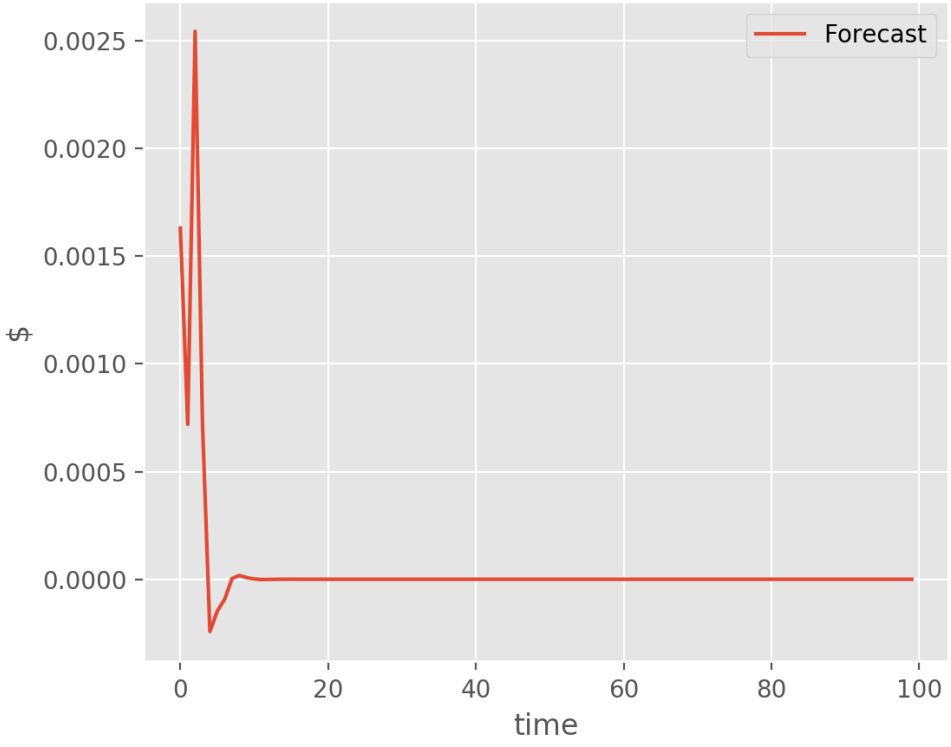

100 Day CDE ARMA(3, 2) Out-of-Sample Return Forecast.png
Forecast
0.010
0.008
0.006
0.004
0.002
0.000
−0.002
$
0
20
40
60
80
100
time

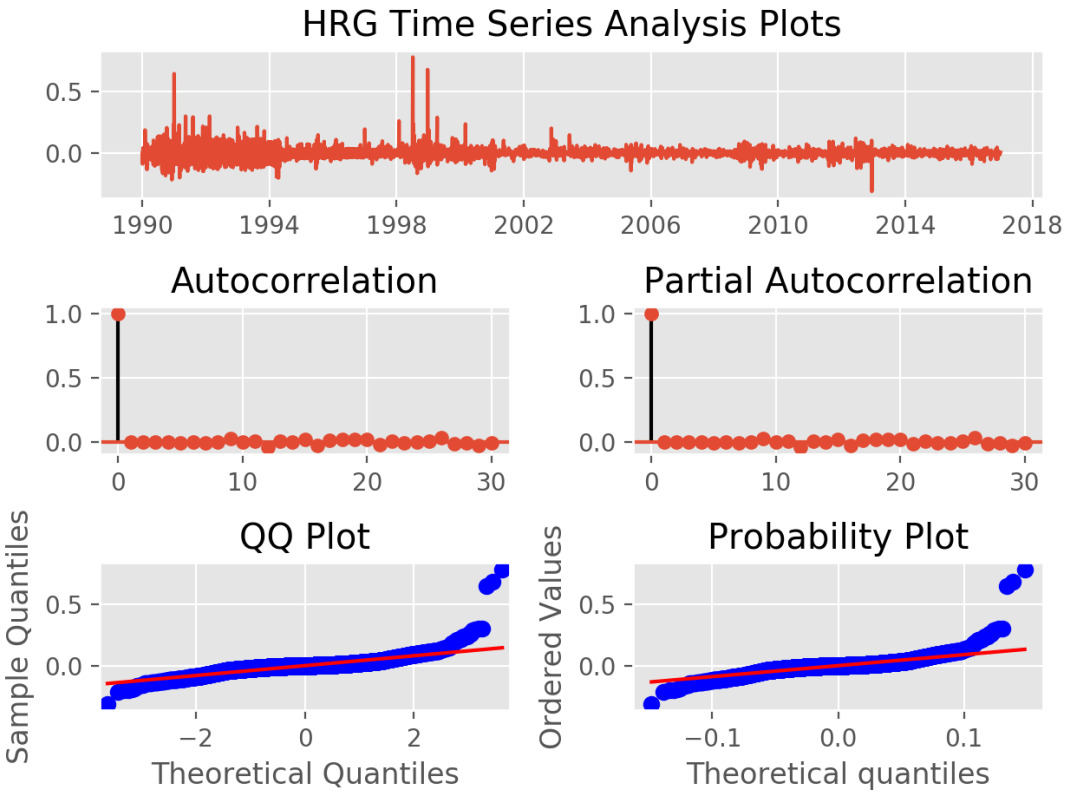

NAVB Time Series Analysis Plots
1
0
1993
1997
2001
2005
2009
2013
2017
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
1
0
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
1
0
−0.2
−0.1
0.0
0.1
0.2
Theoretical quantiles

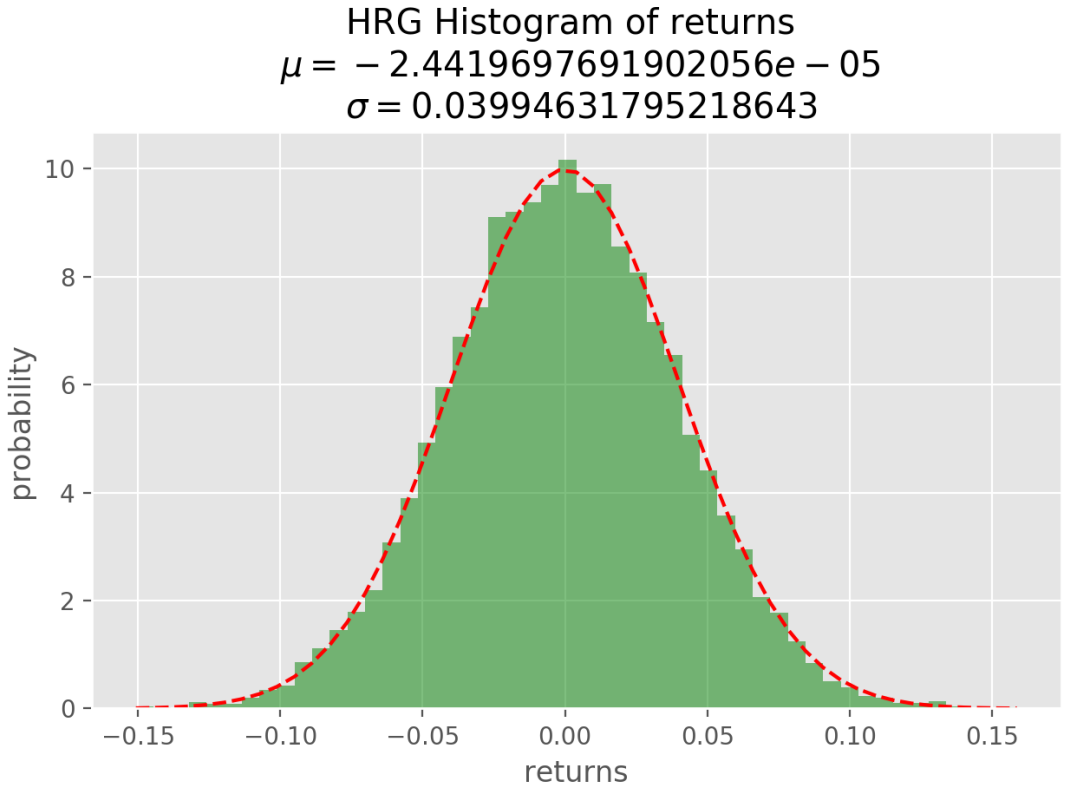

NAVB Histogram of returns
$\mu = -0.0004610181863314519$
$\sigma = 0.06364713779816368$
probability
6
5
4
3
2
1
0
−0.2
−0.1
0.0
0.1
0.2
returns

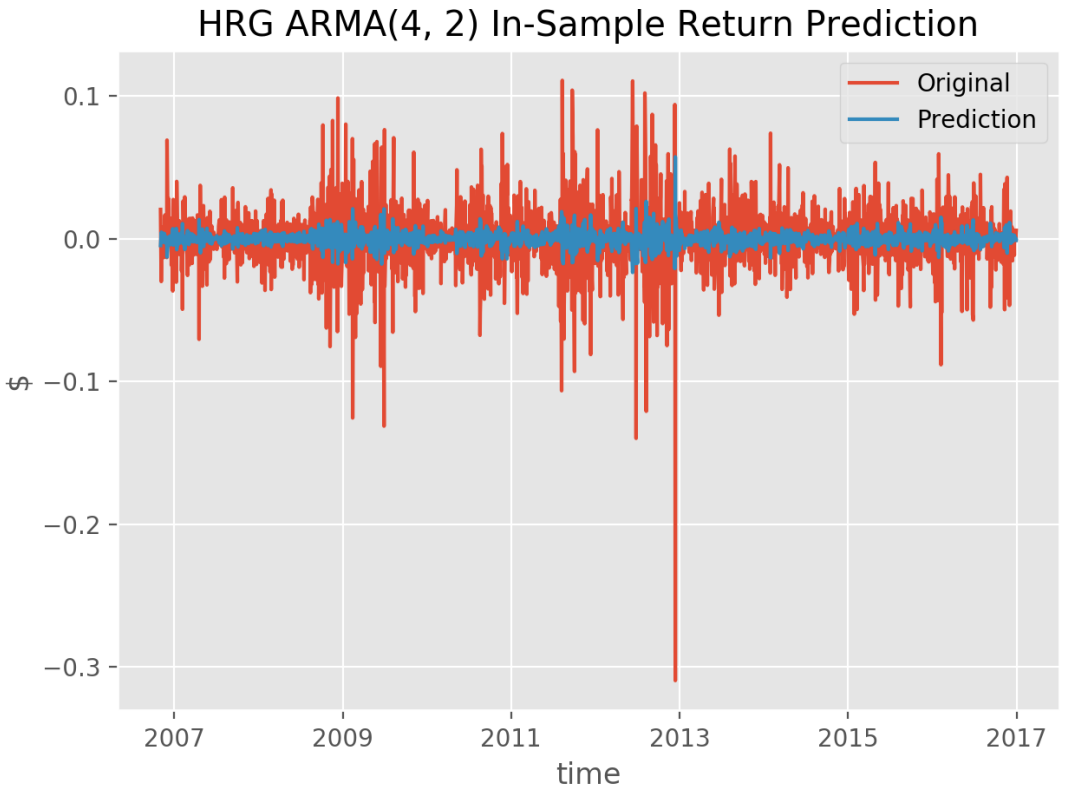


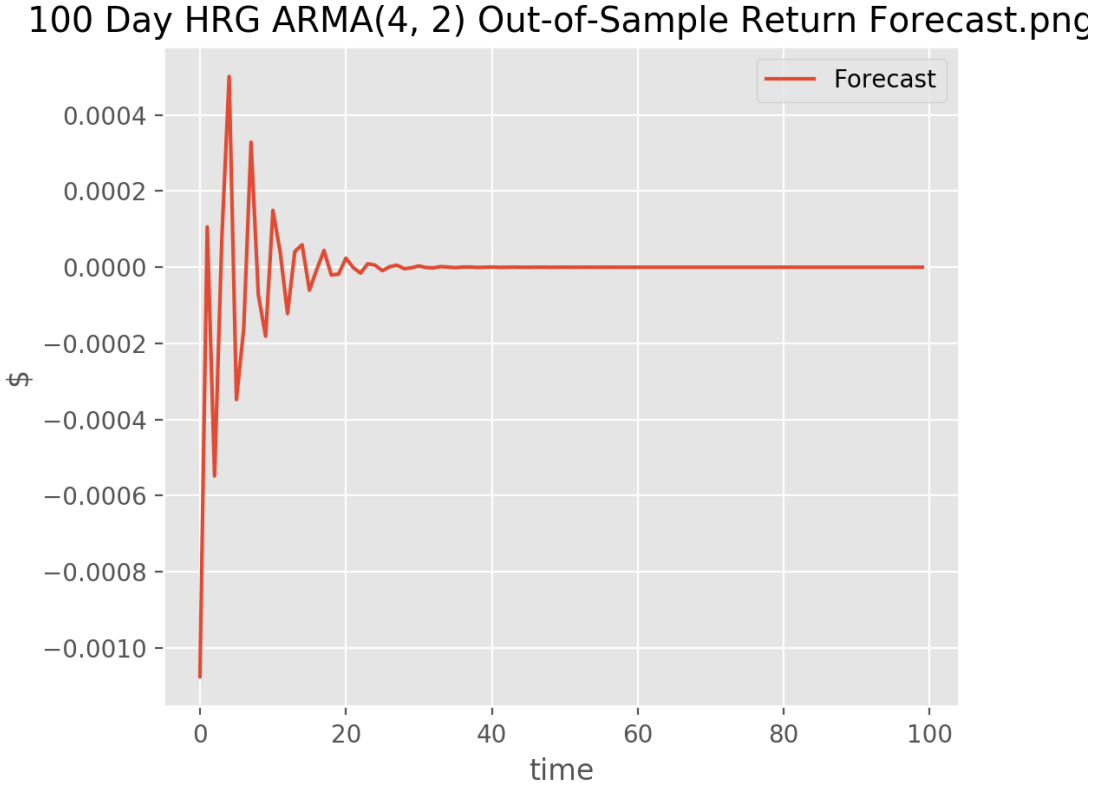
100 Day NAVB ARMA(4, 0) Out-of-Sample Return Forecast.png

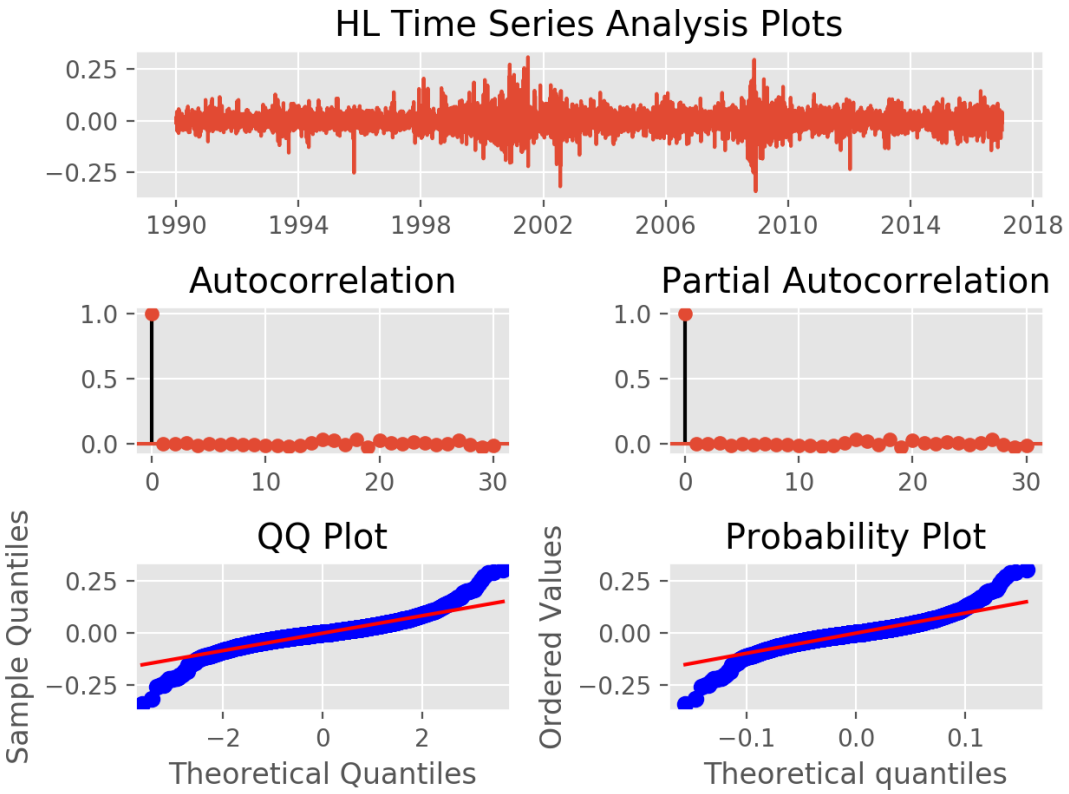


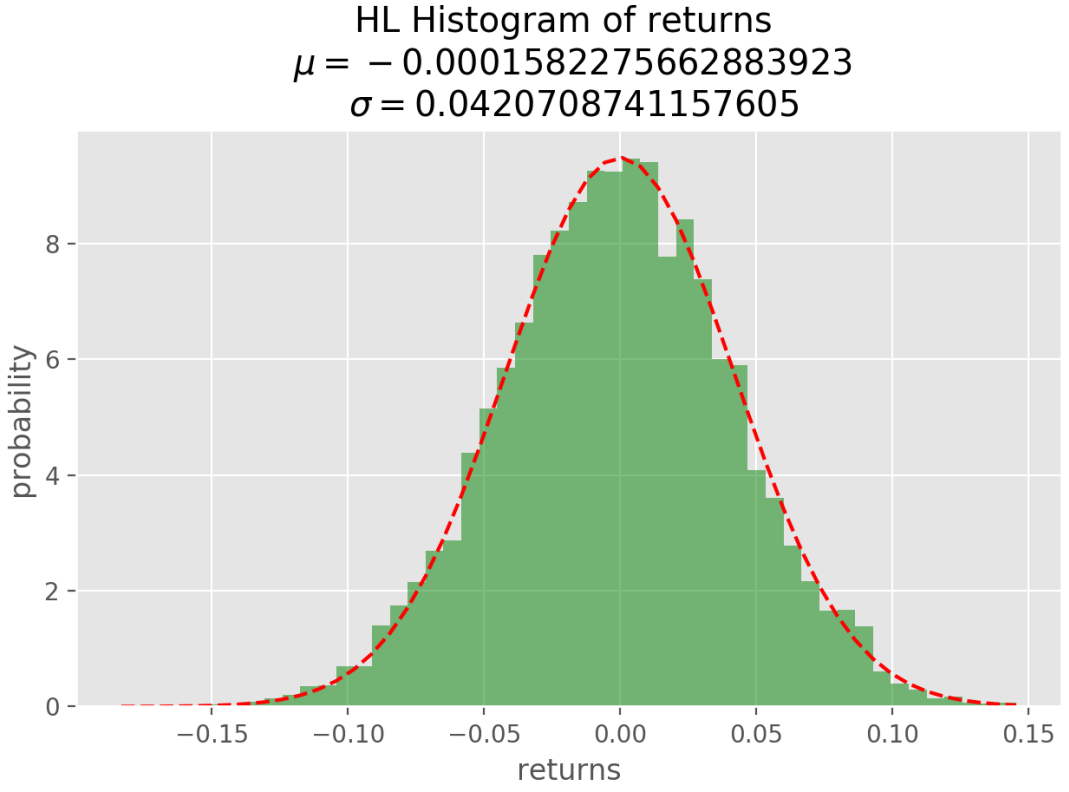

HRG Time Series Analysis Plots
0.5
0.0
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles

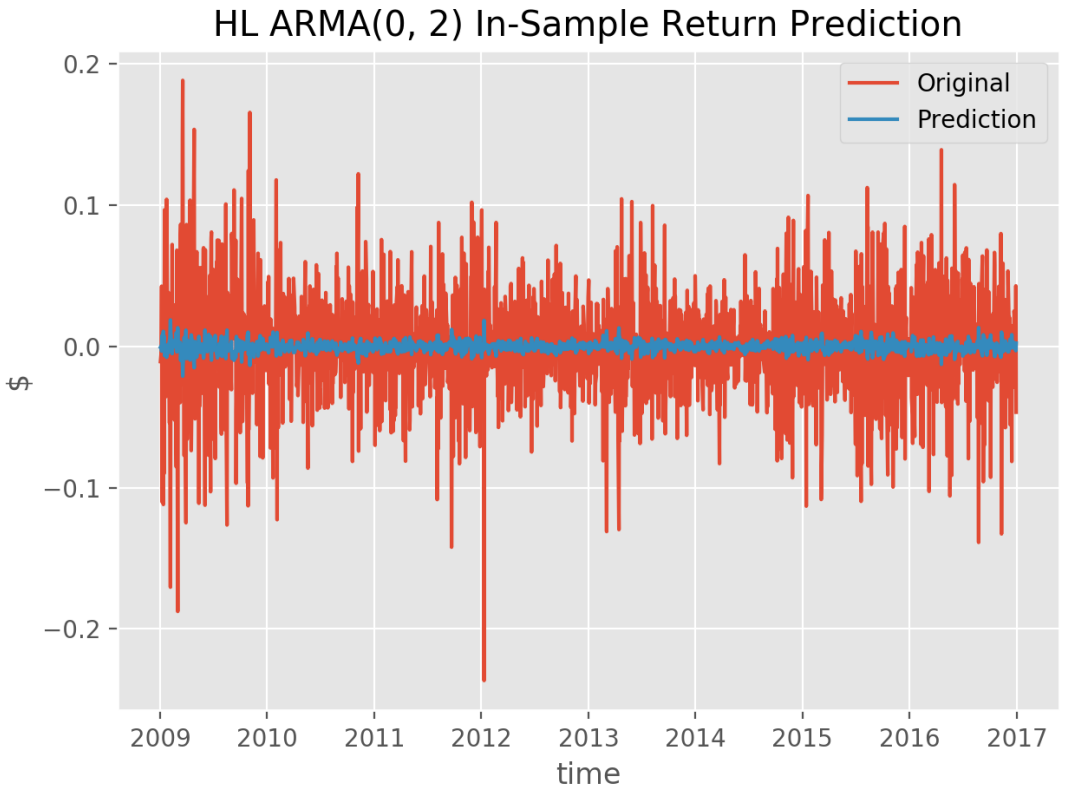

HRG Histogram of returns
$\mu = -2.4419697691902056e-05$
$\sigma = 0.03994631795218643$
probability
returns
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

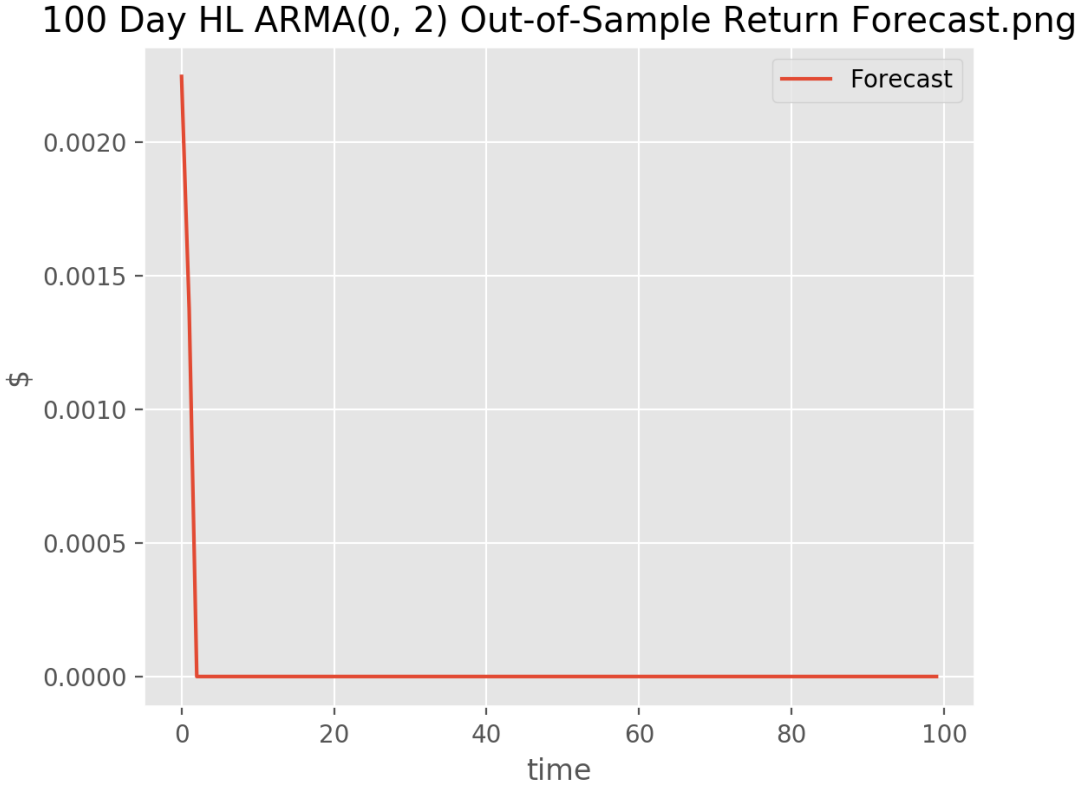

HRG ARMA(4, 2) In-Sample Return Prediction
Original
Prediction
0.1
0.0
−0.1
−0.2
−0.3
$
2007
2009
2011
2013
2015
2017
time


100 Day HRG ARMA(4, 2) Out-of-Sample Return Forecast.png
Forecast
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
$
0
20
40
60
80
100
time

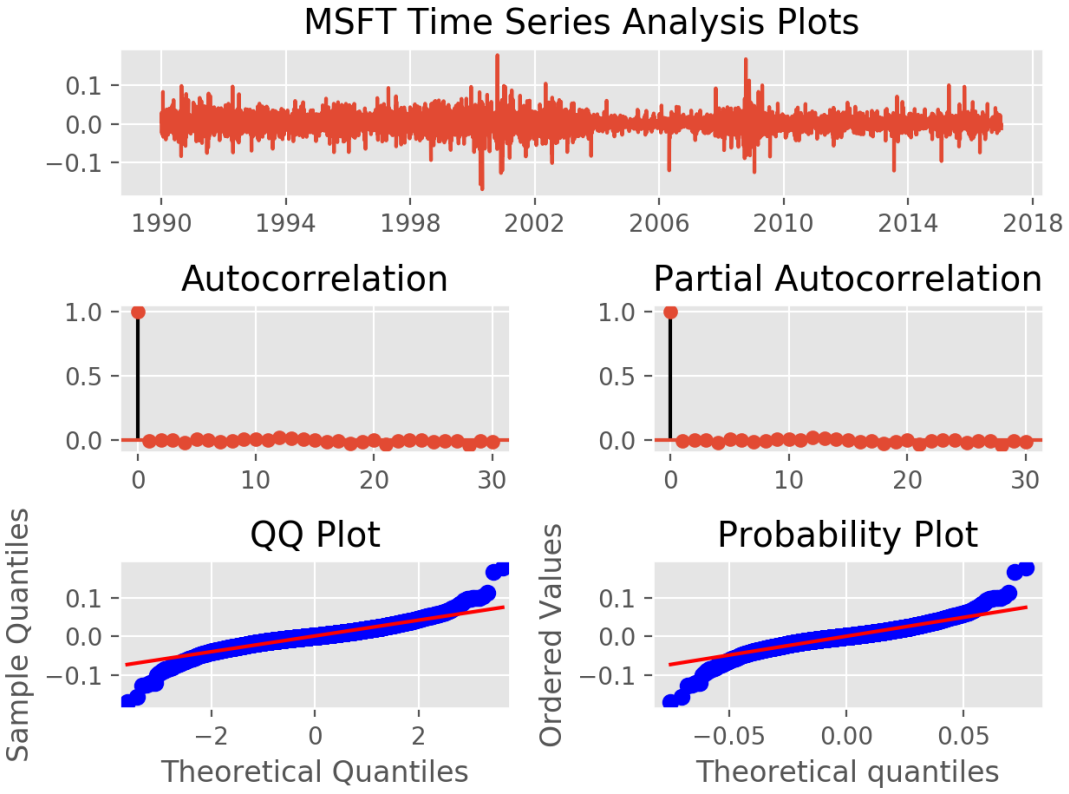

HL Time Series Analysis Plots
0.25
0.00
−0.25
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

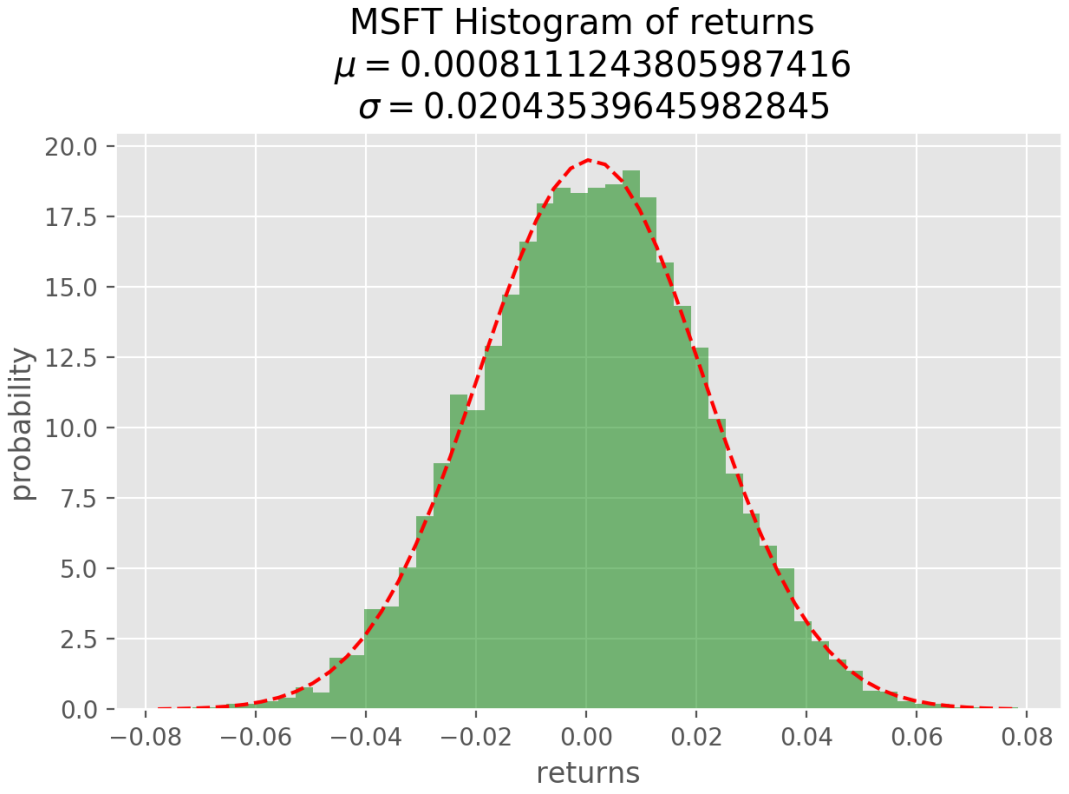

HL Histogram of returns
$\mu = -0.0001582275662883923$
$\sigma = 0.0420708741157605$
probability
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

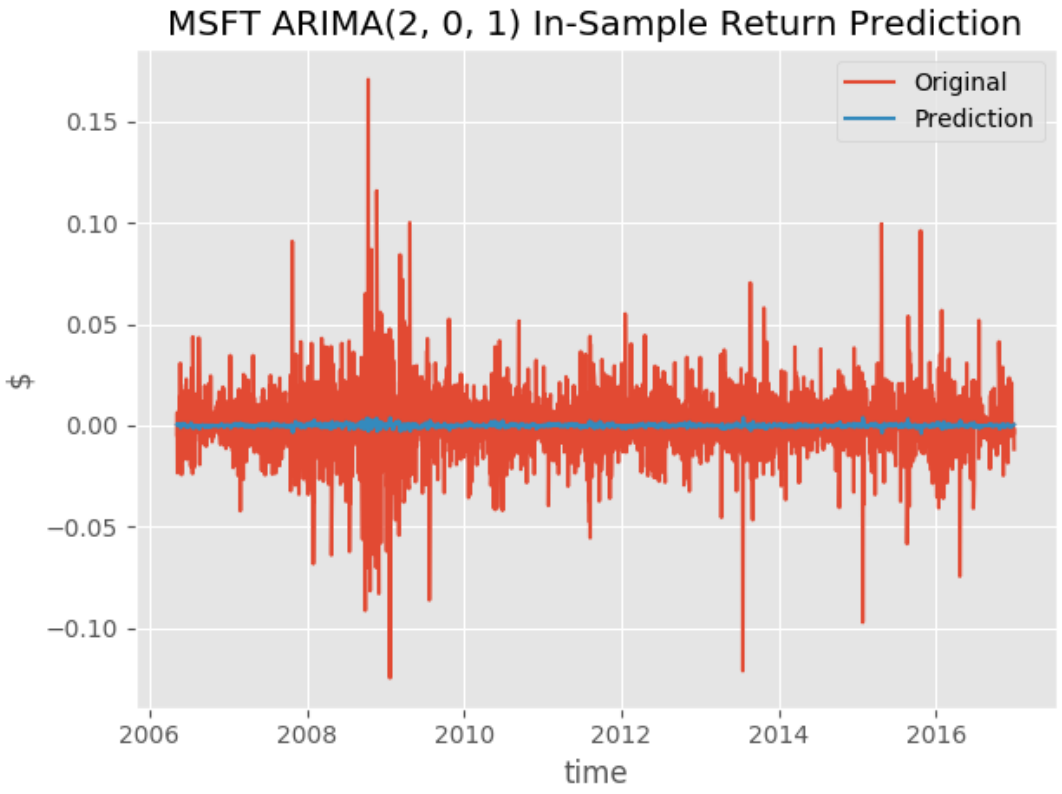

HL ARMA(0, 2) In-Sample Return Prediction
Original
Prediction
0.2
0.1
0.0
−0.1
−0.2
$
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

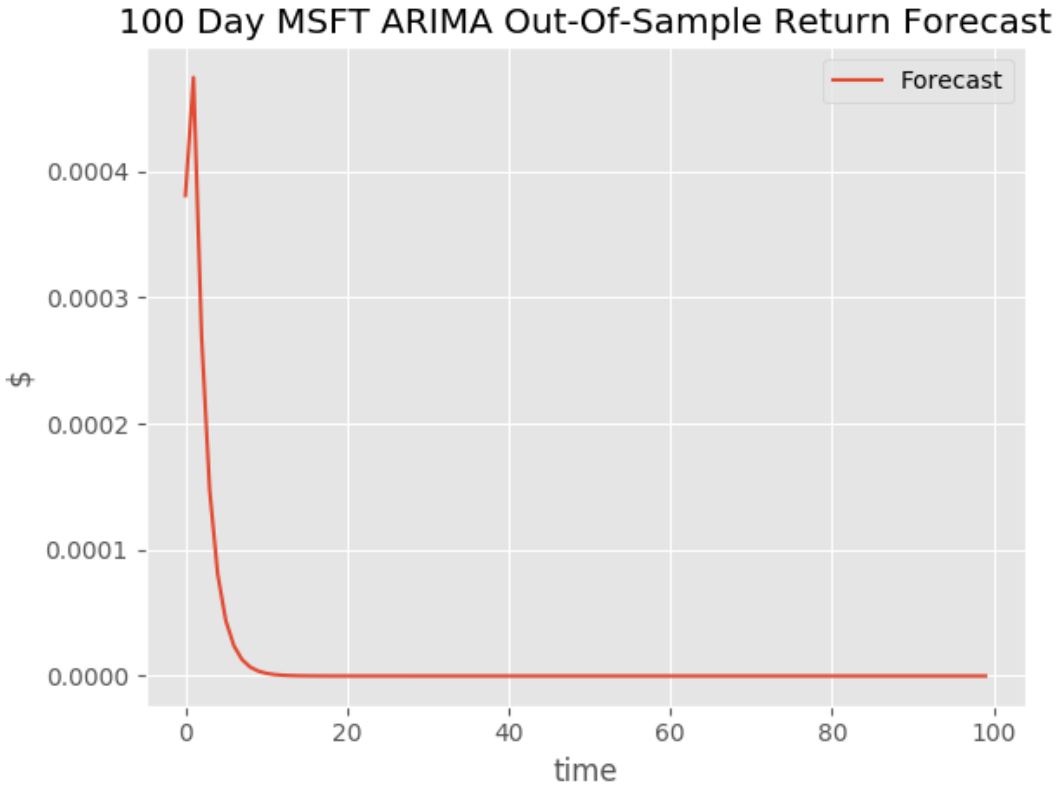

100 Day HL ARMA(0, 2) Out-of-Sample Return Forecast.png
Forecast
0.0020
0.0015
0.0010
0.0005
0.0000
$
0
20
40
60
80
100
time

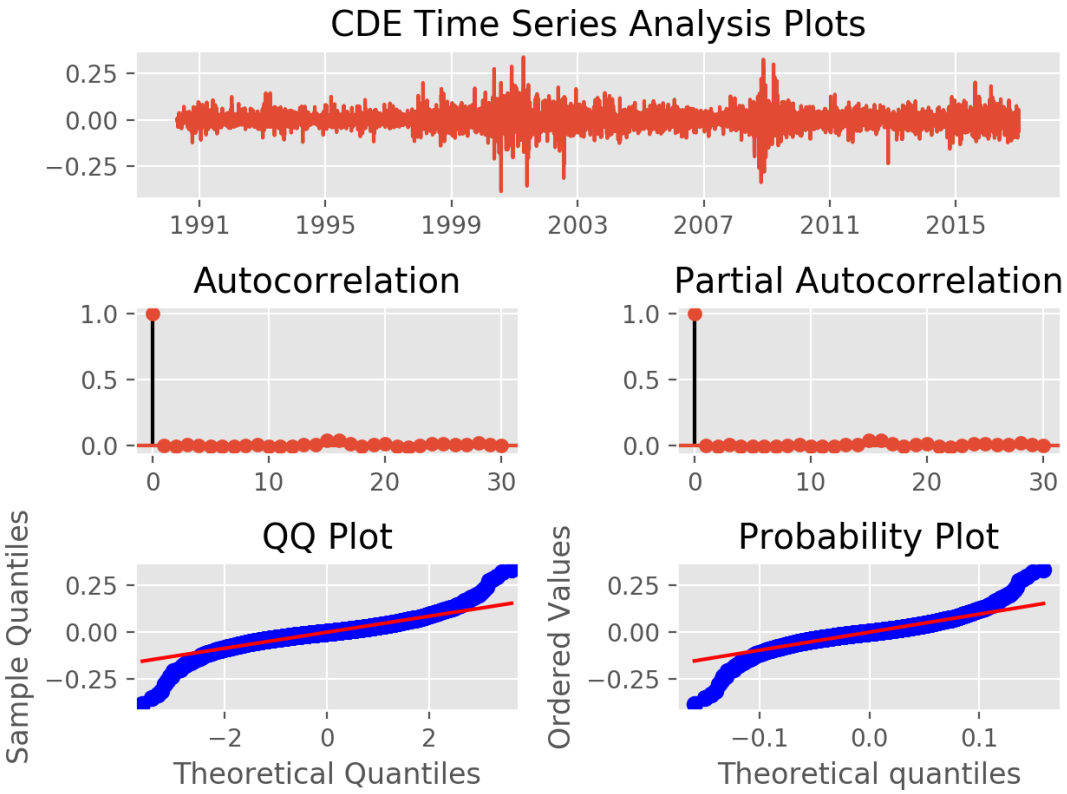
4.1.6 Auto Regressive Integrated Moving Average (ARIMA)

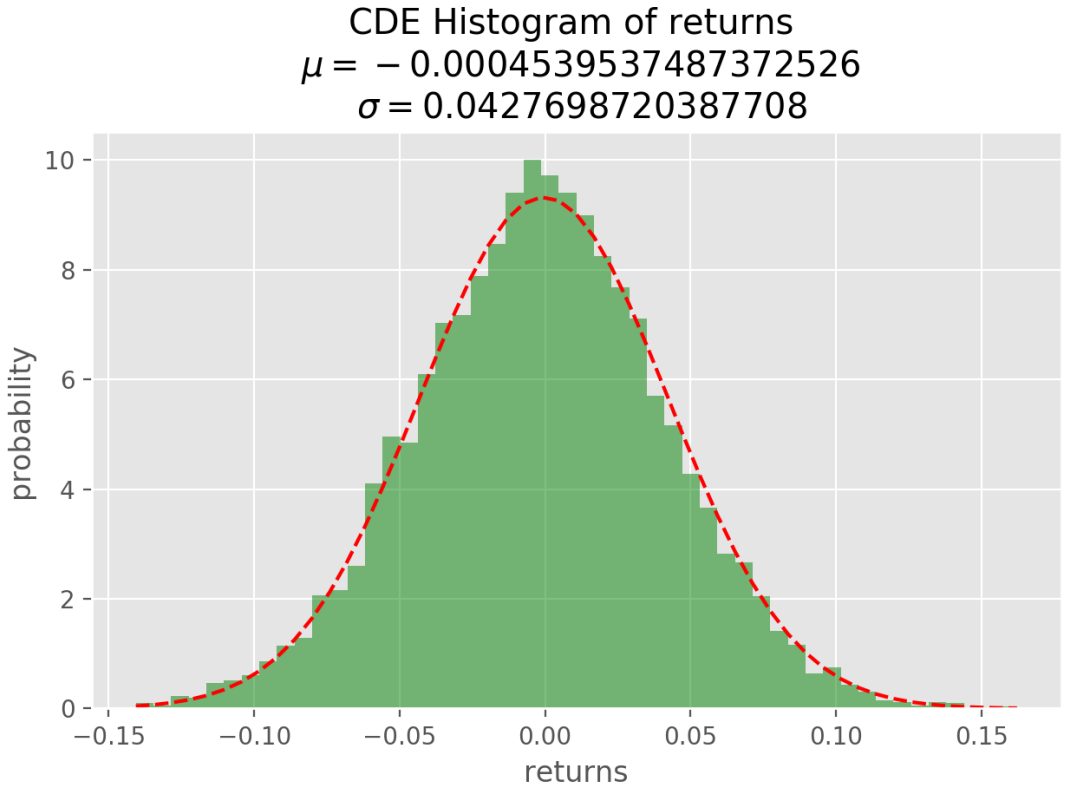


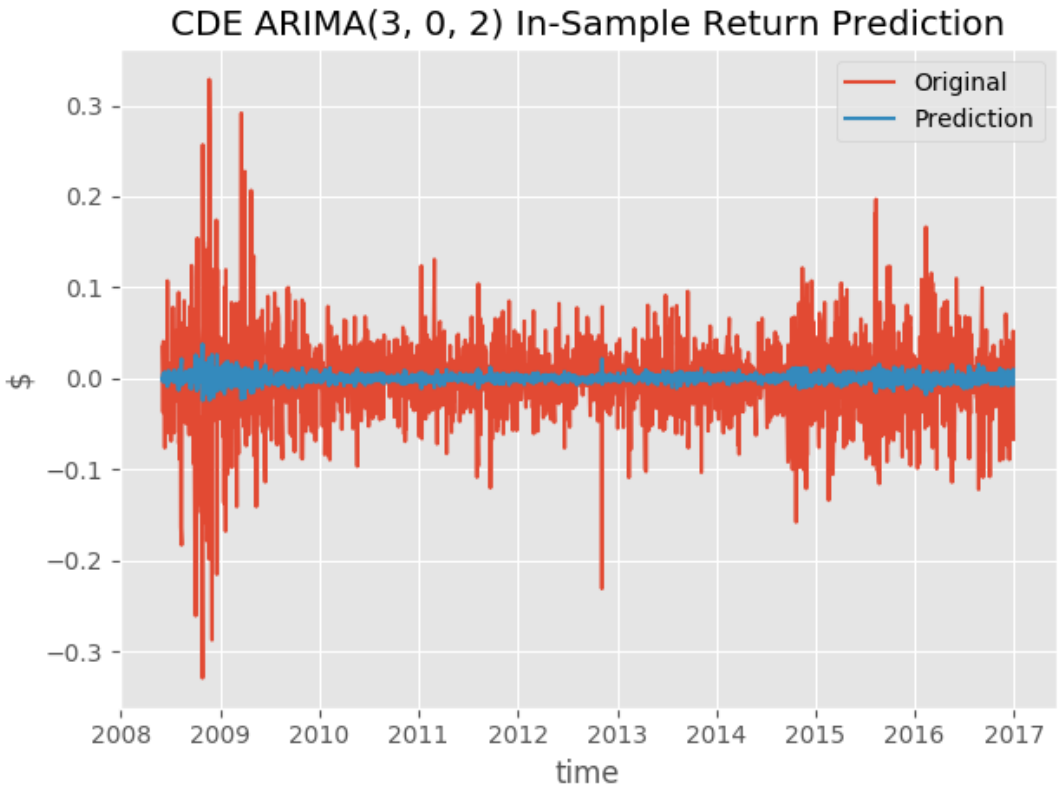


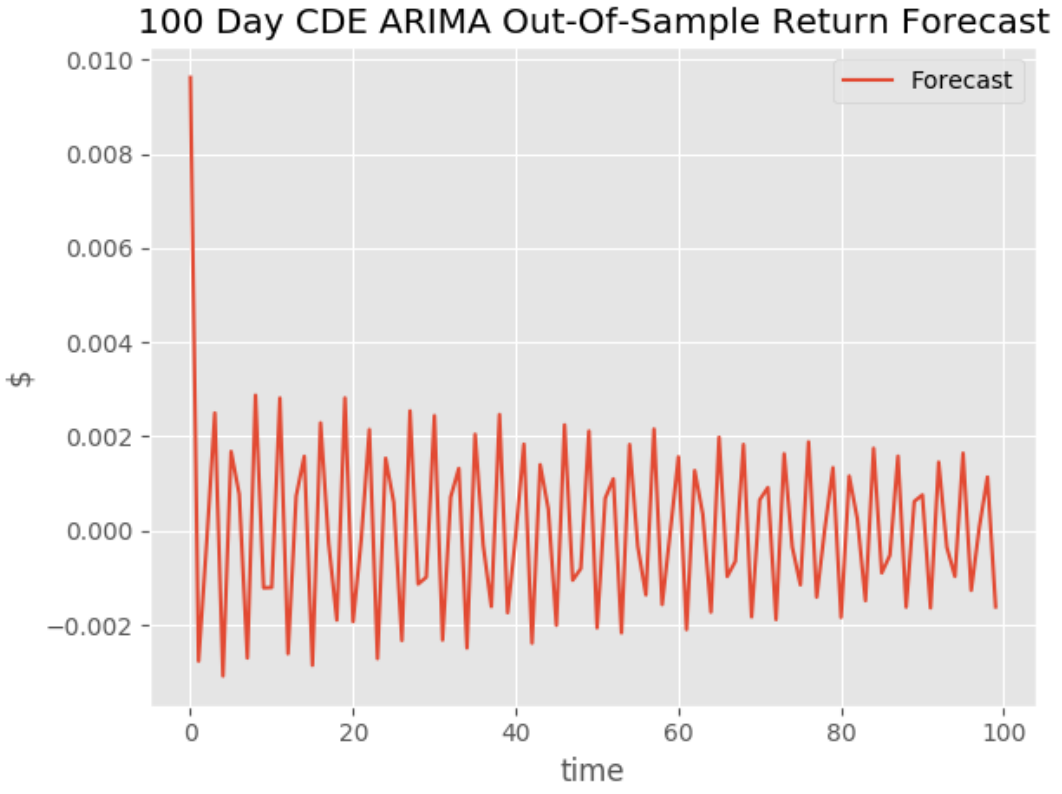

MSFT ARIMA(2, 0, 1) In-Sample Return Prediction
Original
Prediction
0.15
0.10
0.05
0.00
−0.05
−0.10
$
2006
2008
2010
2012
2014
2016
time

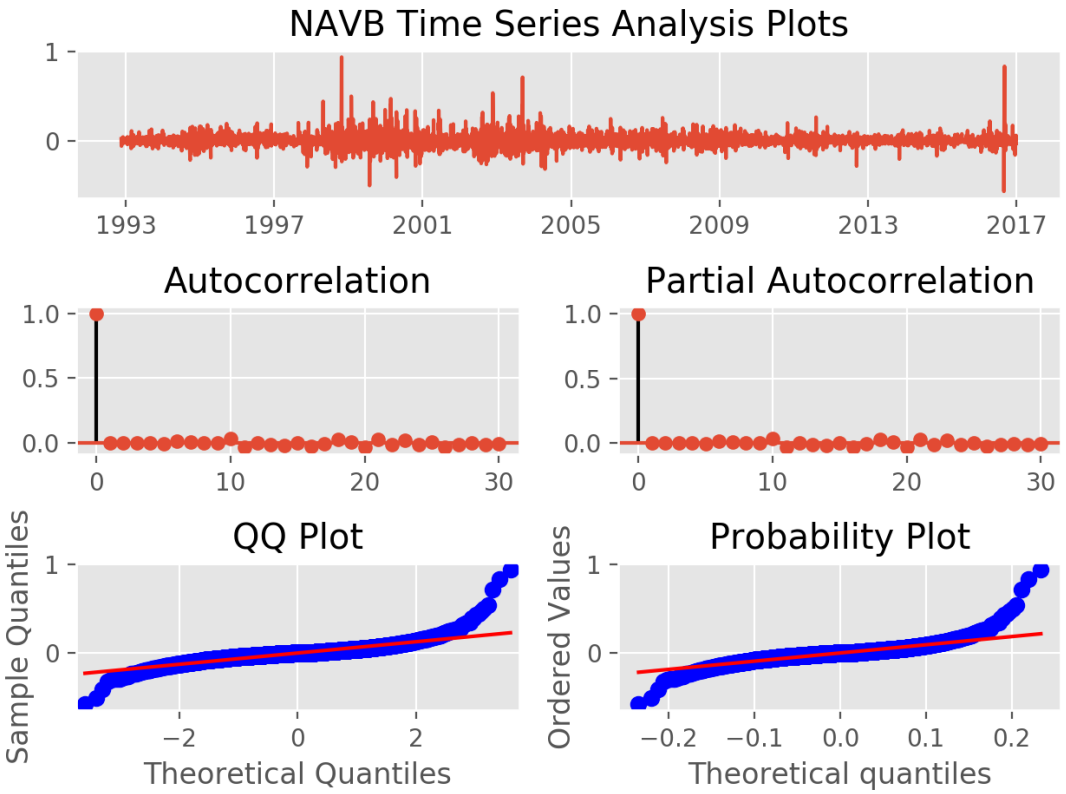

100 Day MSFT ARIMA Out-Of-Sample Return Forecast
Forecast
0.0004
0.0003
0.0002
0.0001
0.0000
$
0
20
40
60
80
100
time

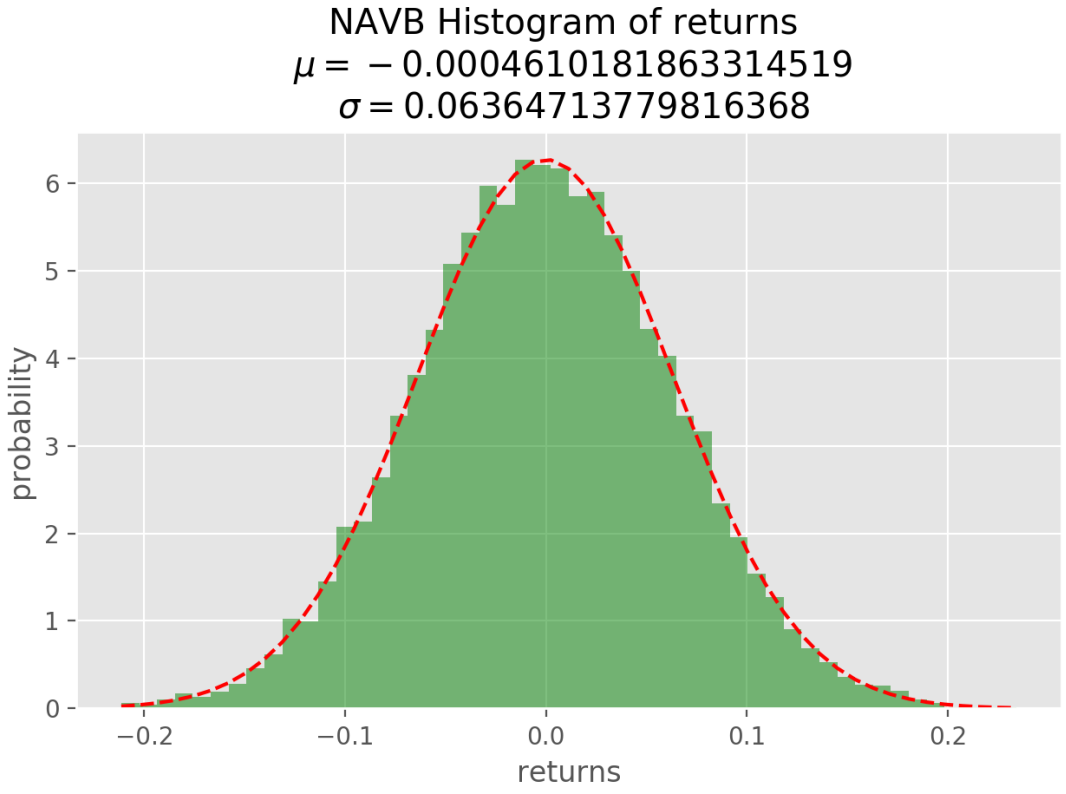

CDE Time Series Analysis Plots
0.25
0.00
−0.25
1991
1995
1999
2003
2007
2011
2015
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles

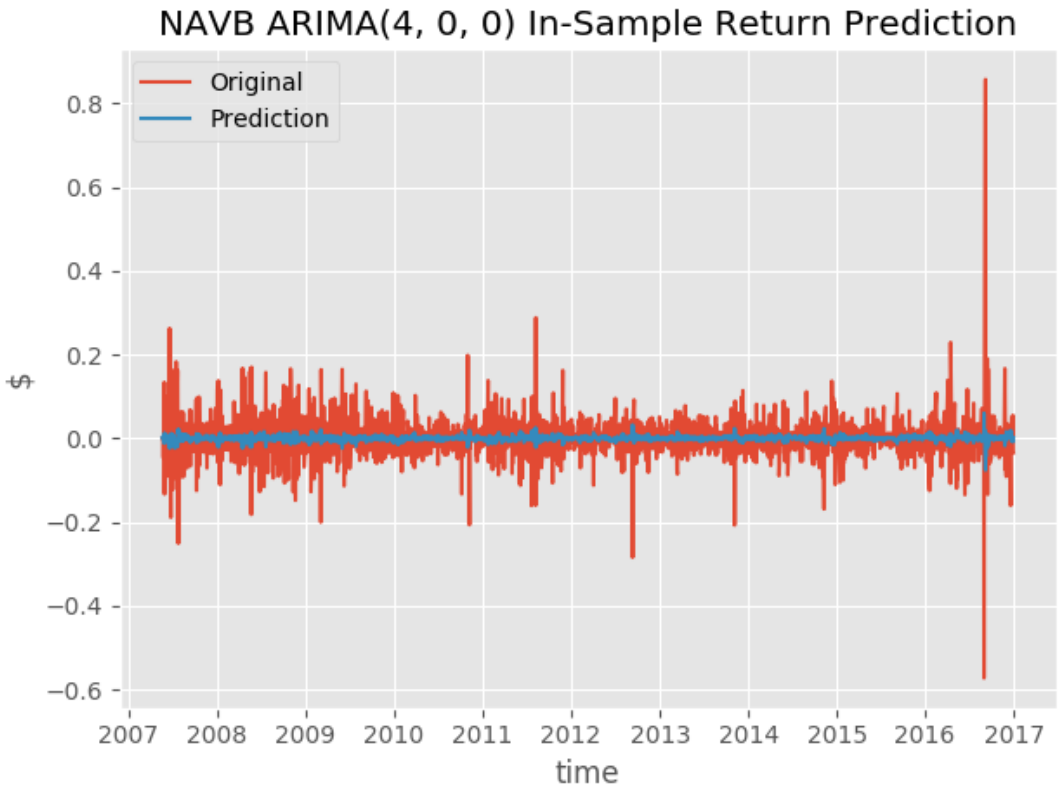

CDE Histogram of returns
$\mu = -0.0004539537487372526$
$\sigma = 0.0427698720387708$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns

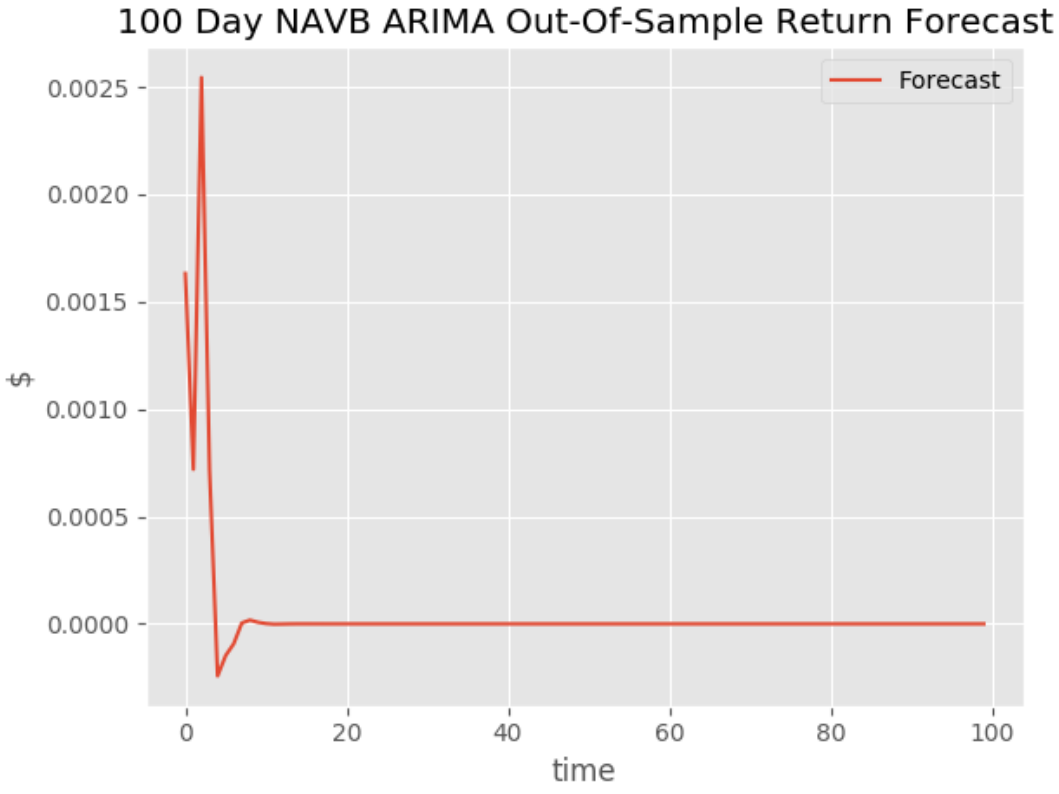

CDE ARIMA(3, 0, 2) In-Sample Return Prediction
Original
Prediction
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
$
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

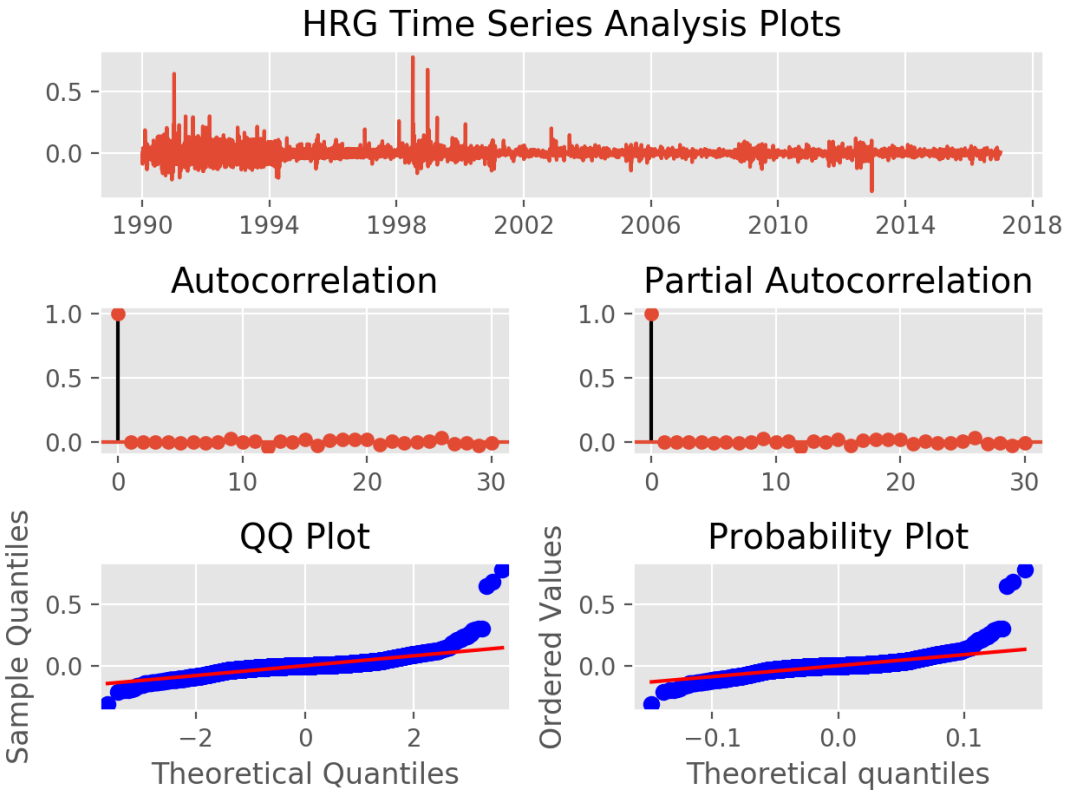

100 Day CDE ARIMA Out-Of-Sample Return Forecast
Forecast
0.010
0.008
0.006
0.004
0.002
0.000
−0.002
$
0
20
40
60
80
100
time

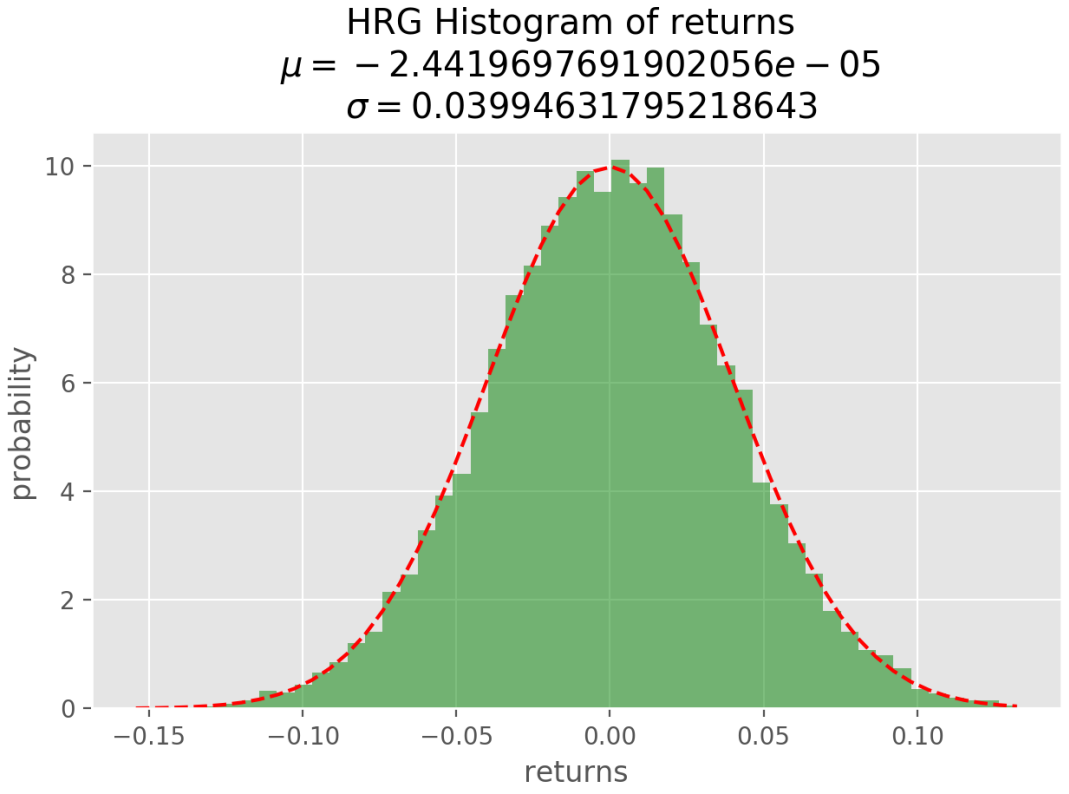

NAVB Time Series Analysis Plots
1
0
1993
1997
2001
2005
2009
2013
2017
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
1
0
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
1
0
−0.2
−0.1
0.0
0.1
0.2
Theoretical quantiles

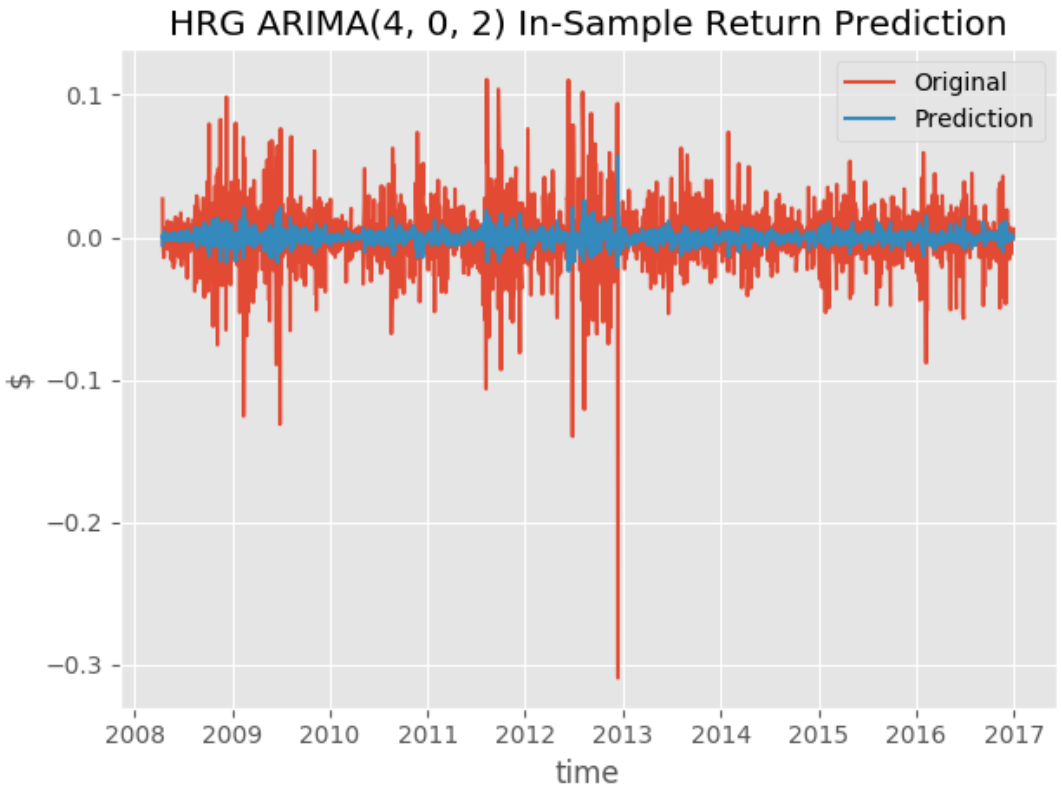

NAVB Histogram of returns
$\mu = -0.0004610181863314519$
$\sigma = 0.06364713779816368$
probability
6
5
4
3
2
1
0
−0.2
−0.1
0.0
0.1
0.2
returns

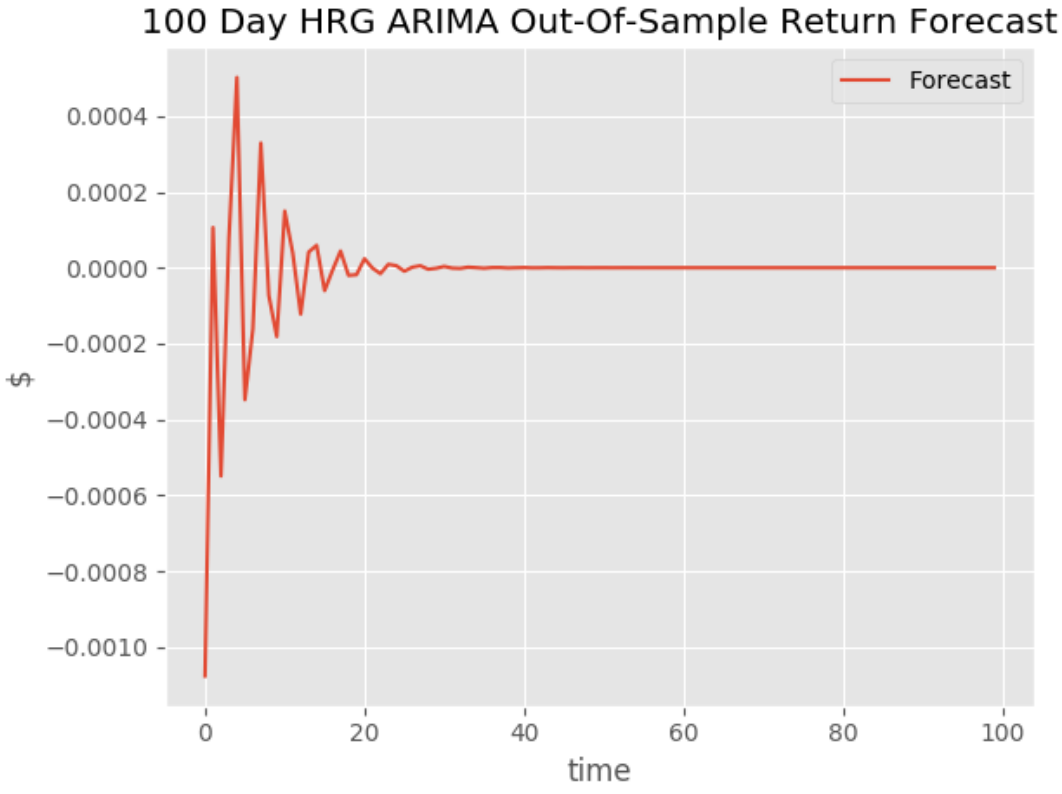

NAVB ARIMA(4, 0, 0) In-Sample Return Prediction
Original
Prediction
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
$
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time

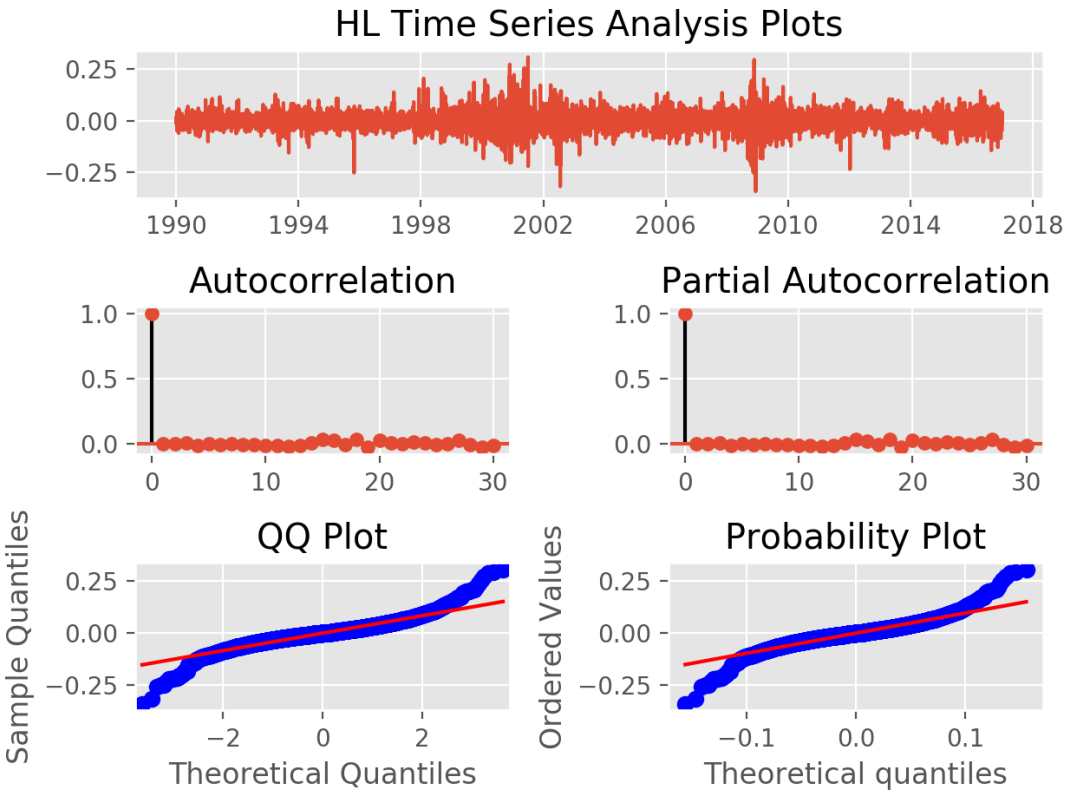

100 Day NAVB ARIMA Out-Of-Sample Return Forecast
Forecast
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
$
0
20
40
60
80
100
time

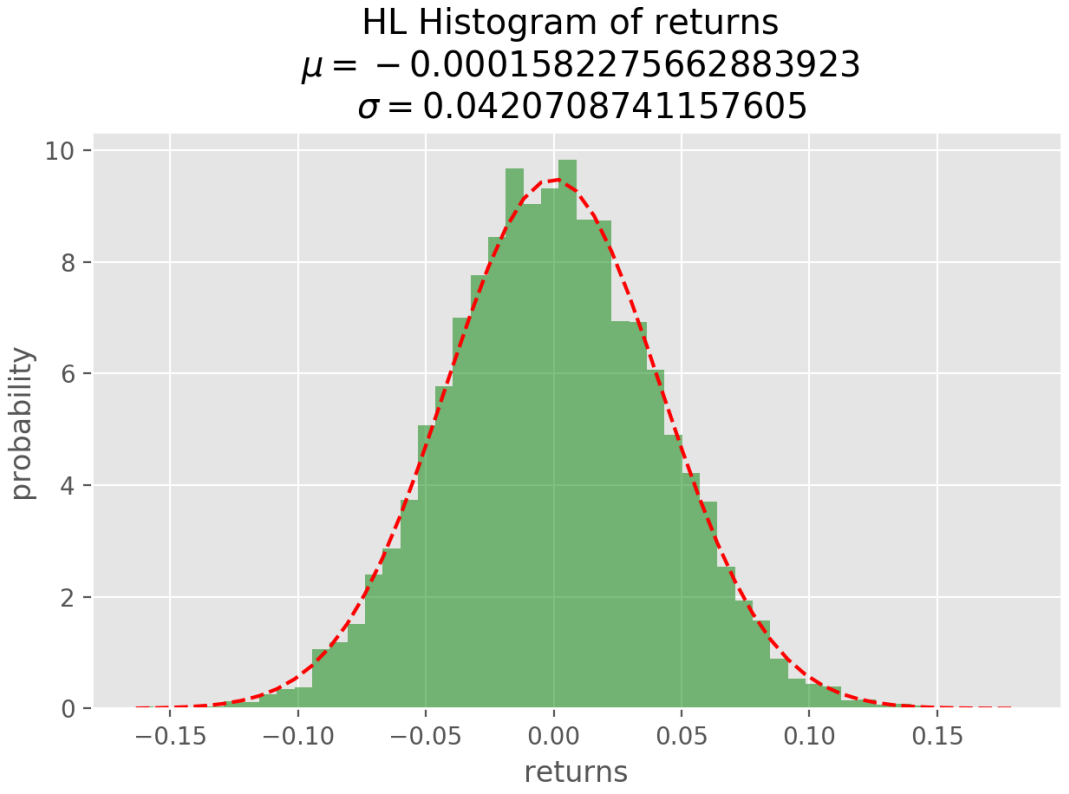

HRG Time Series Analysis Plots
0.5
0.0
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Probability Plot
Sample Quantiles
Ordered Values
−2
0
2
Theoretical Quantiles
−0.1
0.0
0.1
Theoretical quantiles

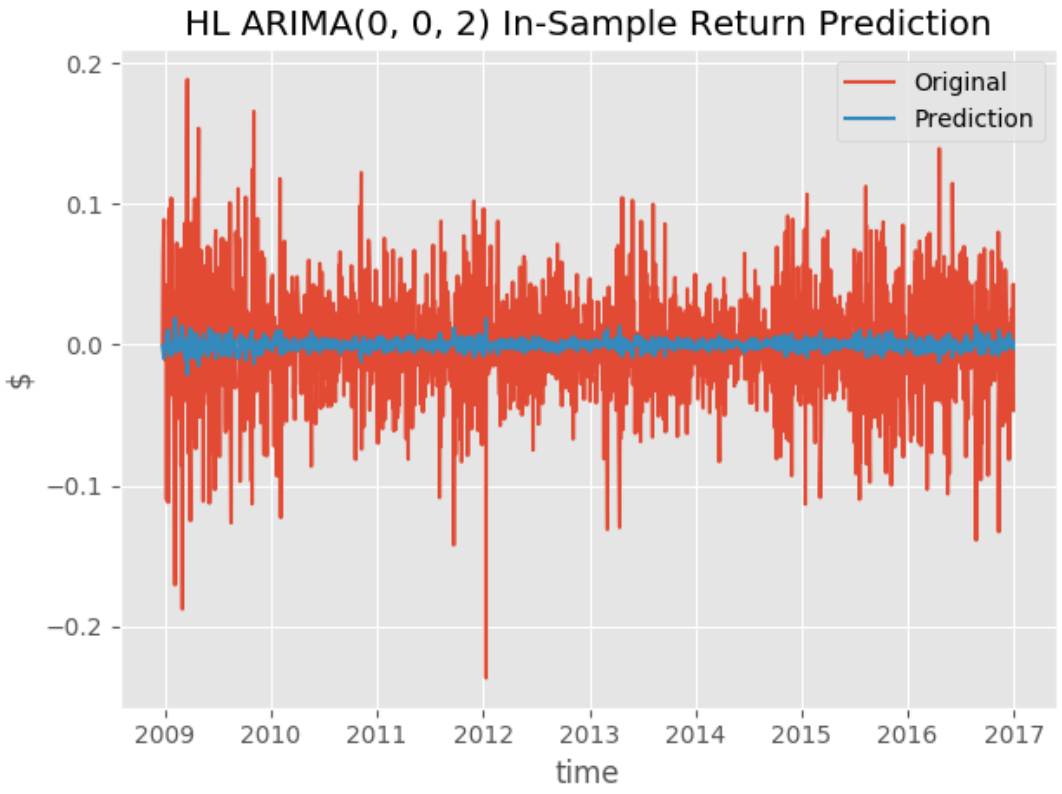

HRG Histogram of returns
$\mu = -2.4419697691902056e-05$
$\sigma = 0.03994631795218643$
probability
returns
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10

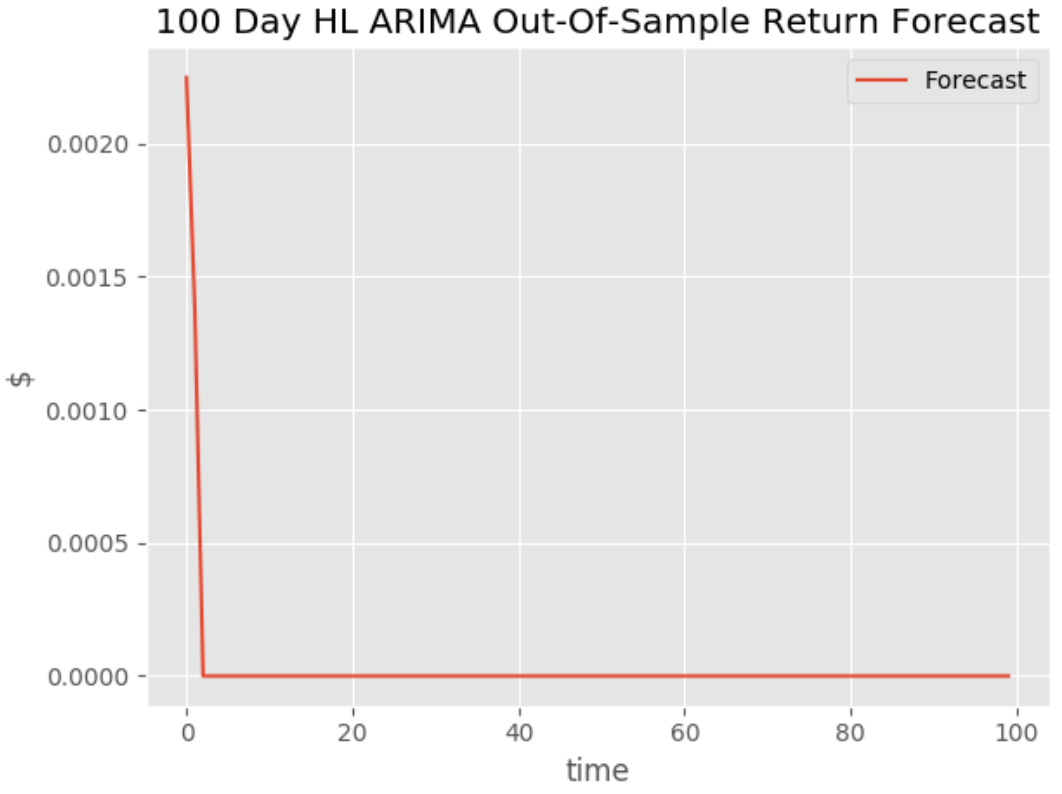

HRG ARIMA(4, 0, 2) In-Sample Return Prediction
Original
Prediction
0.1
0.0
−0.1
−0.2
−0.3
$
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
time


100 Day HRG ARIMA Out-Of-Sample Return Forecast
Forecast
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
$
0
20
40
60
80
100
time


HL Time Series Analysis Plots
0.25
0.00
−0.25
1990
1994
1998
2002
2006
2010
2014
2018
Autocorrelation
1.0
0.5
0.0
0
10
20
30
Partial Autocorrelation
1.0
0.5
0.0
0
10
20
30
QQ Plot
Sample Quantiles
0.25
0.00
−0.25
−2
0
2
Theoretical Quantiles
Probability Plot
Ordered Values
0.25
0.00
−0.25
−0.1
0.0
0.1
Theoretical quantiles


HL Histogram of returns
$\mu = -0.0001582275662883923$
$\sigma = 0.0420708741157605$
probability
10
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
returns


HL ARIMA(0, 0, 2) In-Sample Return Prediction
Original
Prediction
0.2
0.1
0.0
−0.1
−0.2
$
2009
2010
2011
2012
2013
2014
2015
2016
2017
time


100 Day HL ARIMA Out-Of-Sample Return Forecast
Forecast
0.0020
0.0015
0.0010
0.0005
0.0000
$
0
20
40
60
80
100
time

4.2 Machine Learning

4.2.1 Classification

4.2.1.1 Decision Tree

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	131	547	190
CDE	0.79	565	144	504	133
NAVB	0.76	592	165	331	126
HRG	0.75	553	210	462	135
HL	0.79	603	135	475	147

4.2.1.2 Boosted Decision Tree

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	130	548	190
CDE	0.79	564	143	505	134
NAVB	0.76	588	164	332	130
HRG	0.75	542	191	481	146
HL	0.79	602	132	478	149

4.2.1.3 Support Vector Machine (SVM)

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	130	548	190
CDE	0.79	564	143	505	134
NAVB	0.76	593	169	327	125
HRG	0.75	542	191	481	146
HL	0.81	579	98	512	171

4.2.1.4 Random Forest

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	131	547	190
CDE	0.79	566	145	503	132
NAVB	0.76	590	164	332	128
HRG	0.75	554	210	462	134
HL	0.81	581	101	509	169

4.2.1.5 K-Nearest Neighbour

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.76	489	130	548	194
CDE	0.80	574	147	501	126
NAVB	0.75	573	161	335	145
HRG	0.74	560	233	439	128
HL	0.81	581	101	509	169

4.2.1.6 Gaussian Naive Bayes

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.73	478	168	510	205
CDE	0.76	555	187	461	143
NAVB	0.74	553	153	343	165
HRG	0.73	491	170	502	197
HL	0.77	590	157	453	160

4.2.1.7 Bernoulli Naive Bayes

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.73	479	169	509	204
CDE	0.76	558	187	461	140
NAVB	0.74	553	153	343	165
HRG	0.73	492	170	502	196
HL	0.77	590	158	452	160

4.2.1.8 Neural Network

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	685	181	787	258
CDE	0.79	616	156	542	152
NAVB	0.76	691	183	403	153
HRG	0.74	690	269	542	164
HL	0.77	1059	270	848	164

4.2.1.9 Logistic Regression

Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	130	548	190
CDE	0.79	564	143	505	134
NAVB	0.76	588	164	332	130
HRG	0.75	542	191	481	146
HL	0.79	601	132	478	149

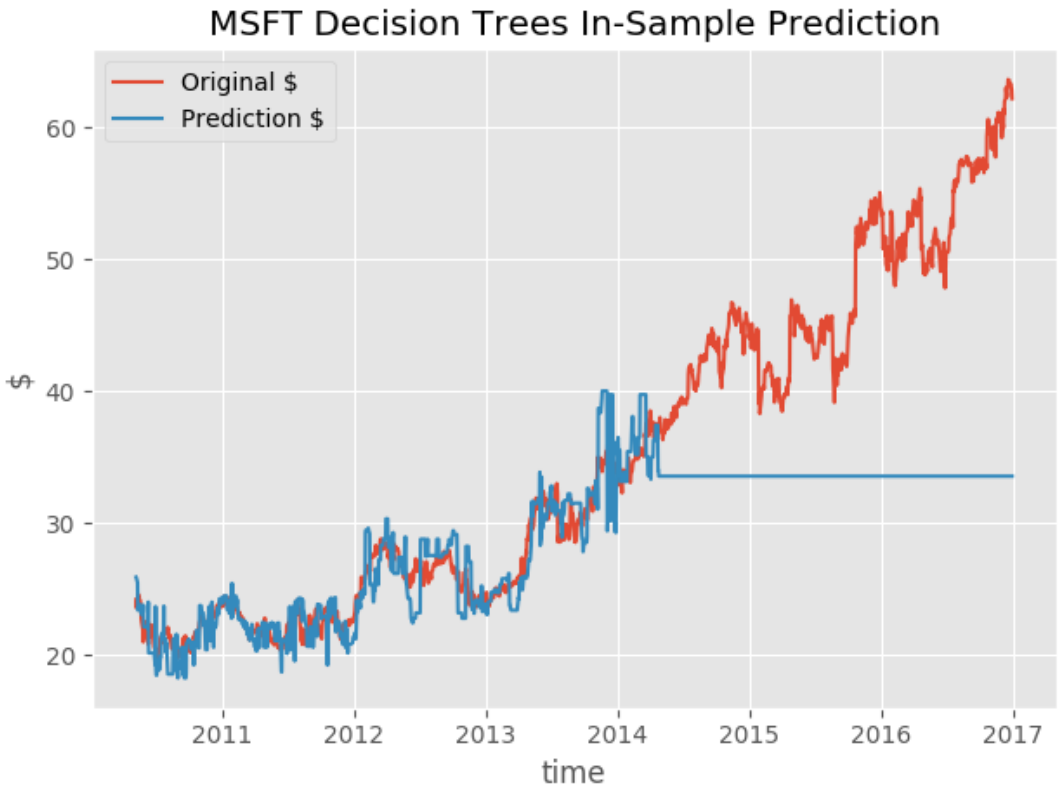
4.2.1.10 Stochastic Gradient Descent

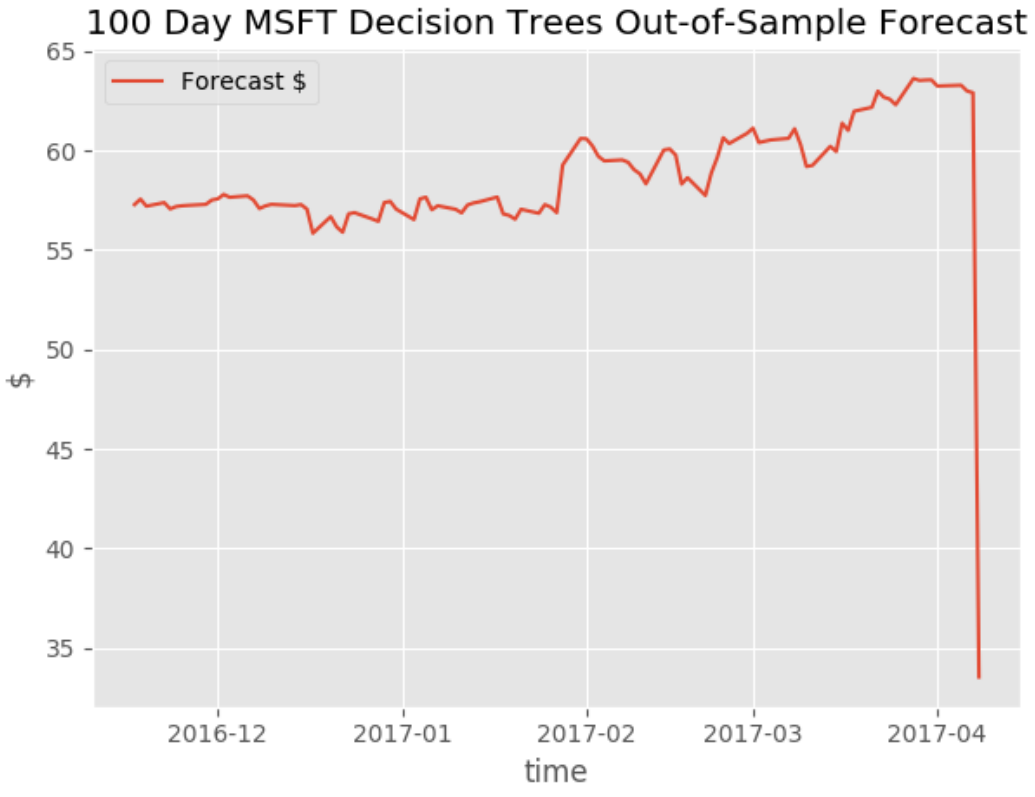
Ticker	Precision	True Negatives	False Negatives	True Positives	False Positives
MSFT	0.77	493	130	548	190
CDE	0.76	462	106	542	236
NAVB	0.59	1032	707	118	80
HRG	0.72	761	225	1059	518
HL	0.80	535	83	527	215

4.2.2 Regression

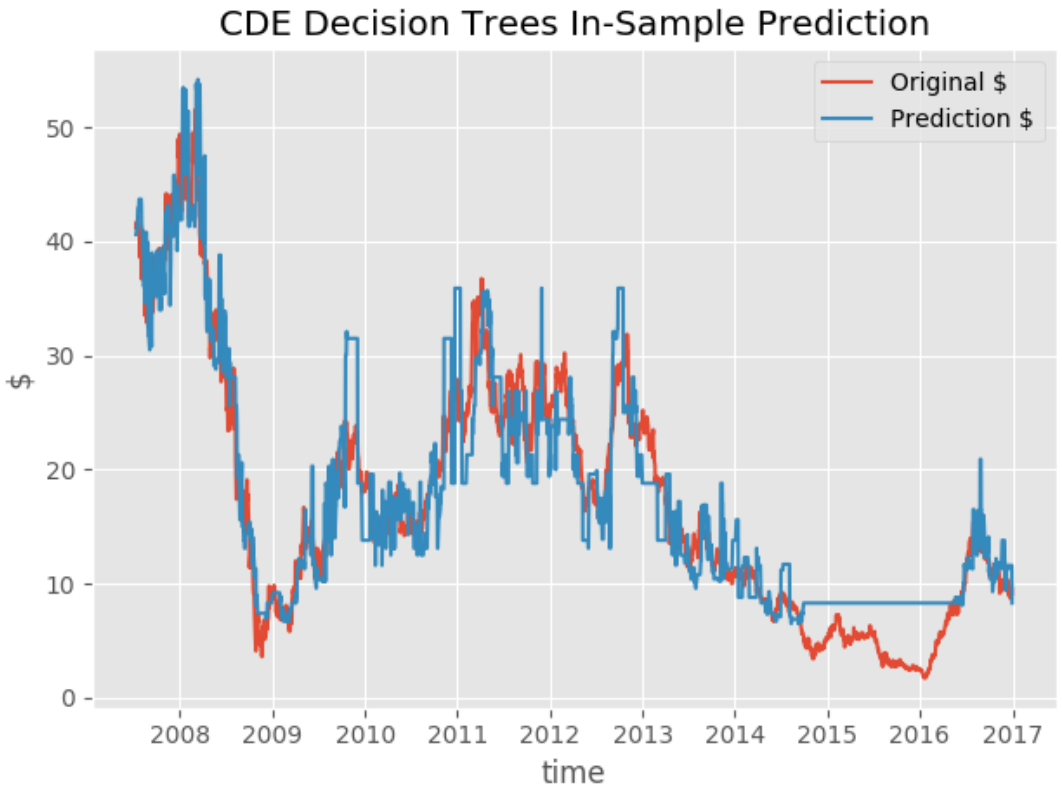
4.2.2.1 Decision Tree

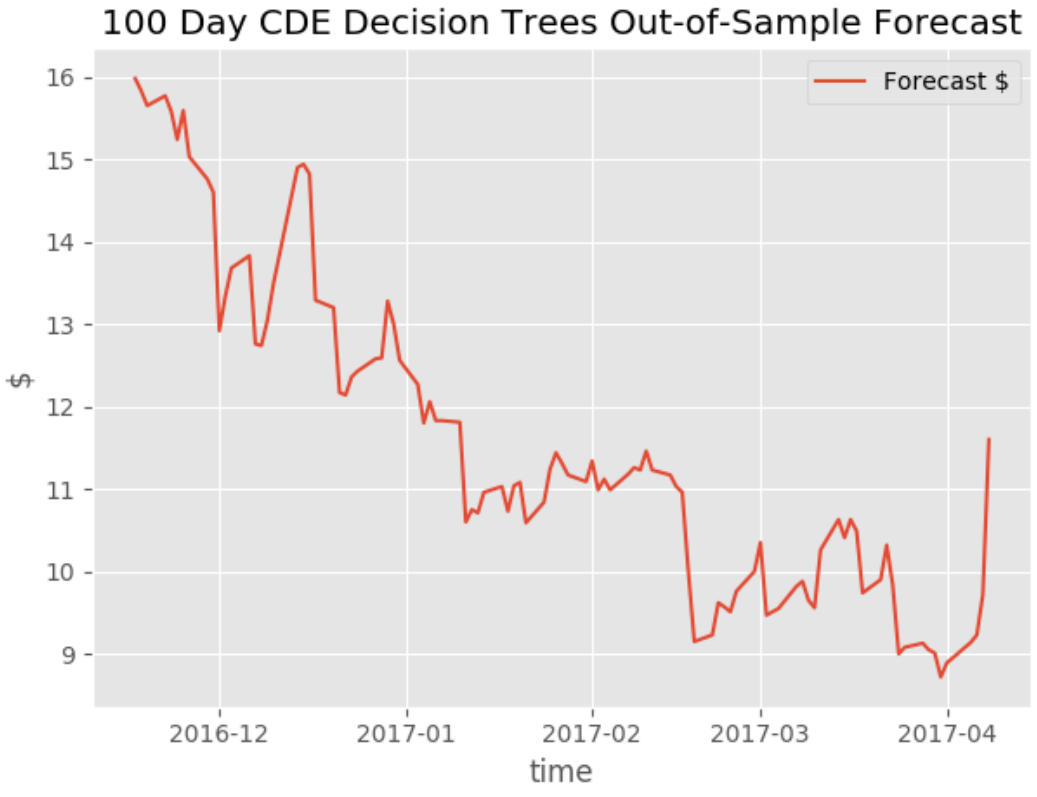
MSFT scored a mean absolute error regression loss of 5.606, and a coefficient of determination of 0.458.



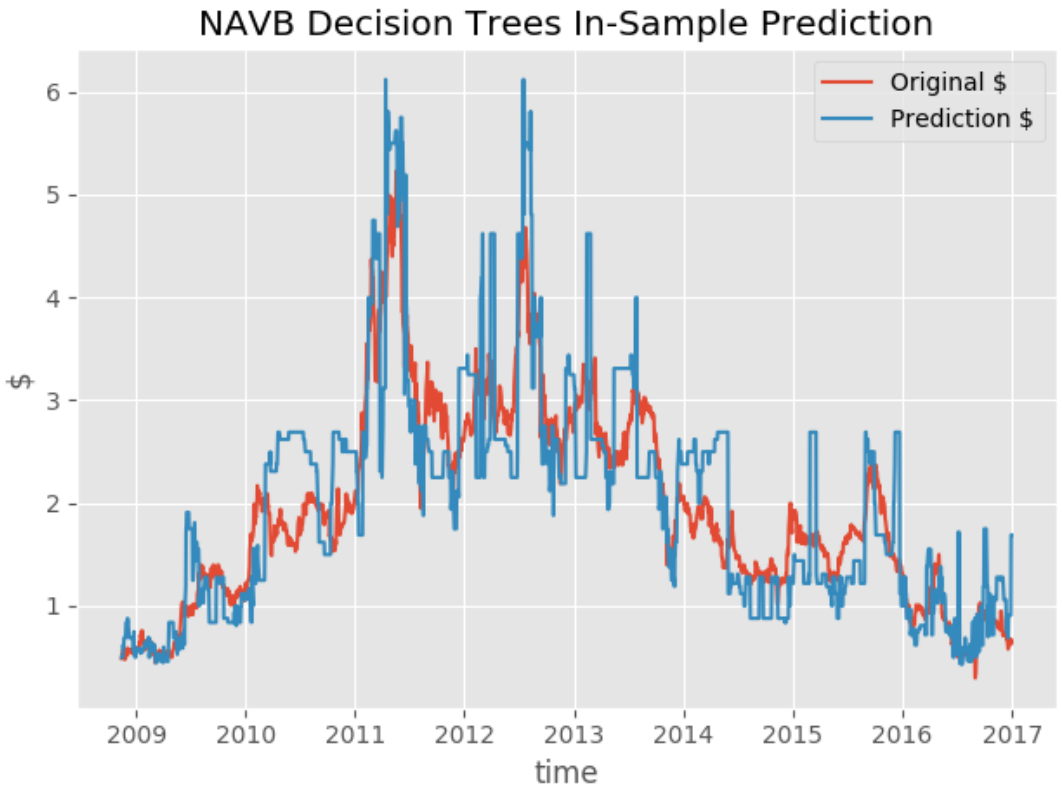


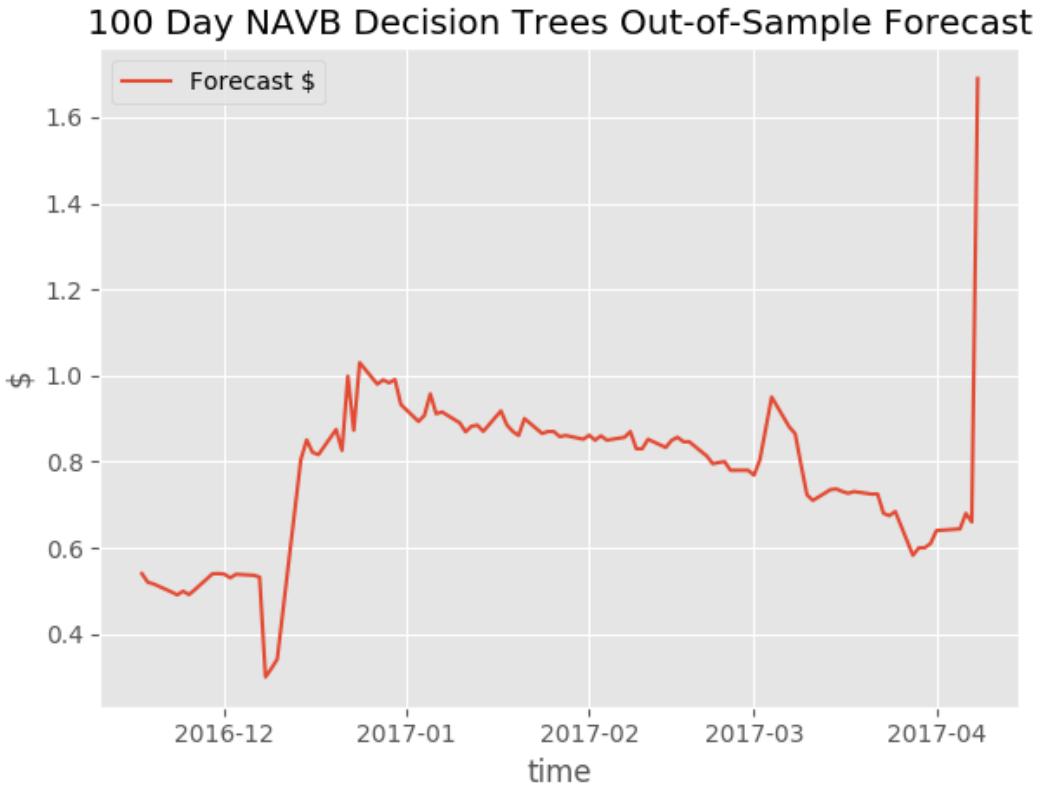
CDE scored a mean absolute error regression loss of 2.906, and a coefficient of determination of 0.816.



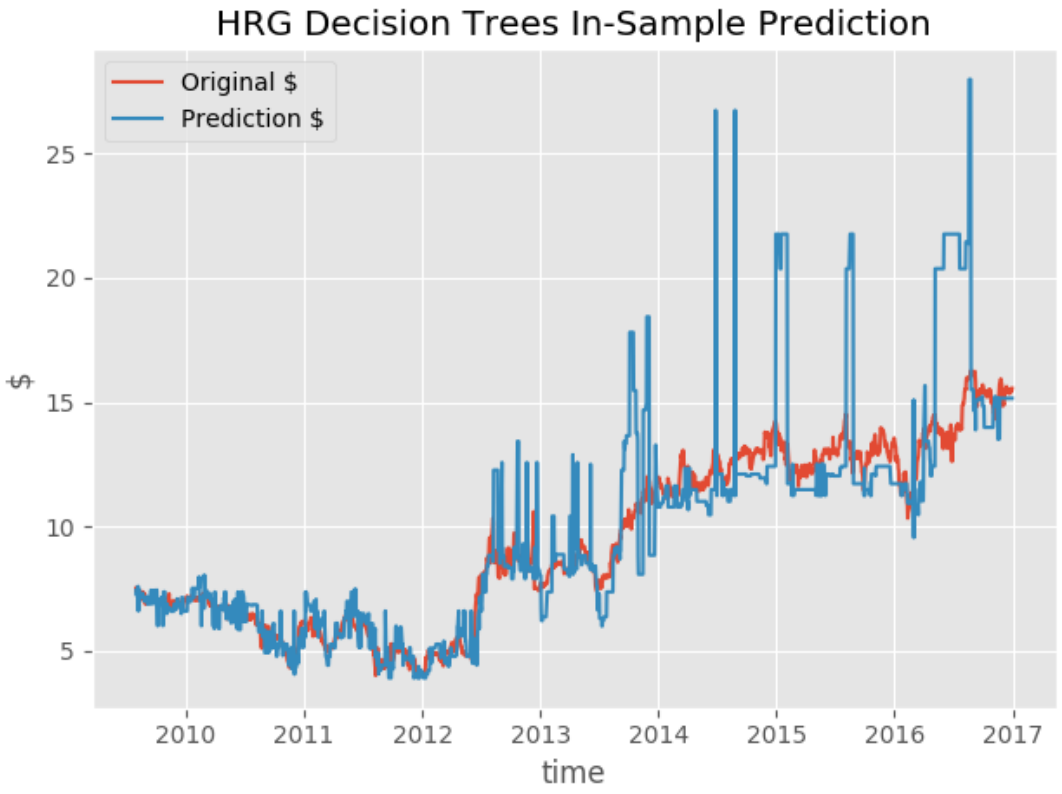


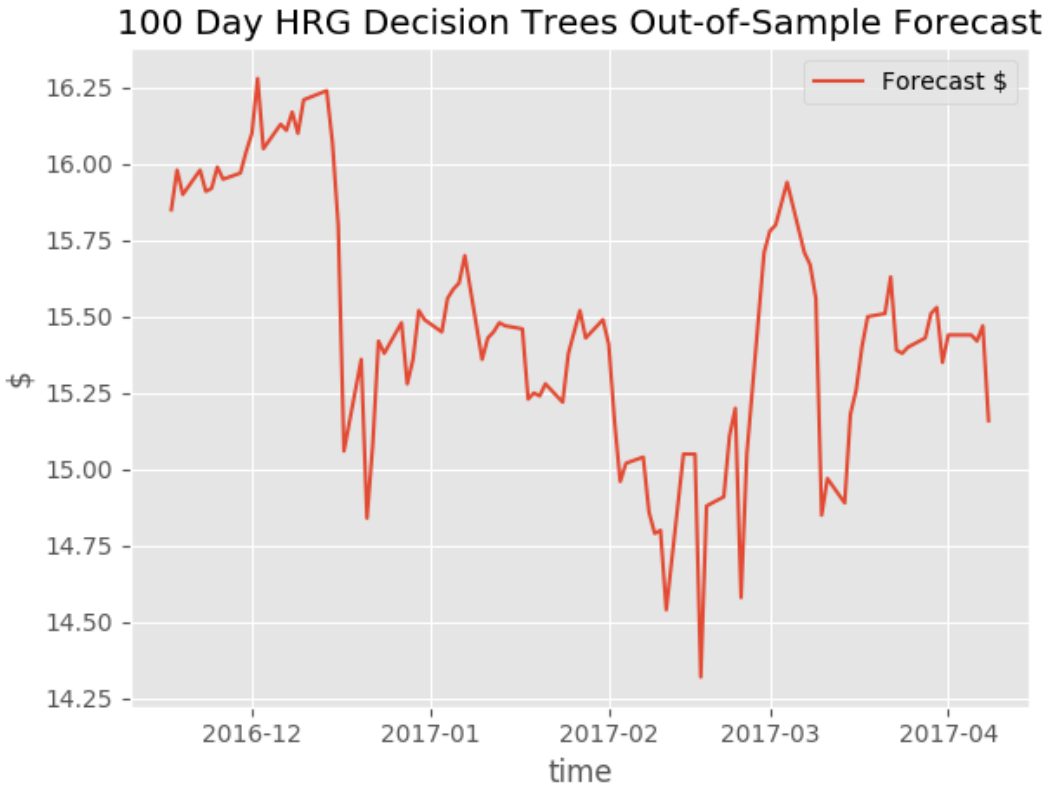
NAVB scored a mean absolute error regression loss of 0.422, and a coefficient of determination of 0.693.



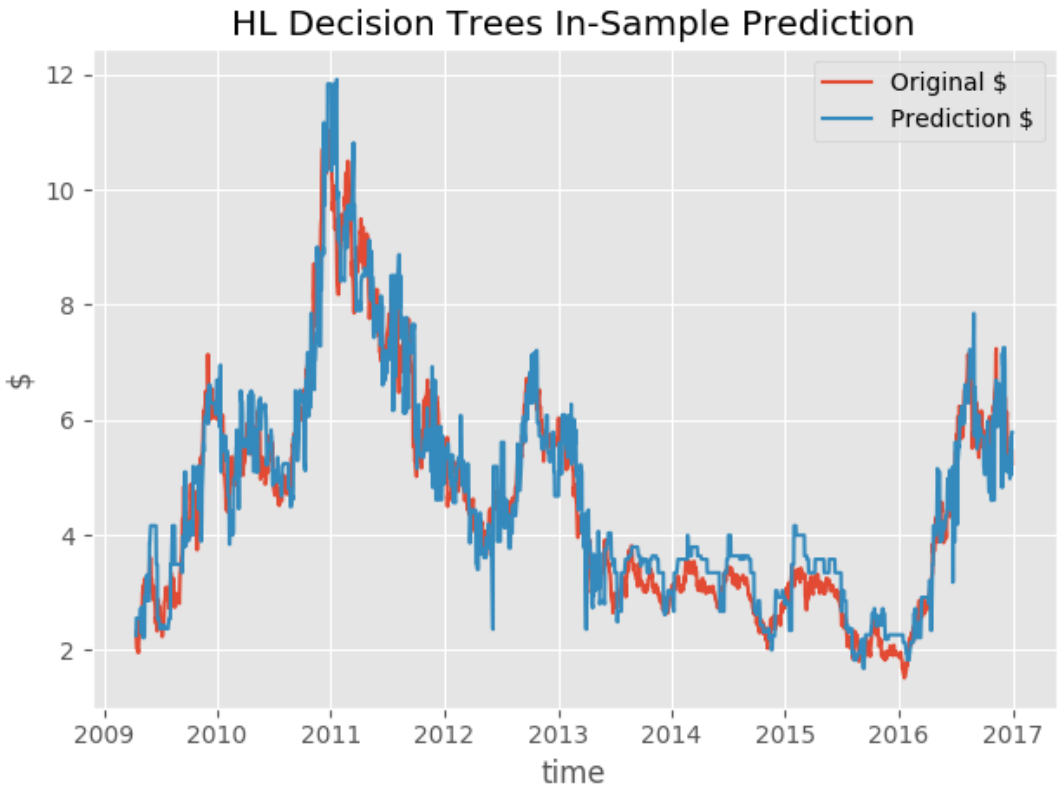


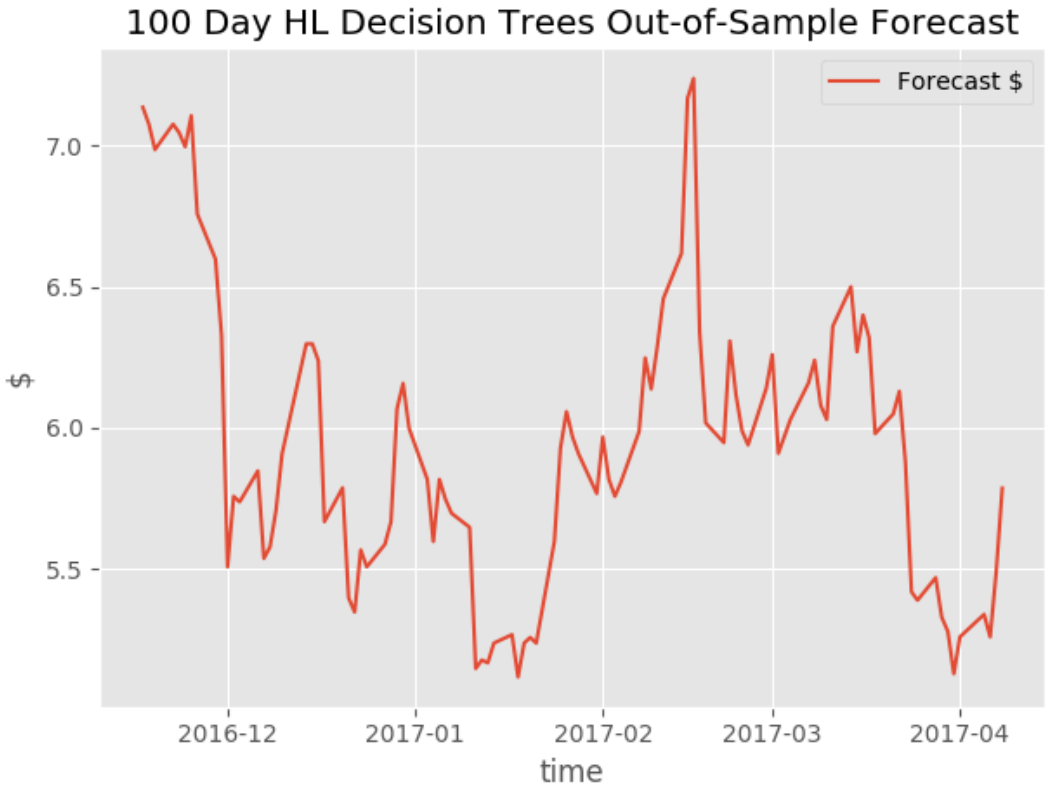
HRG scored a mean absolute error regression loss of 1.415, and a coefficient of determination of 0.464.





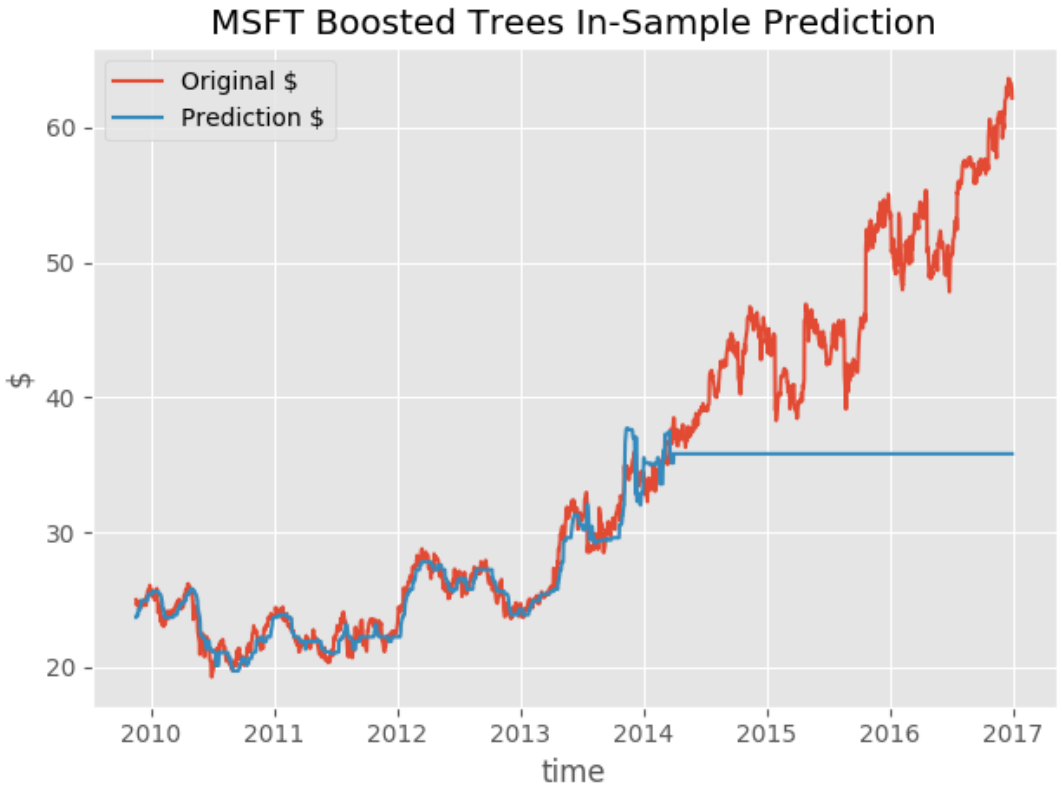
HL scored a mean absolute error regression loss of 0.506, and a coefficient of determination of 0.923.

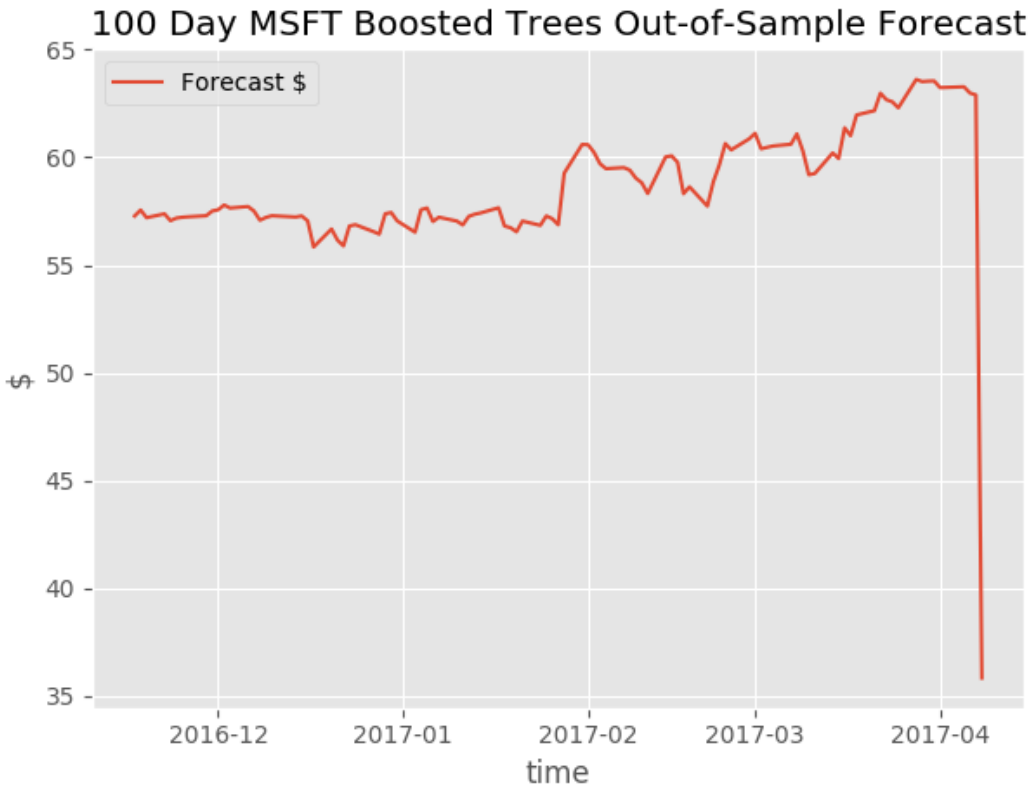

HL Decision Trees In-Sample Prediction
Original $
Prediction $
12
10
8
6
4
2
$
2009
2010
2011
2012
2013
2014
2015
2016
2017
time


100 Day HL Decision Trees Out-of-Sample Forecast
Forecast $
7.0
6.5
6.0
5.5
$
2016-12
2017-01
2017-02
2017-03
2017-04
time

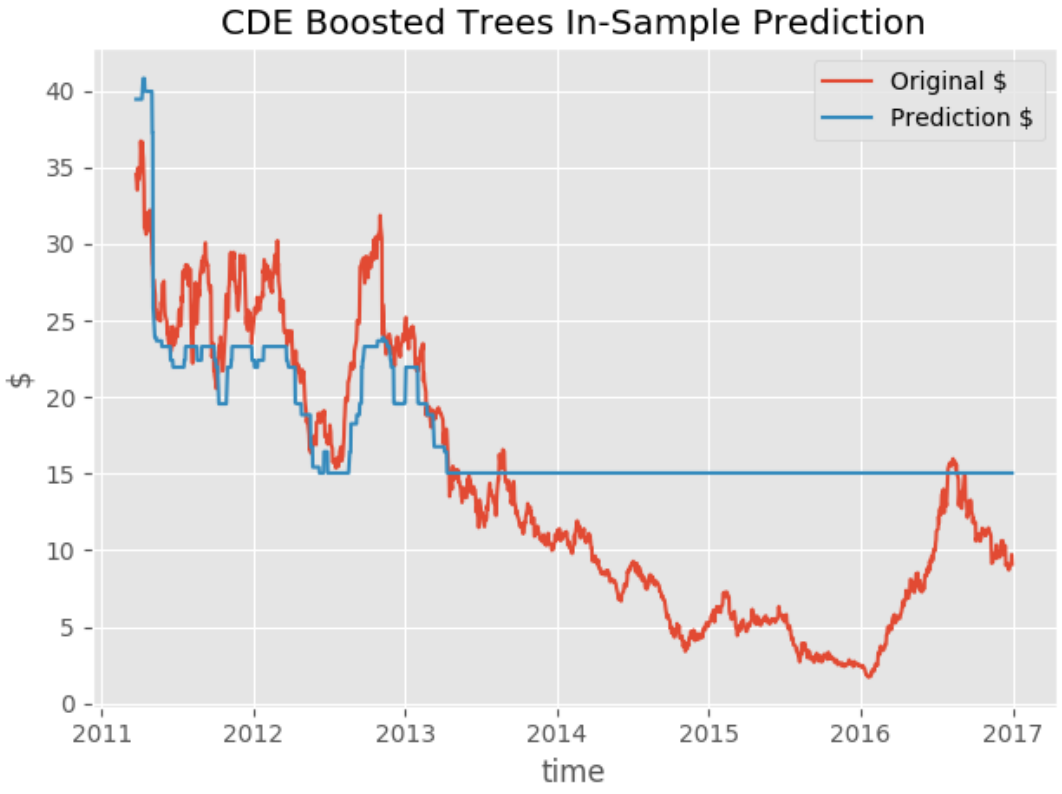
4.2.2.2 Boosted Decision Tree

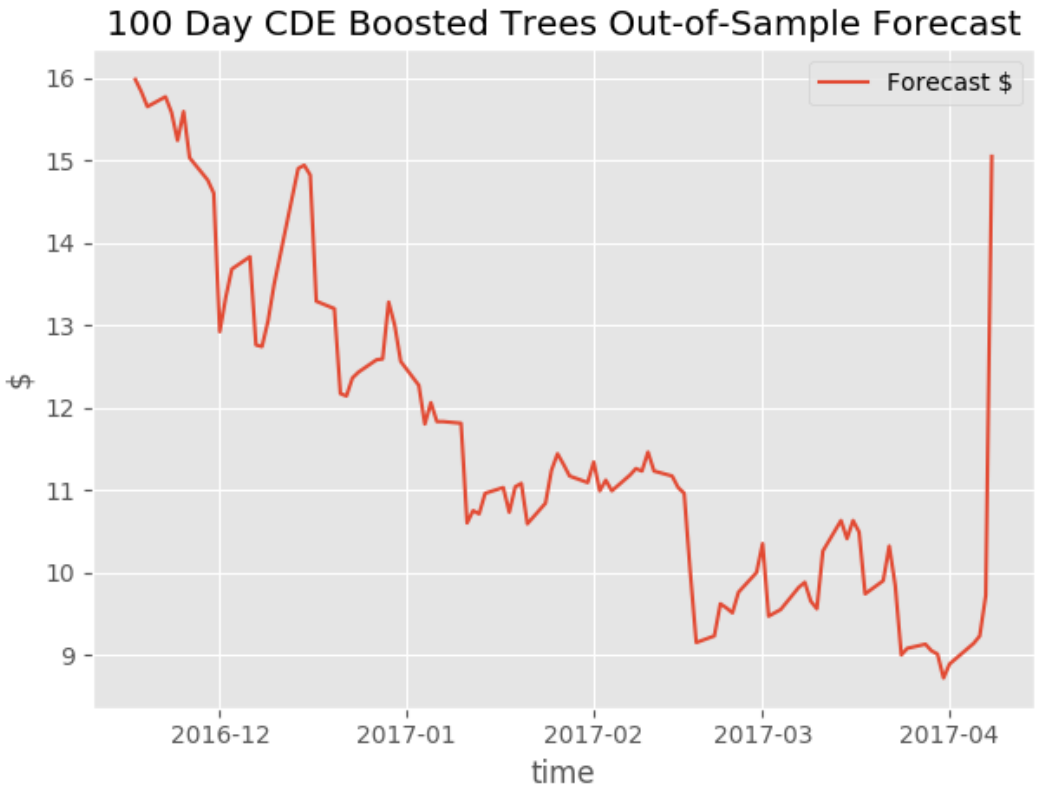
MSFT scored a mean absolute error regression loss of 4.426, and a coefficient of determination of 0.580.



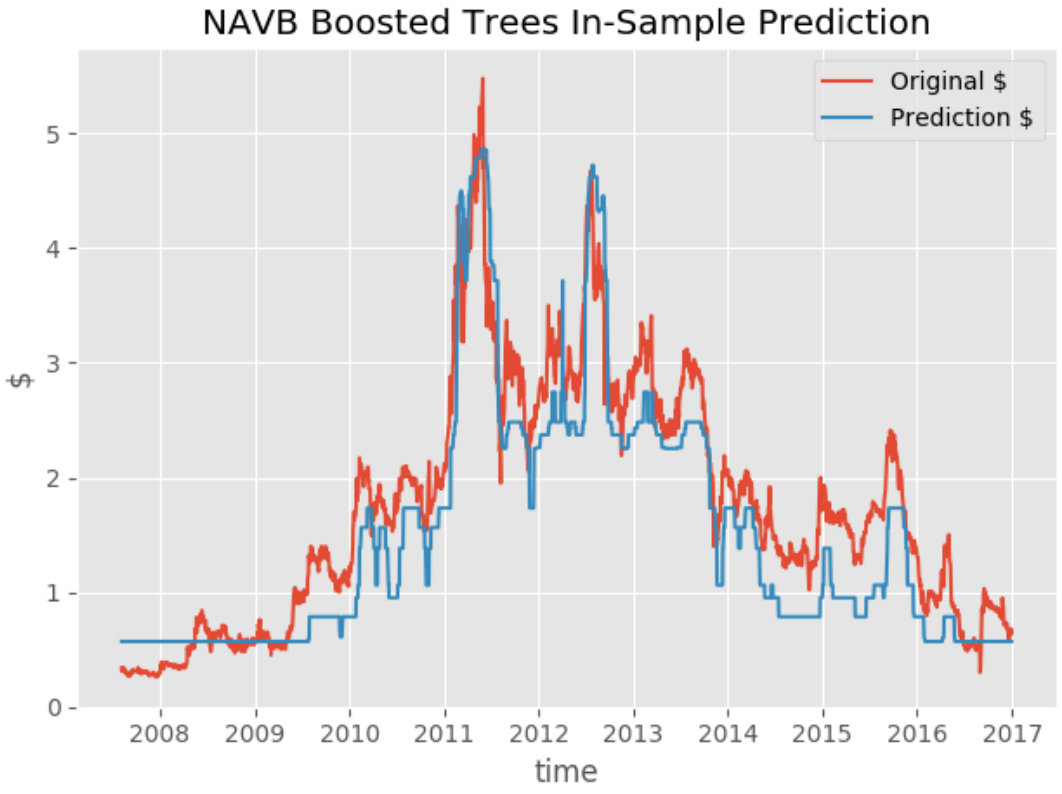


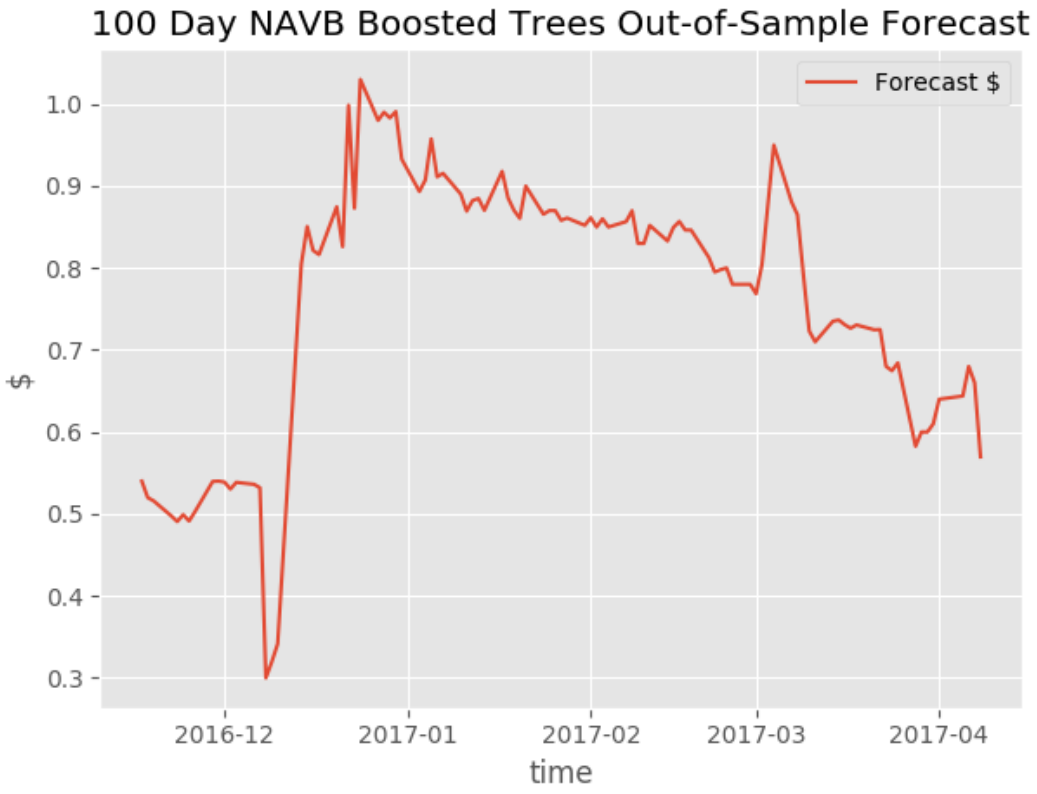
CDE scored a mean absolute error regression loss of 5.452, and a coefficient of determination of 0.581.



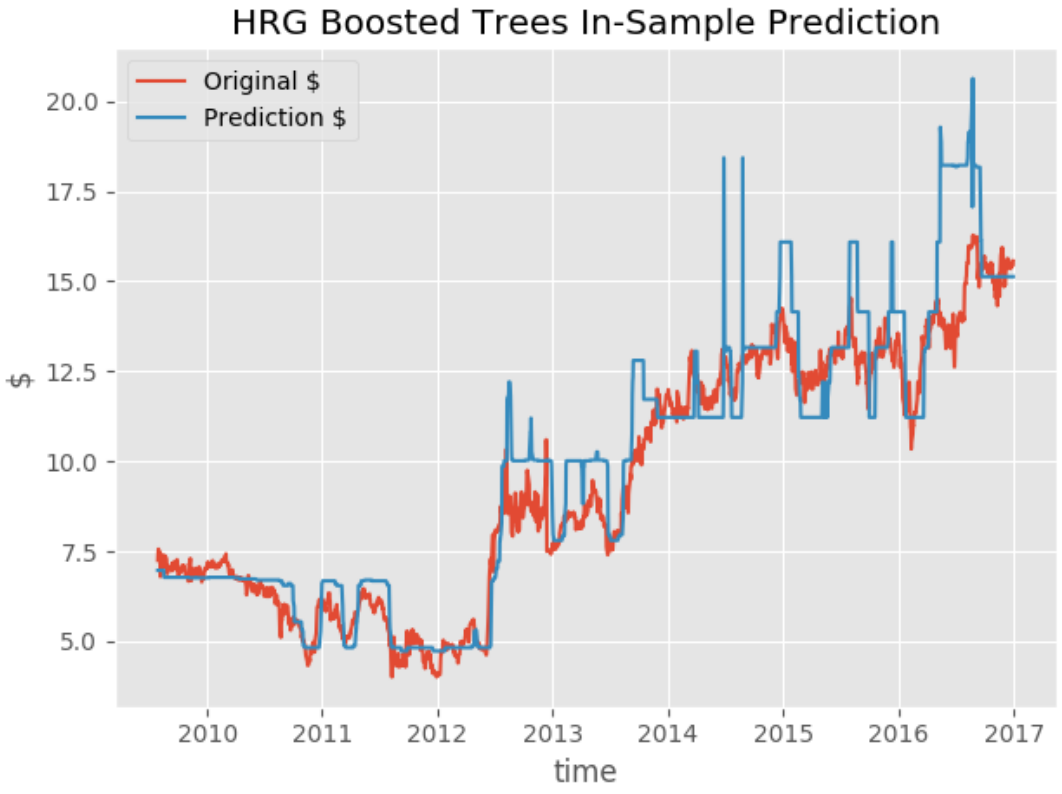


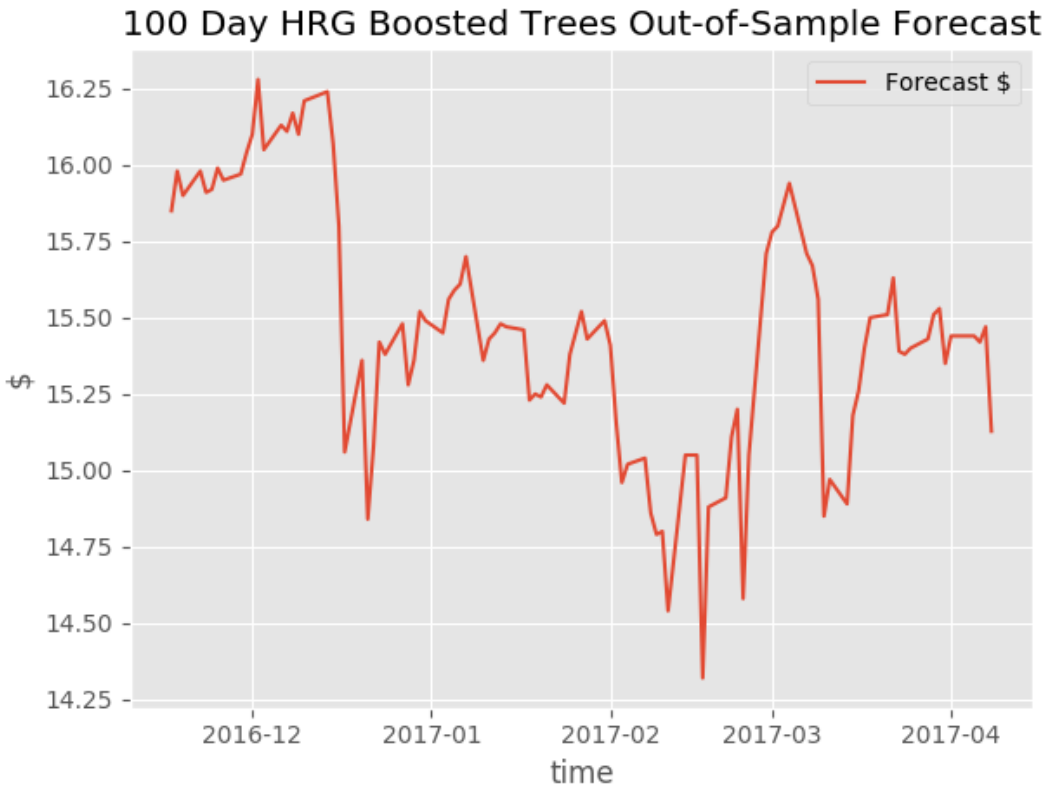
NAVB scored a mean absolute error regression loss of 0.579, and a coefficient of determination of 0.509.



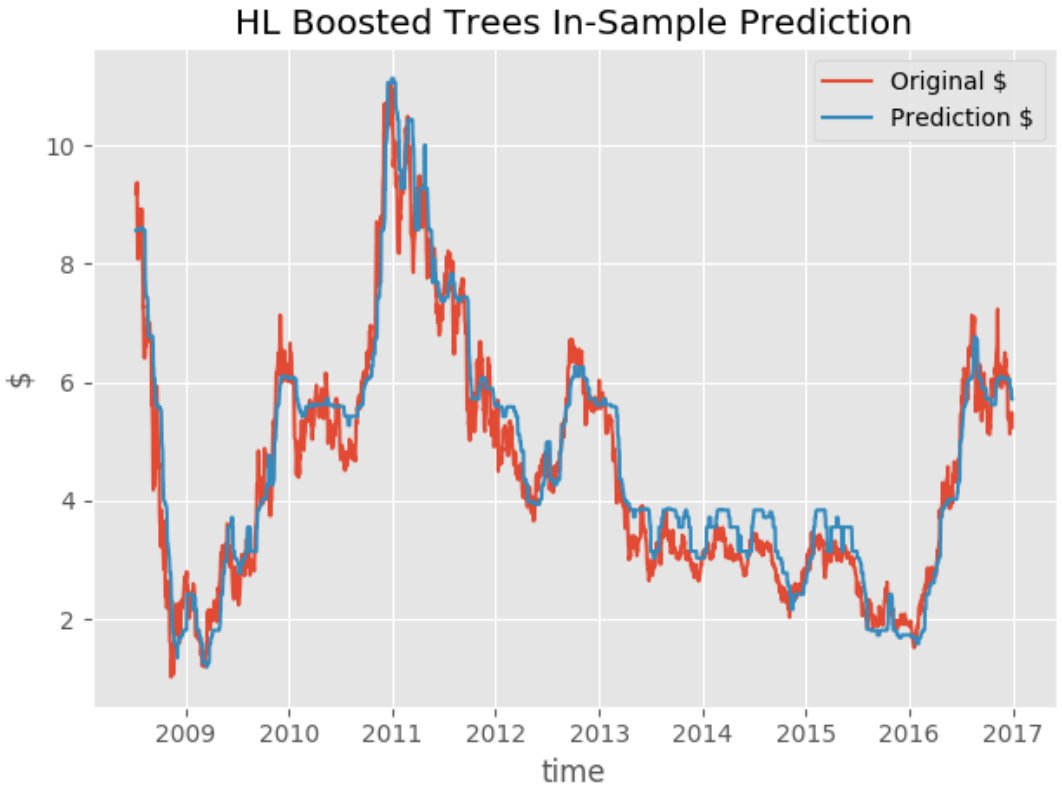


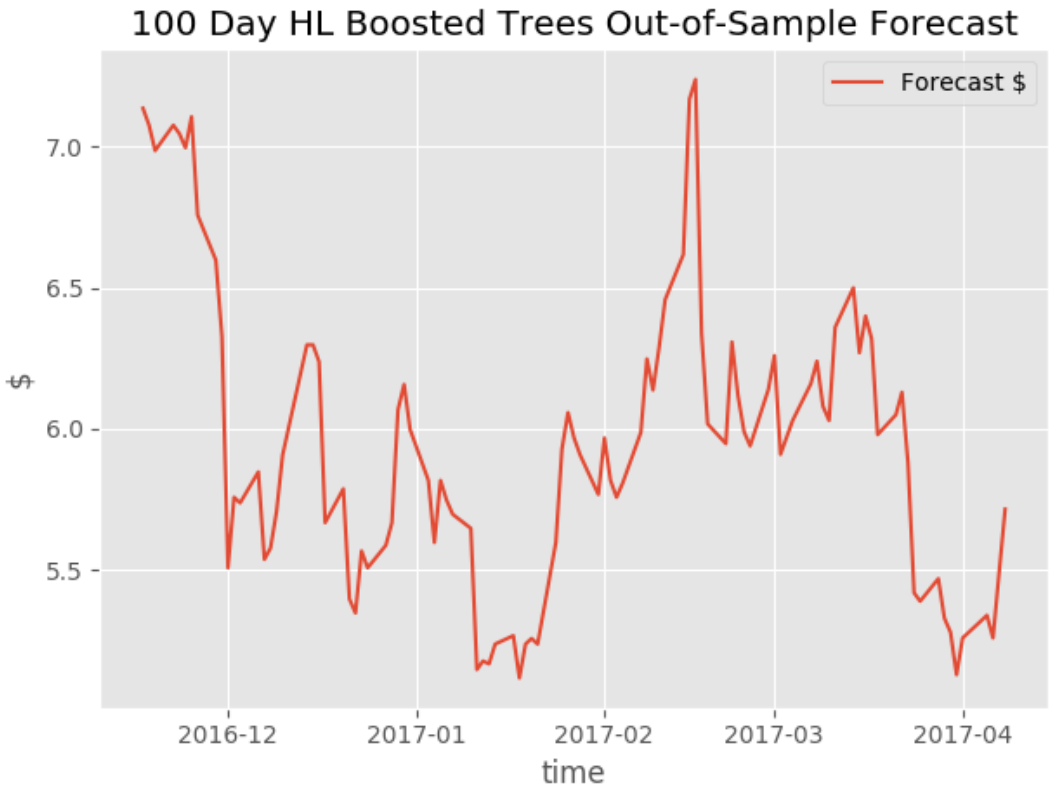
HRG scored a mean absolute error regression loss of 0.980, and a coefficient of determination of 0.855.





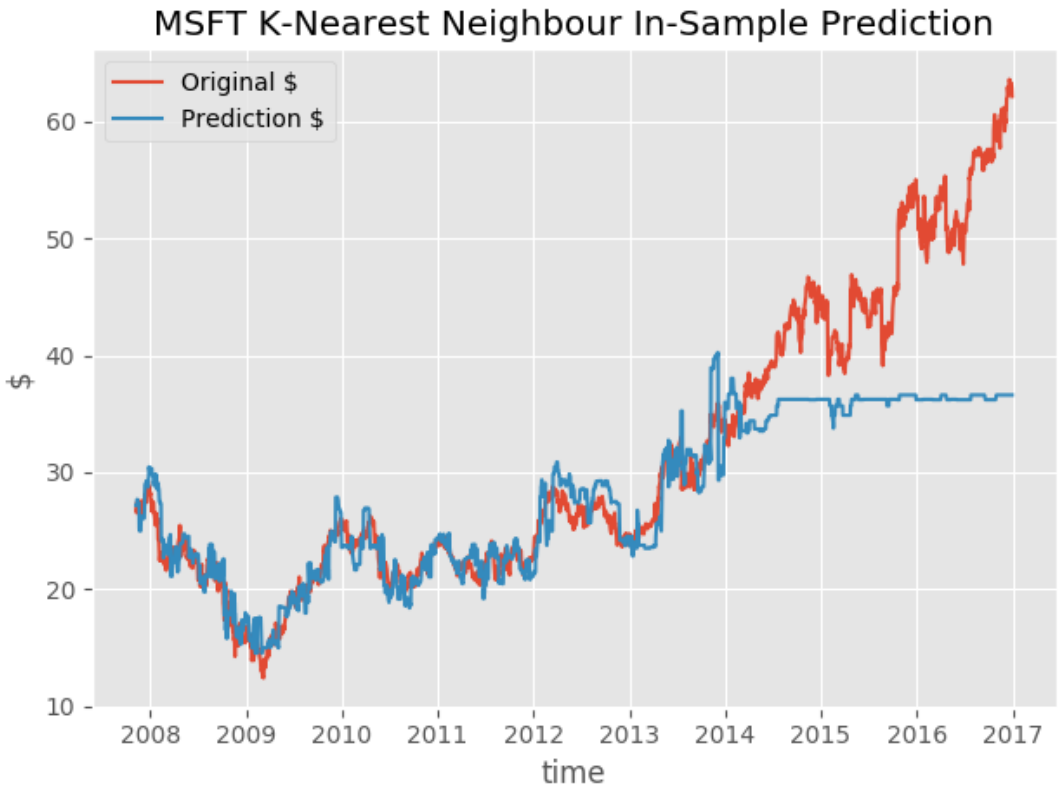
HL scored a mean absolute error regression loss of 0.410, and a coefficient of determination of 0.952.

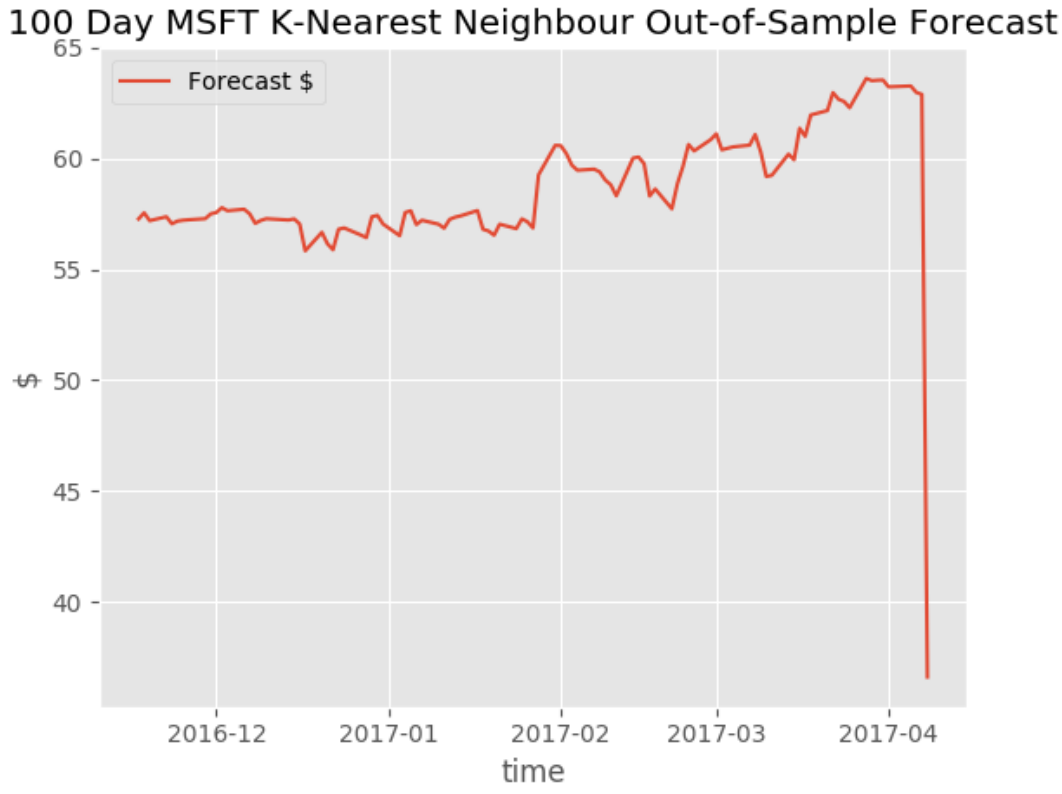




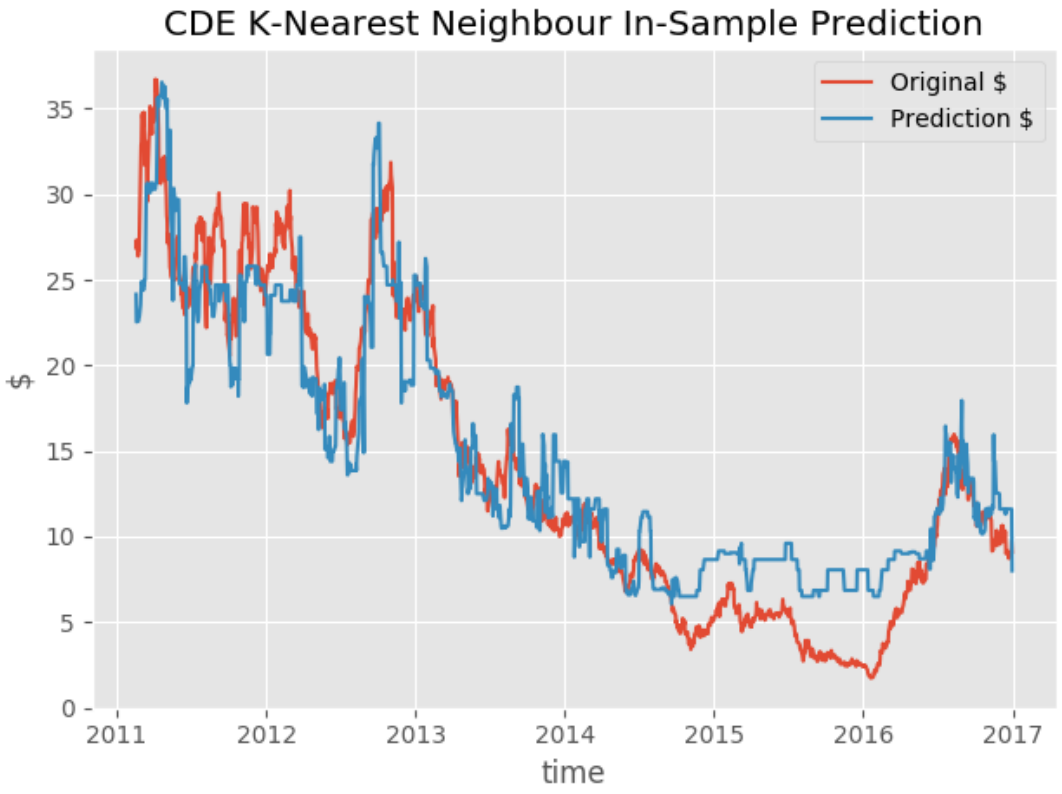
4.2.2.3 K-Nearest Neighbour

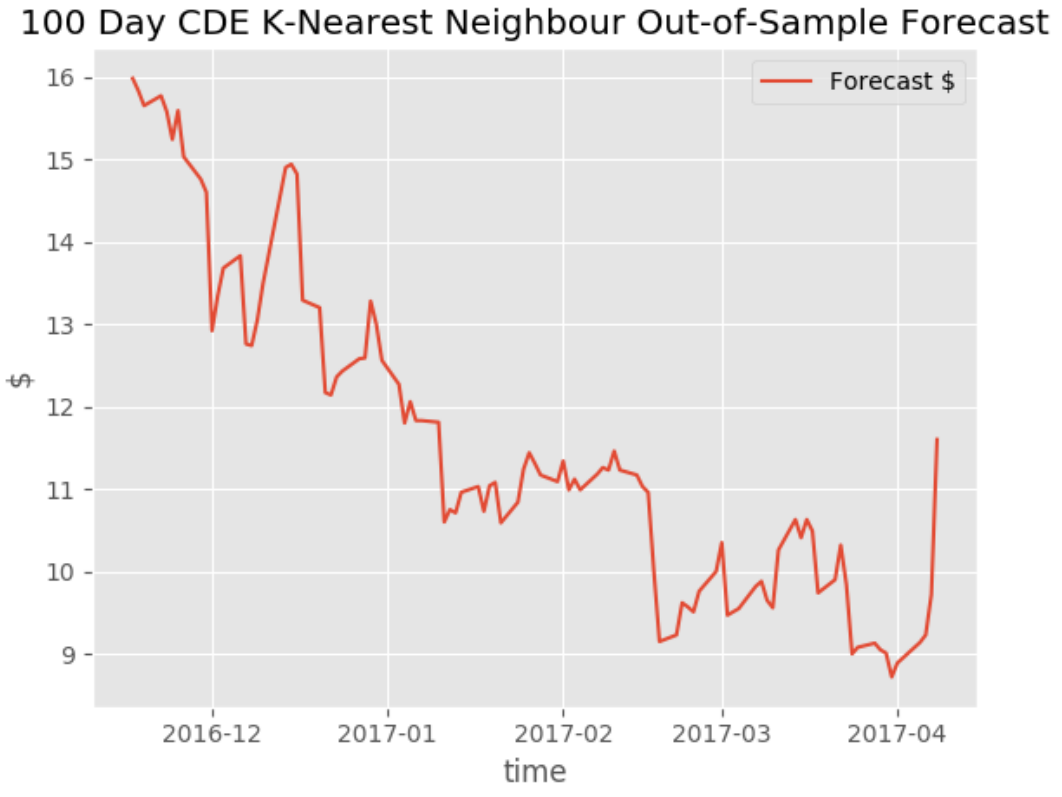
MSFT scored a mean absolute error regression loss of 4.426, and a coefficient of determination of 0.580.



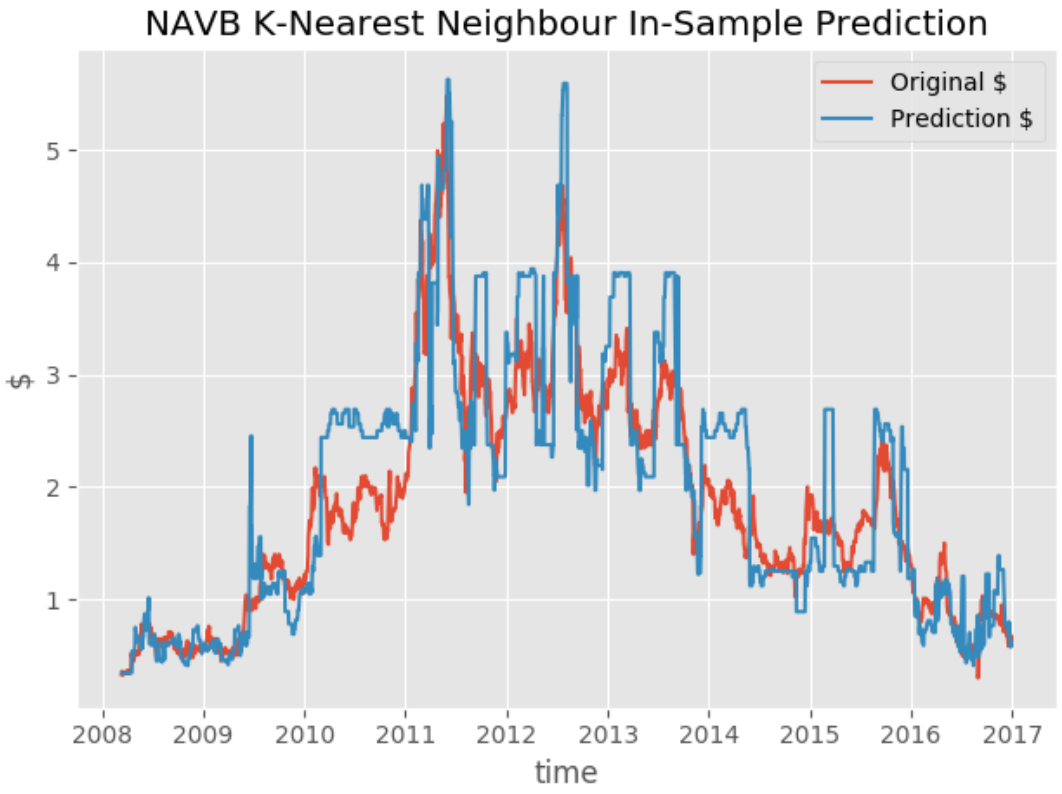


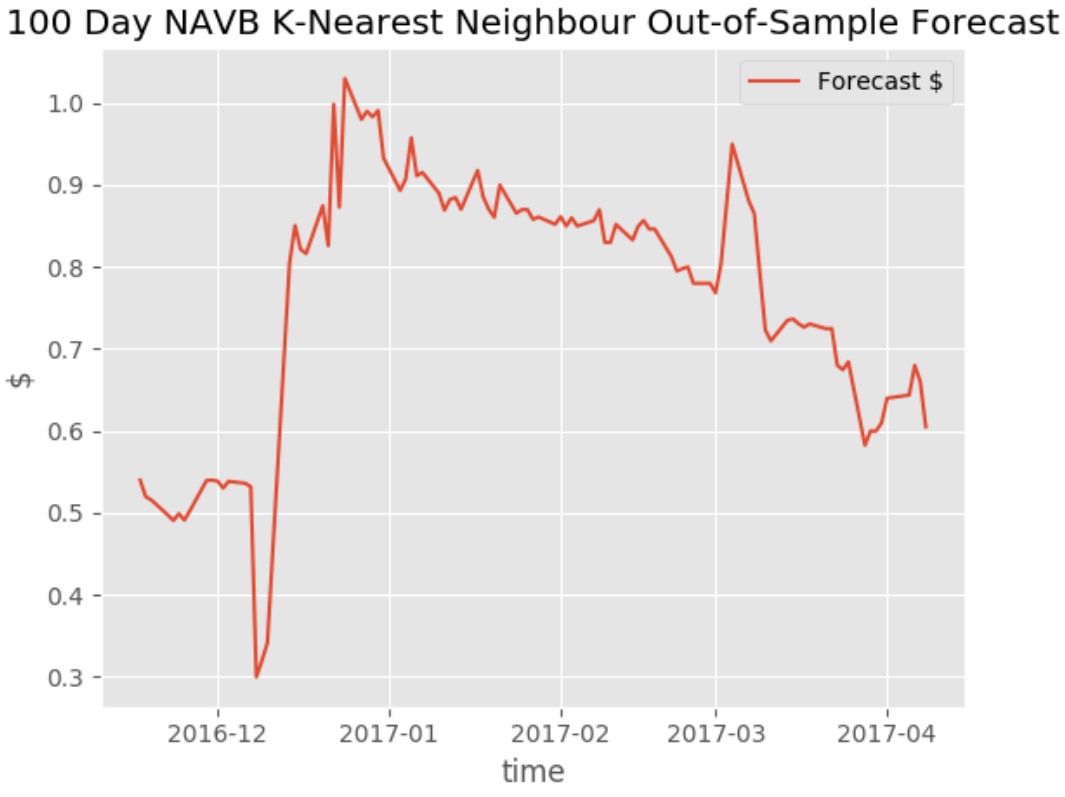
CDE scored a mean absolute error regression loss of 5.452, and a coefficient of determination of 0.581.



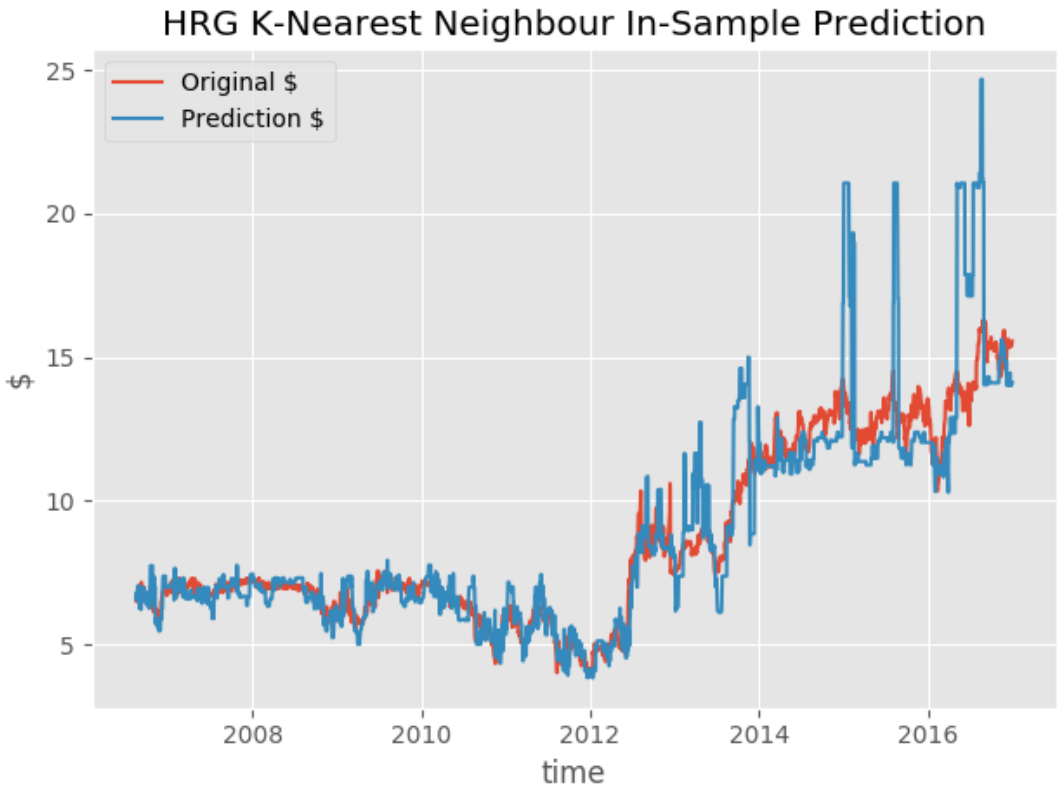


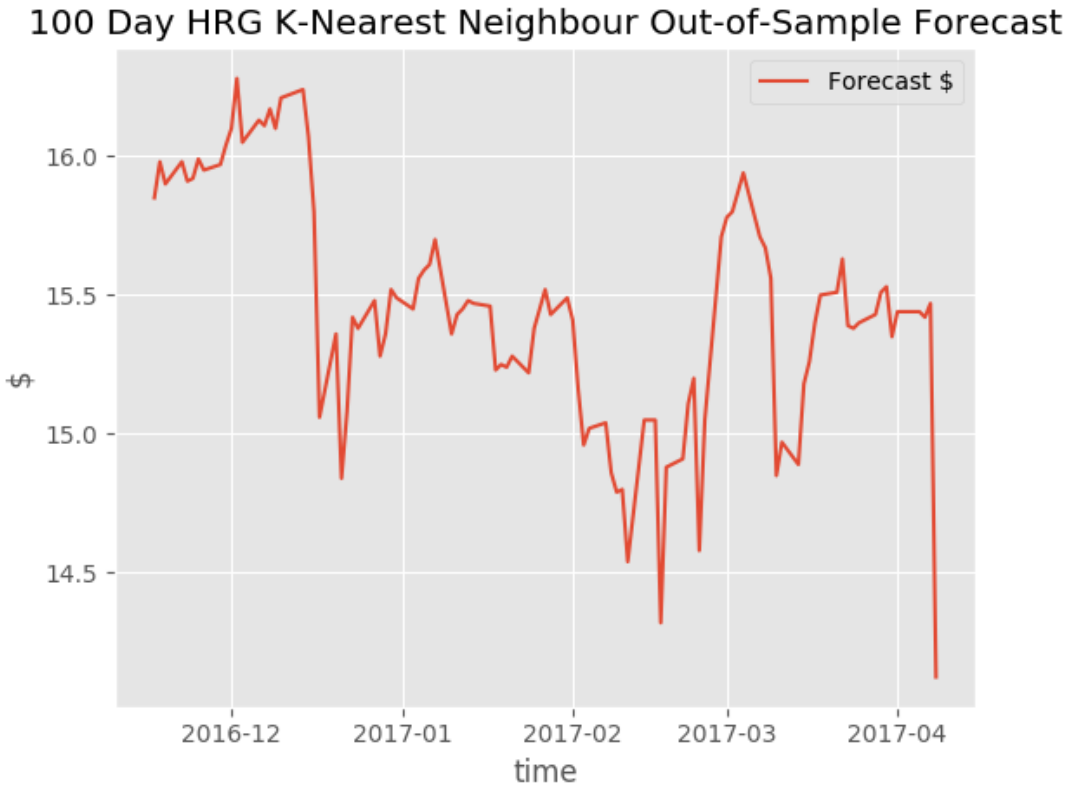
NAVB scored a mean absolute error regression loss of 0.579, and a coefficient of determination of 0.509.



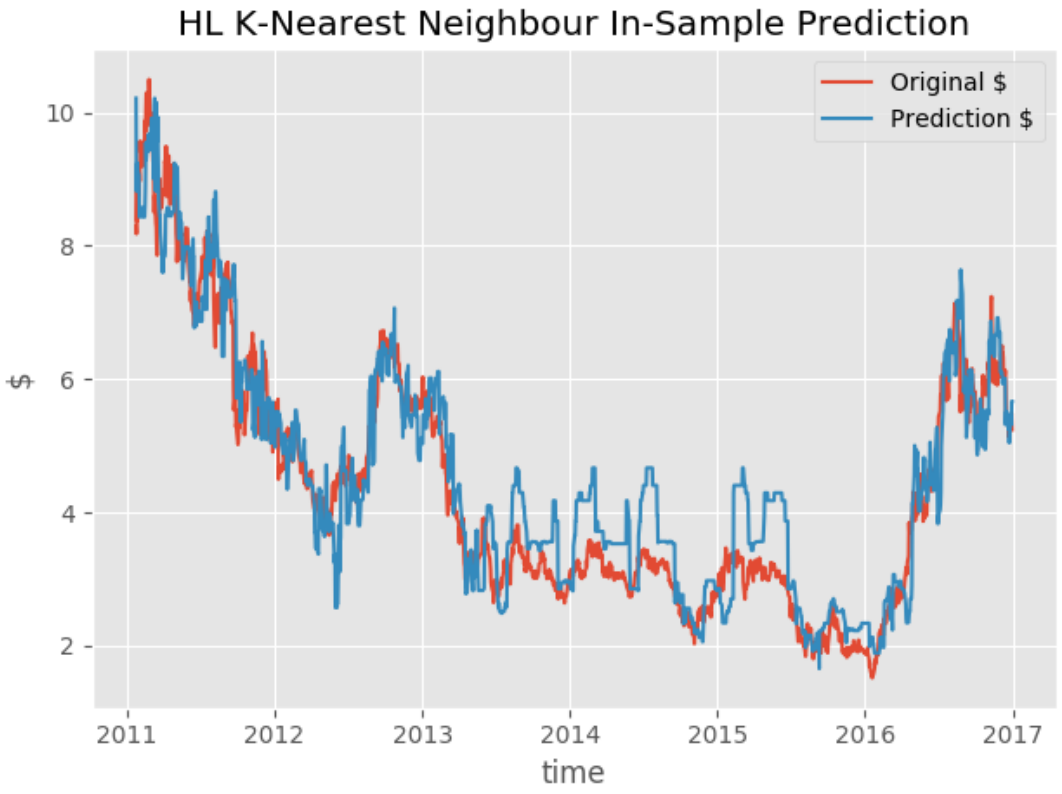


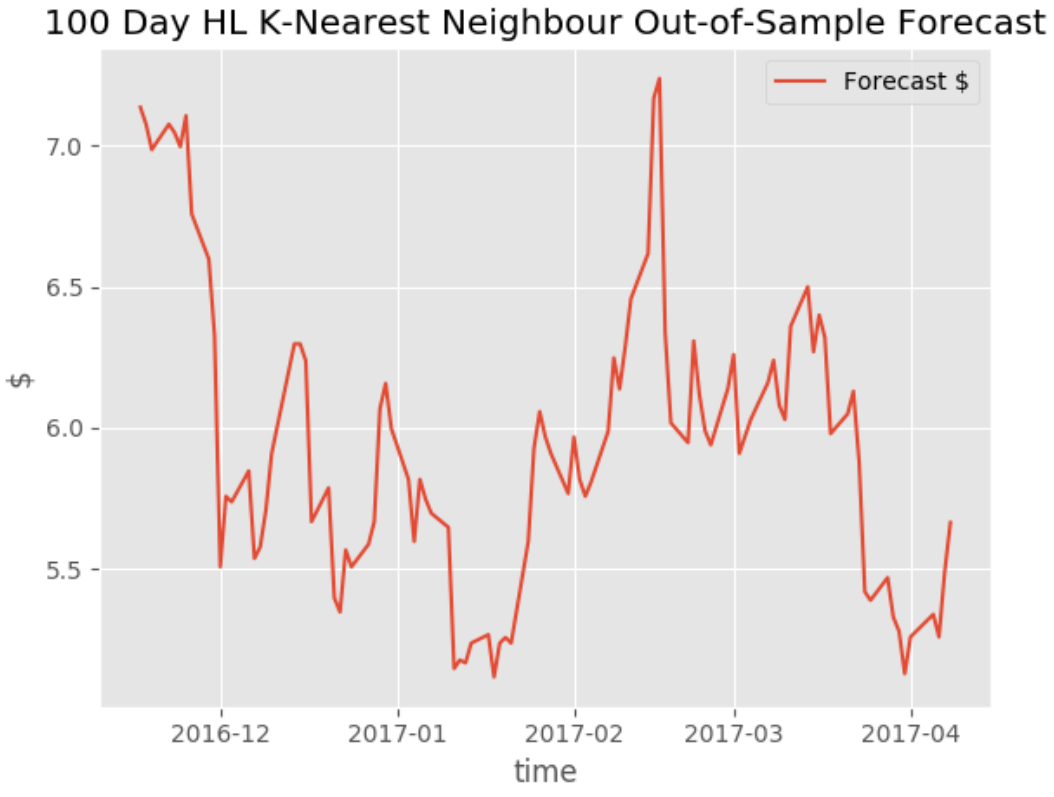
HRG scored a mean absolute error regression loss of 0.980, and a coefficient of determination of 0.855.





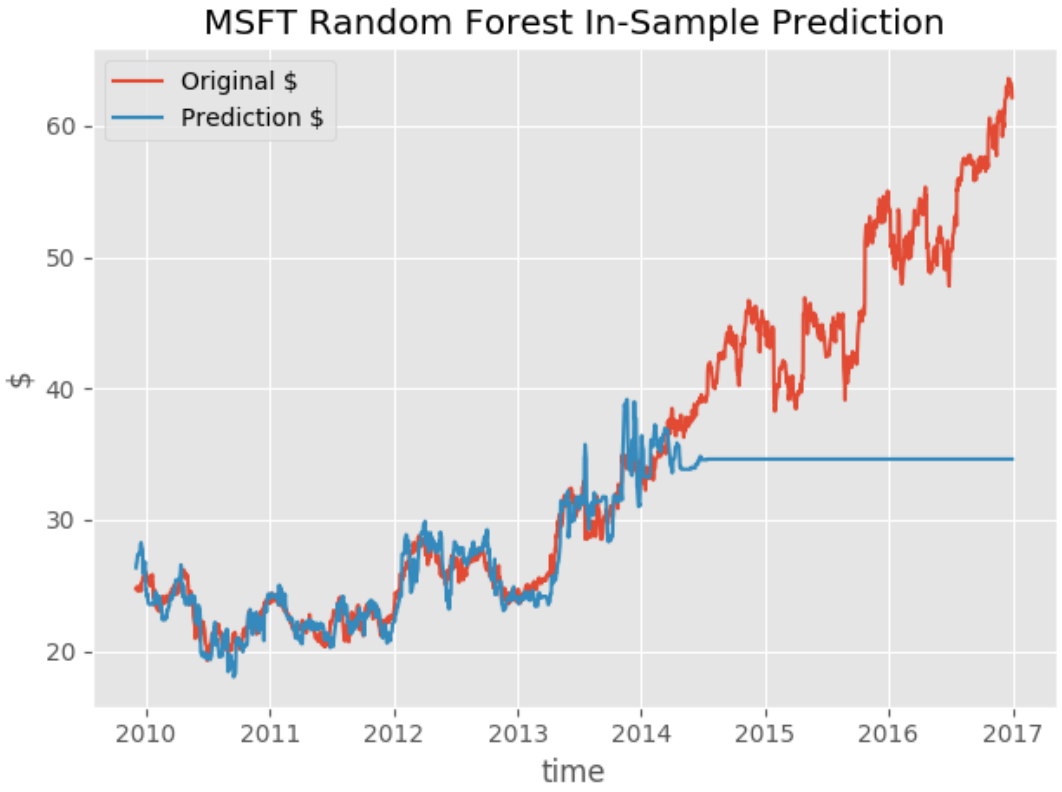
HL scored a mean absolute error regression loss of 0.410, and a coefficient of determination of 0.952.

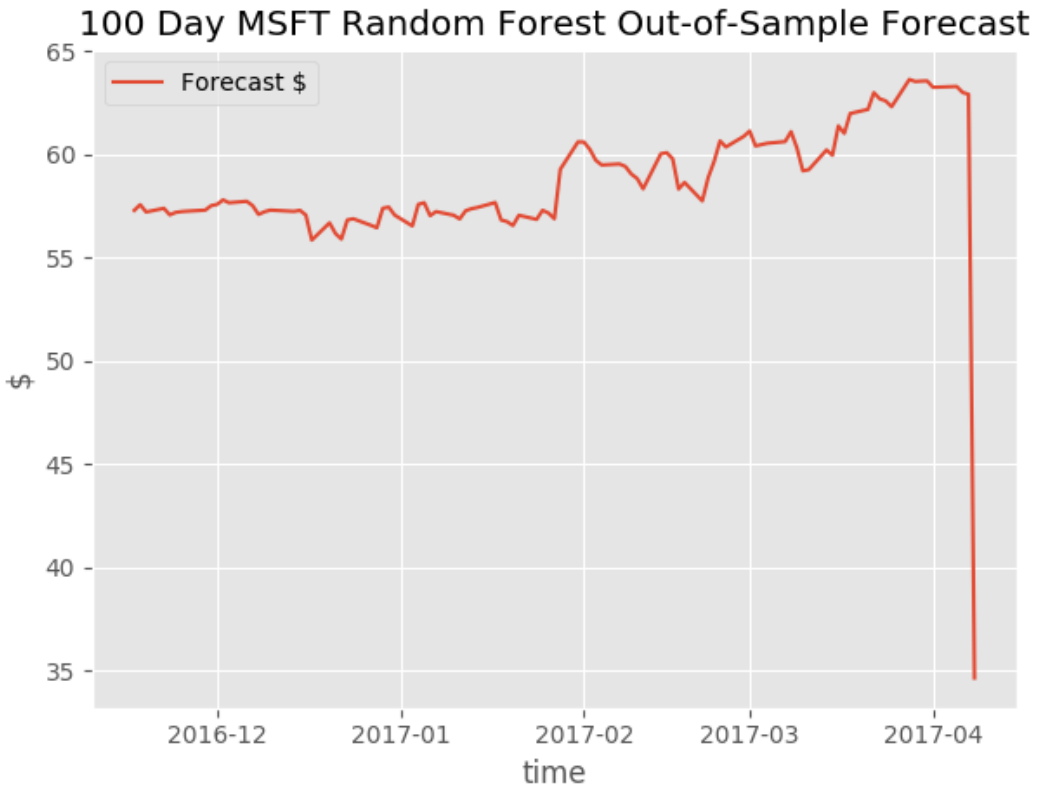

HL K-Nearest Neighbour In-Sample Prediction
Original $
Prediction $
10
8
6
4
2
$
2011
2012
2013
2014
2015
2016
2017
time


100 Day HL K-Nearest Neighbour Out-of-Sample Forecast
Forecast $
7.0
6.5
6.0
5.5
$
2016-12
2017-01
2017-02
2017-03
2017-04
time

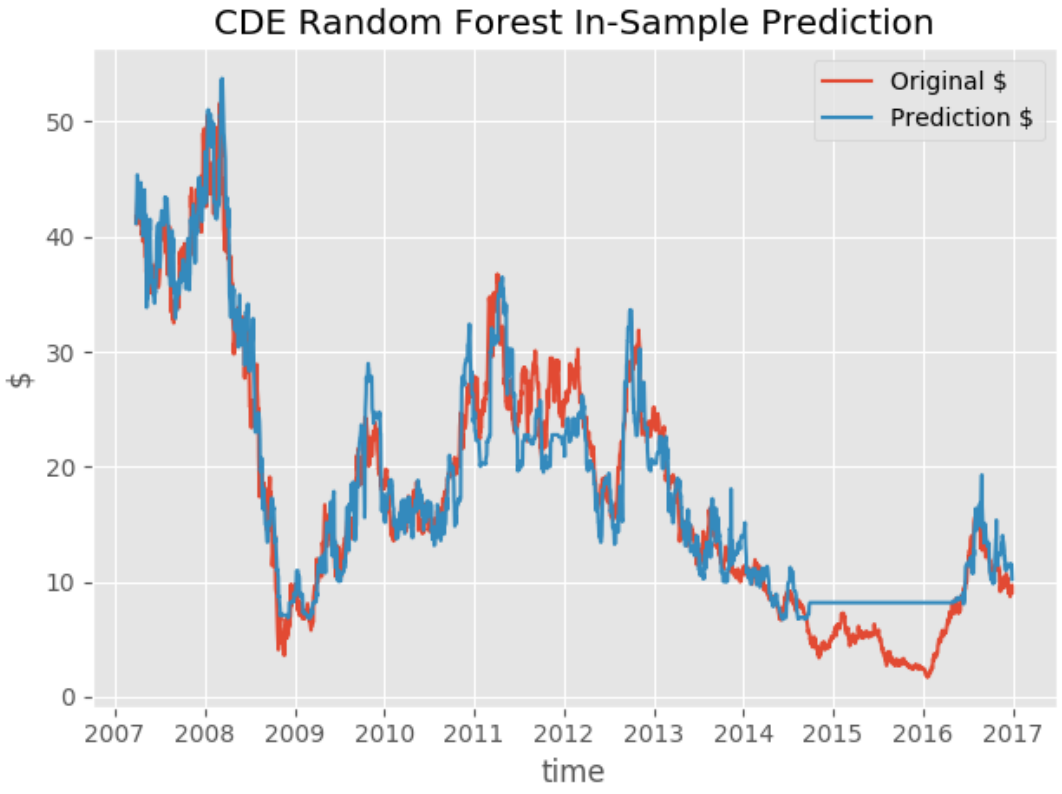
4.2.2.4 Random Forest

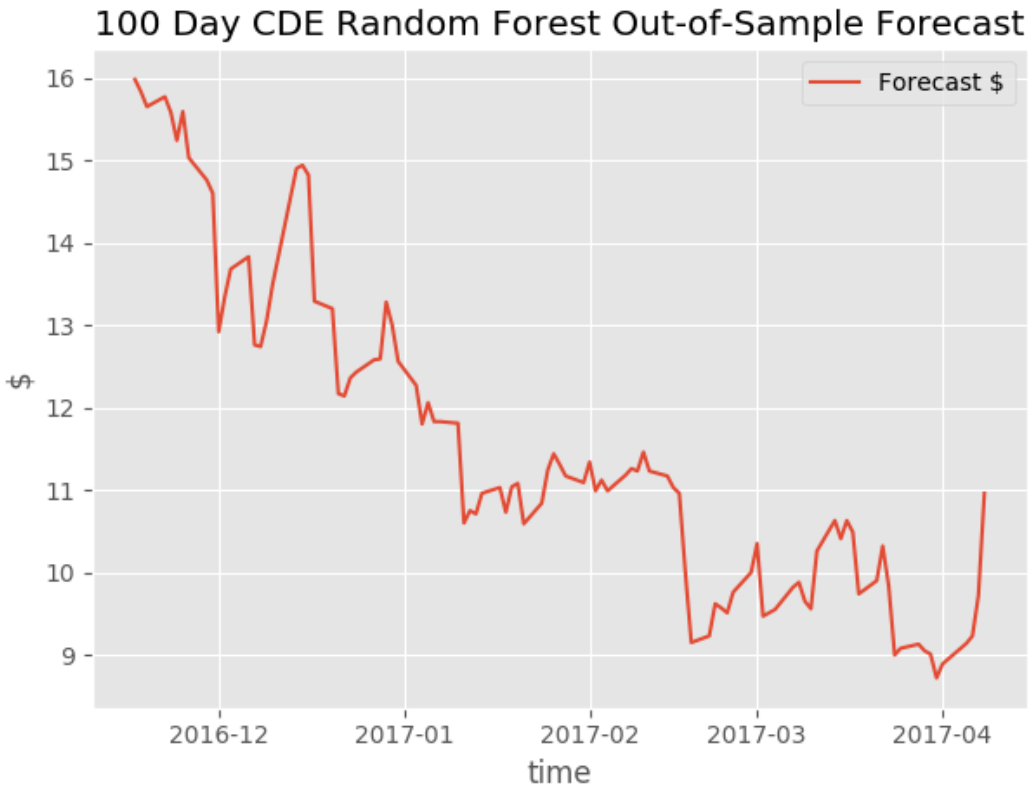
MSFT scored a mean absolute error regression loss of 5.199, and a coefficient of determination of 0.483.



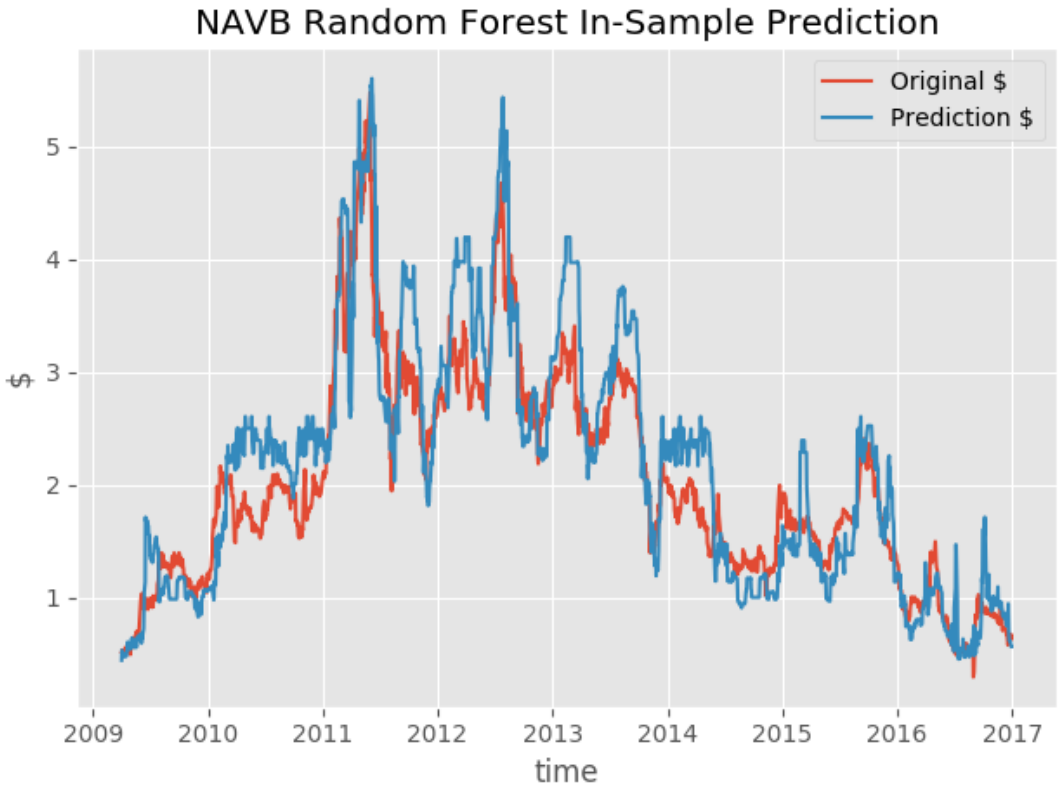


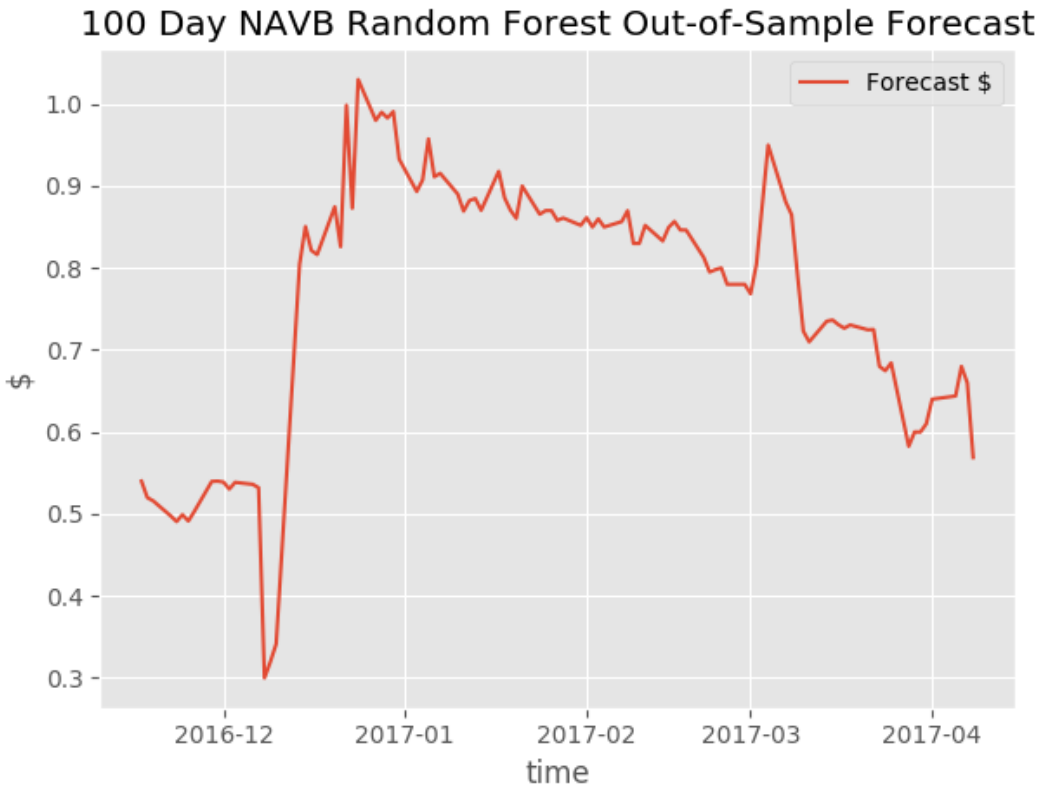
CDE scored a mean absolute error regression loss of 2.505, and a coefficient of determination of 0.861.



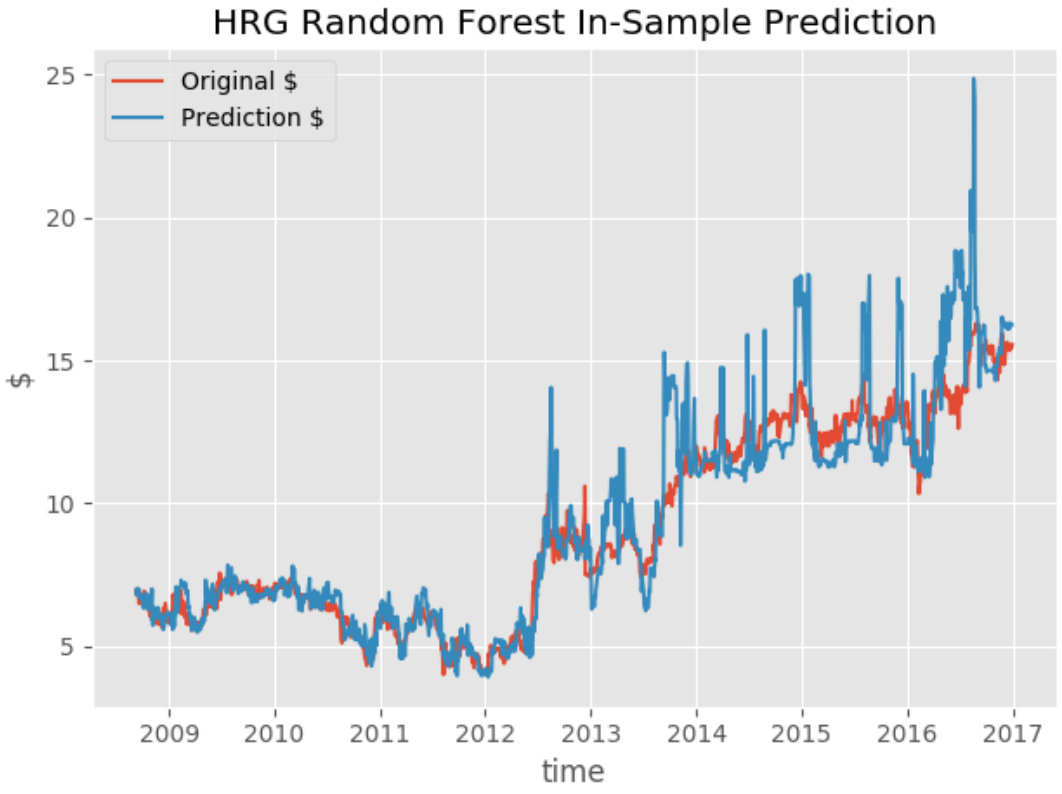


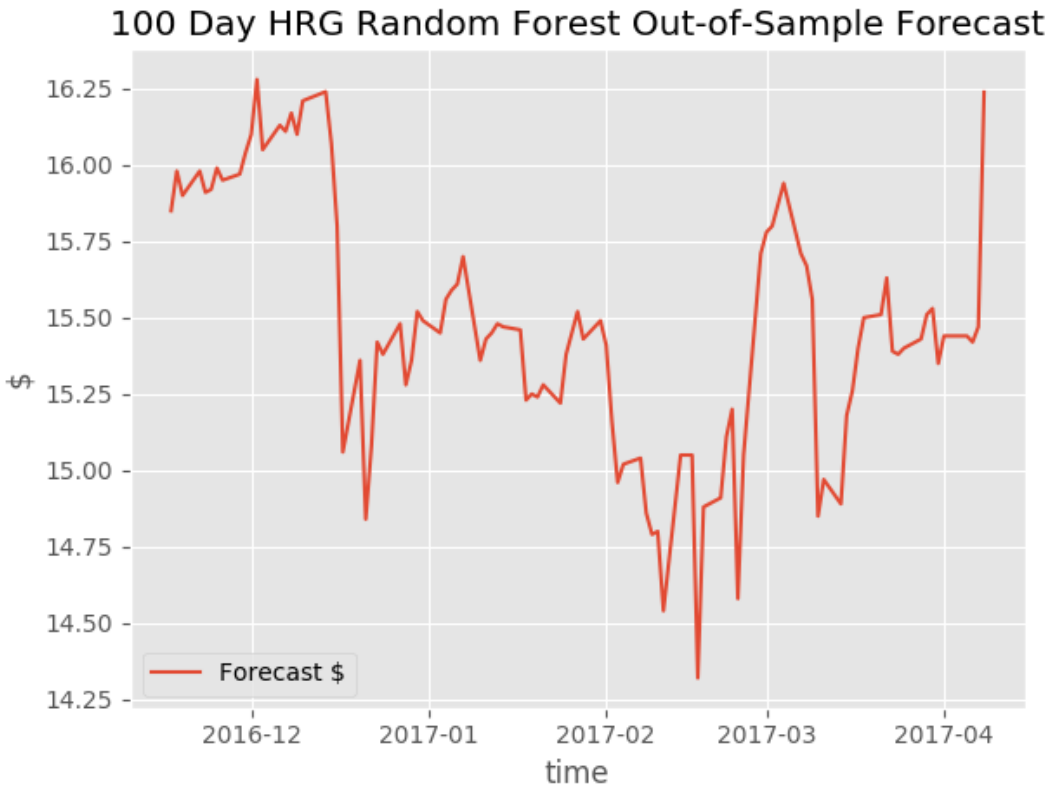
NAVB scored a mean absolute error regression loss of 0.293, and a coefficient of determination of 0.856.



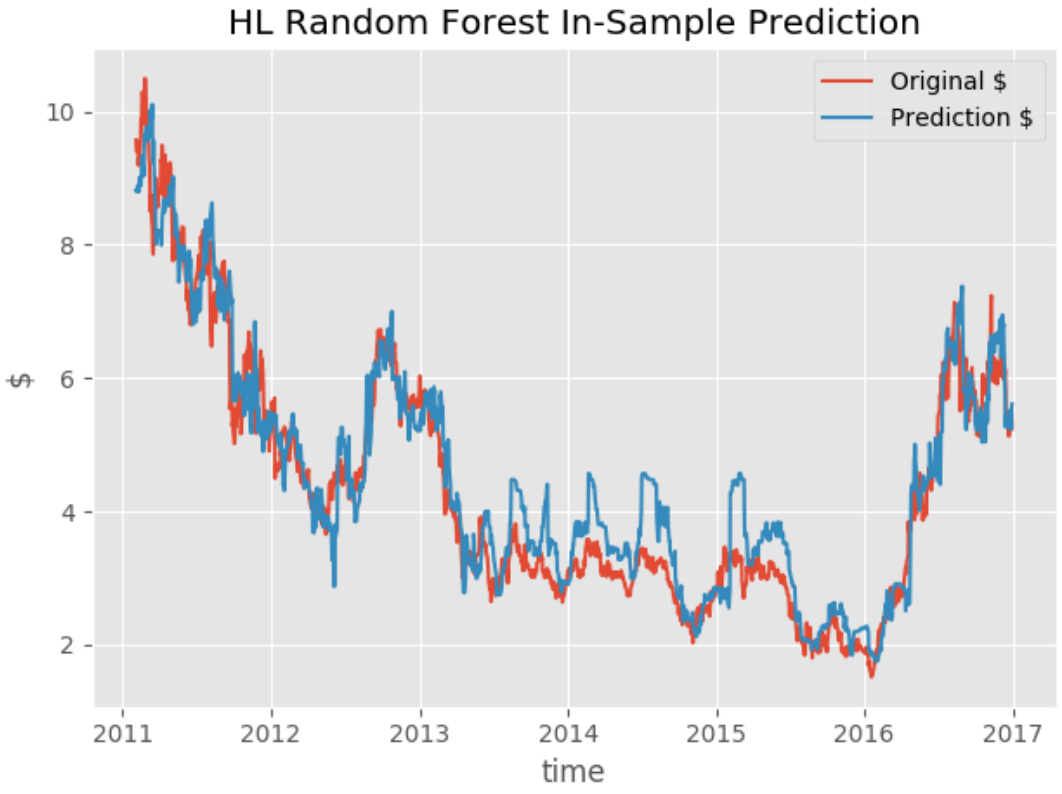


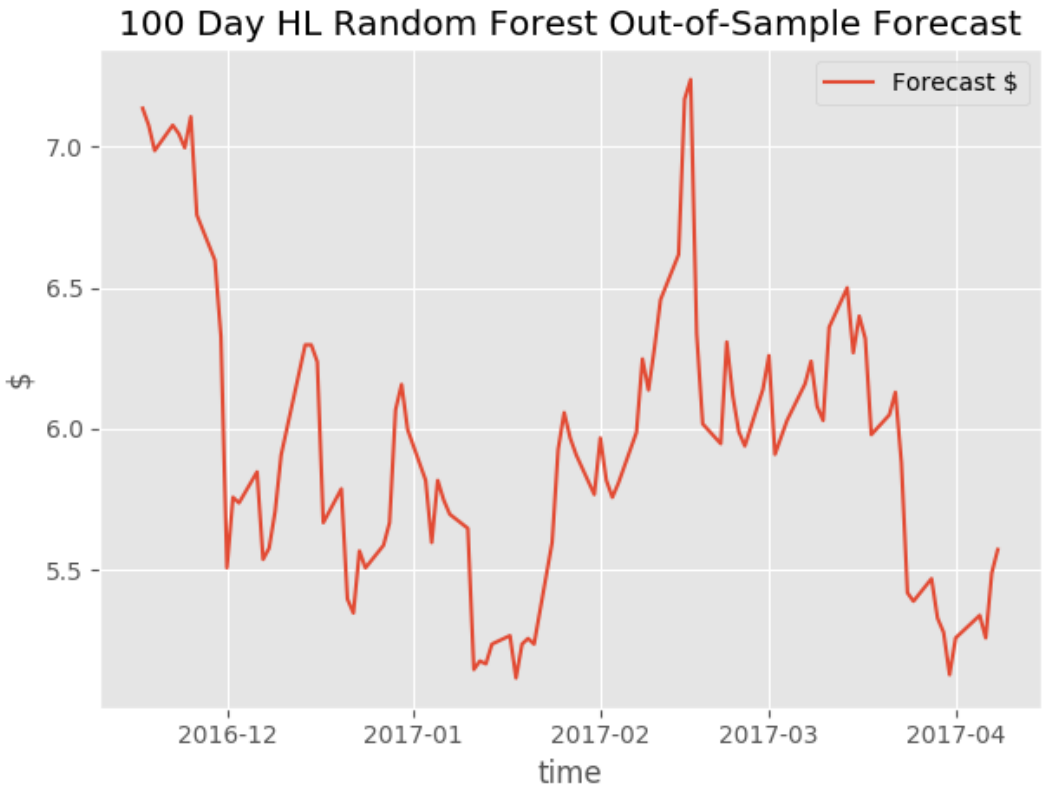
HRG scored a mean absolute error regression loss of 0.675, and a coefficient of determination of 0.907.





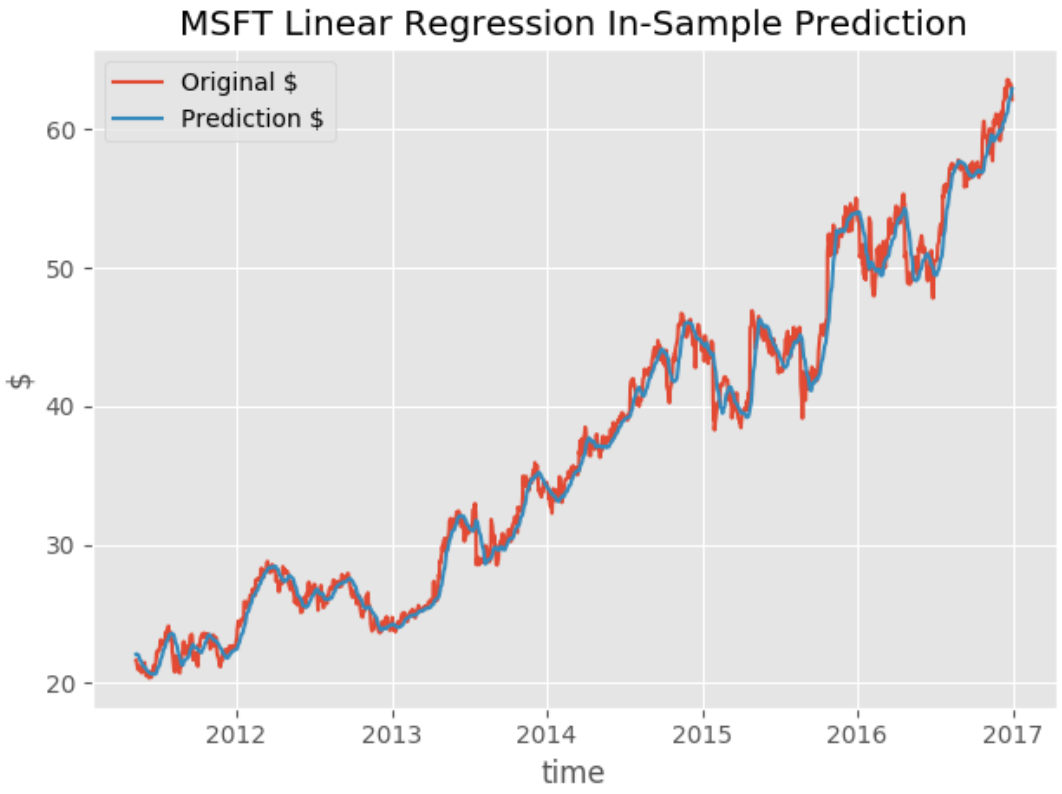
HL scored a mean absolute error regression loss of 0.434, and a coefficient of determination of 0.922.

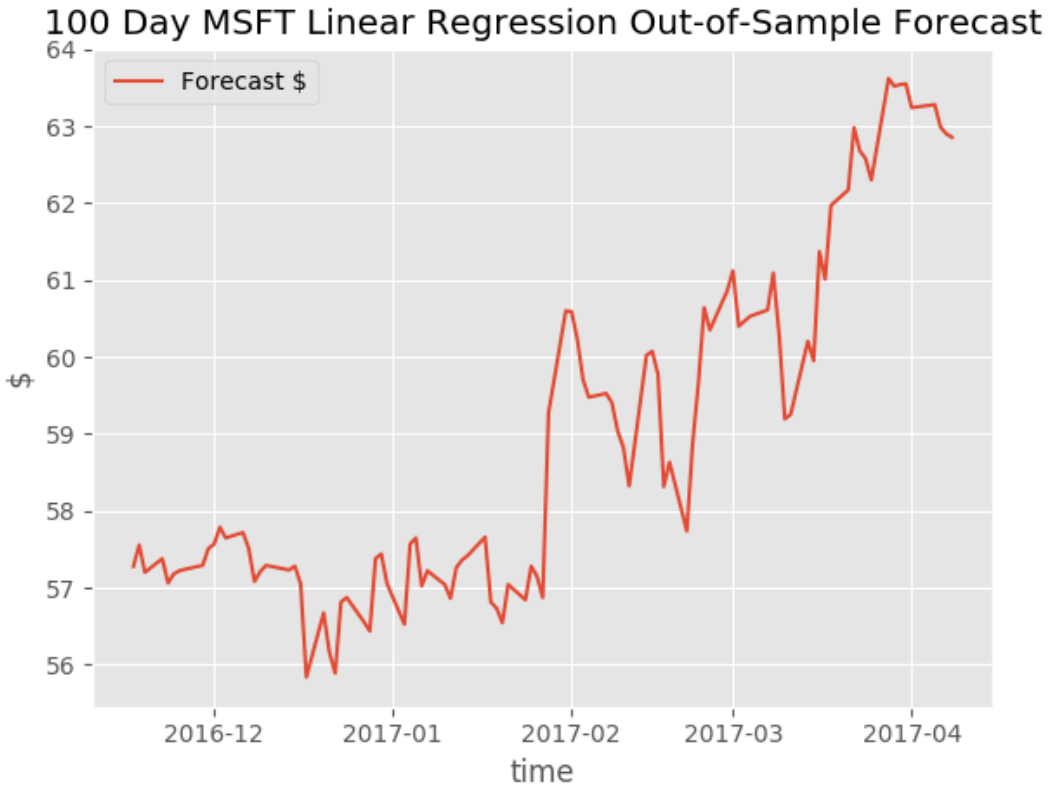




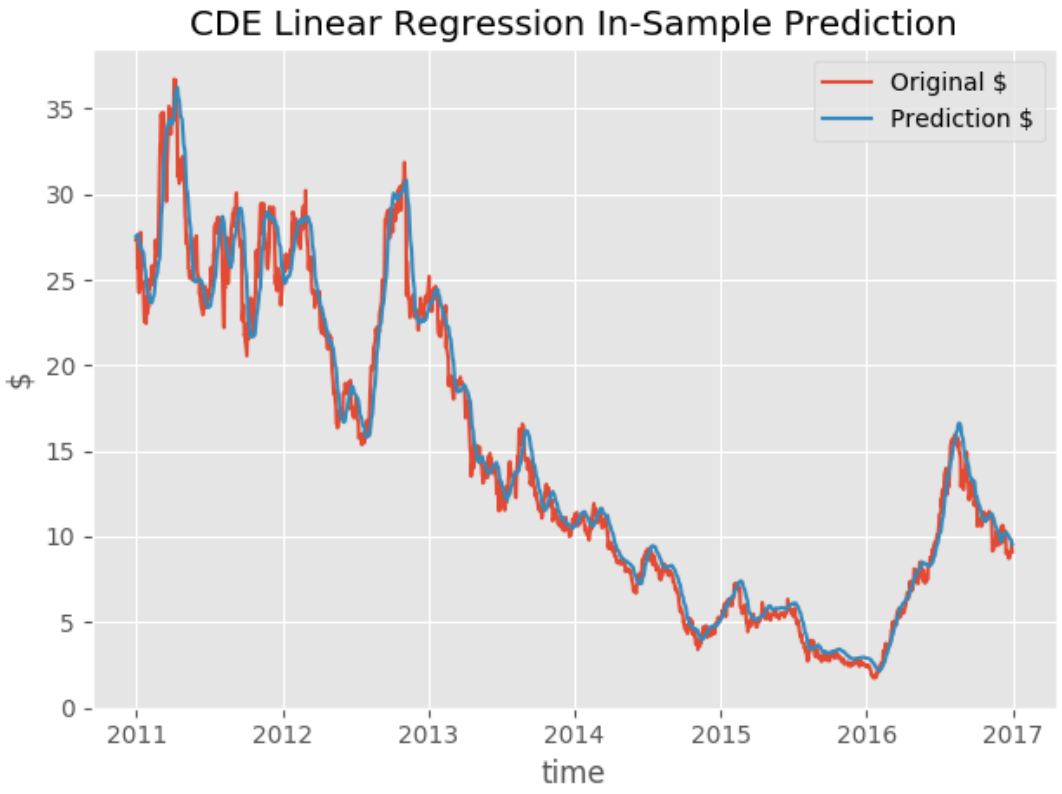
4.2.2.5 Linear Regression

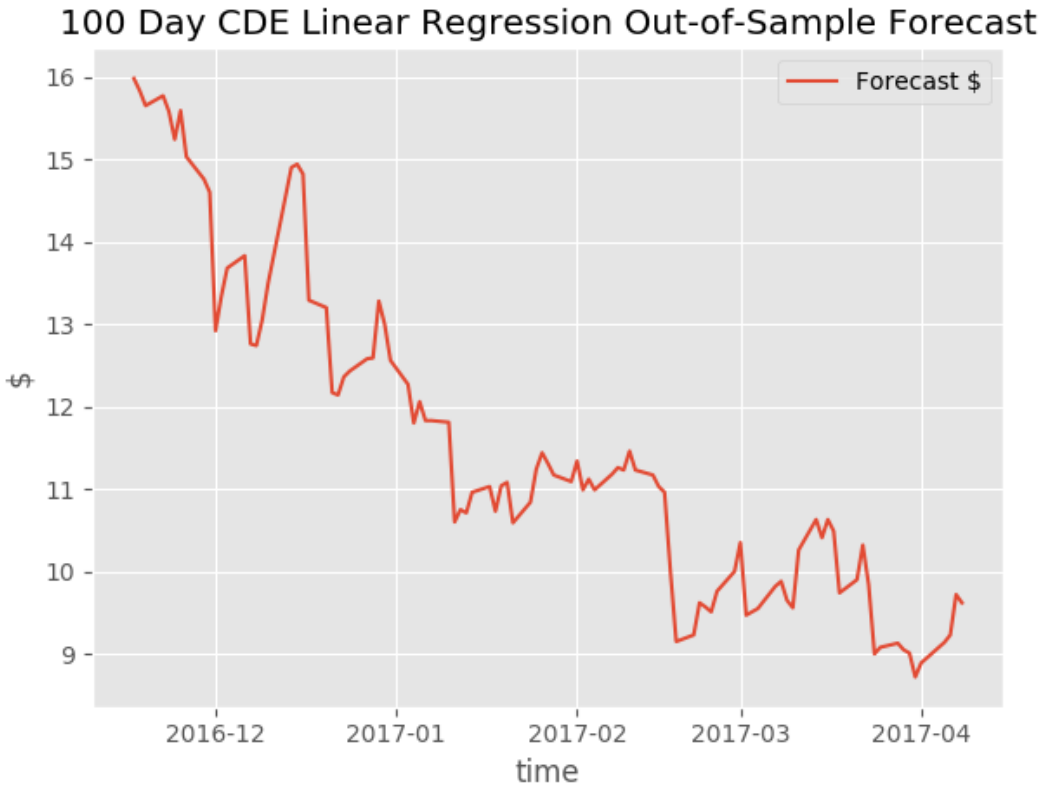
MSFT scored a mean absolute error regression loss of 0.844, and a coefficient of determination of 0.990.



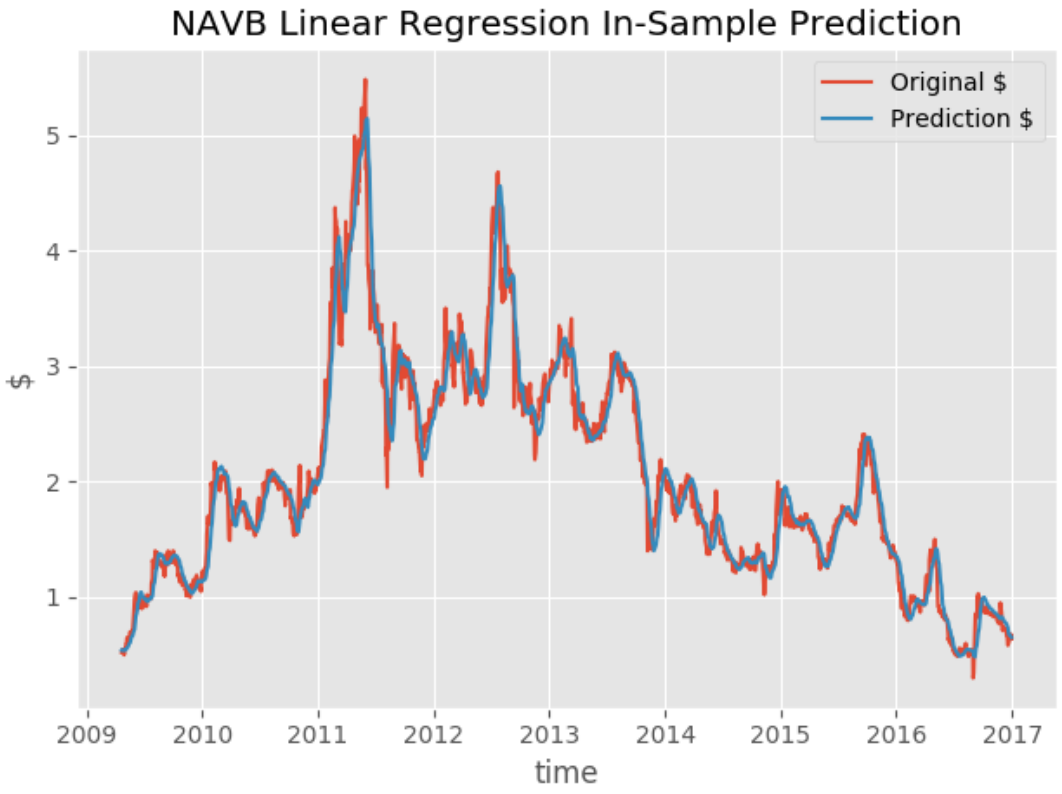


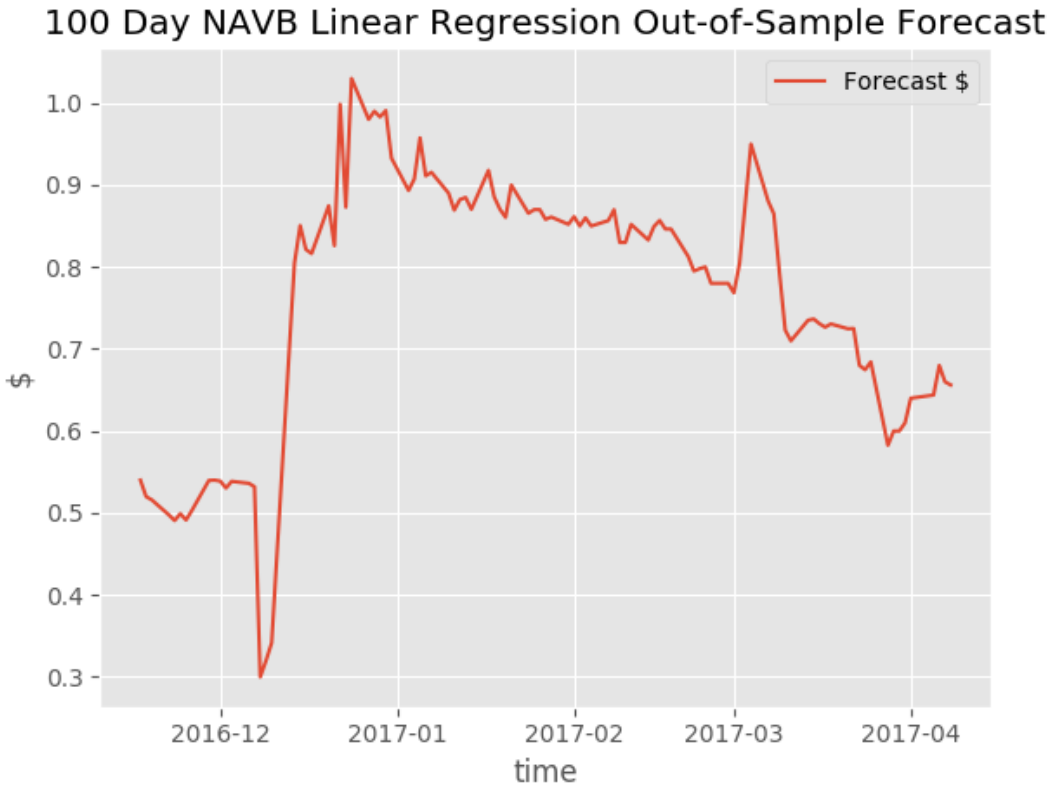
CDE scored a mean absolute error regression loss of 0.990, and a coefficient of determination of 0.972.



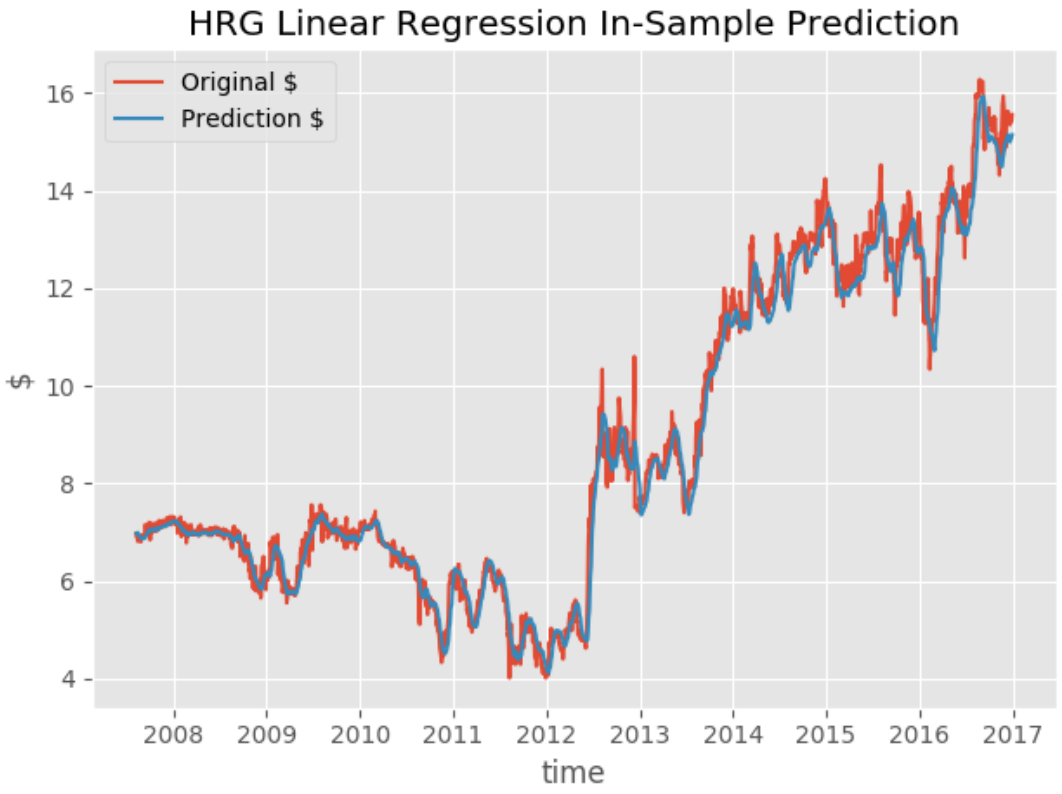


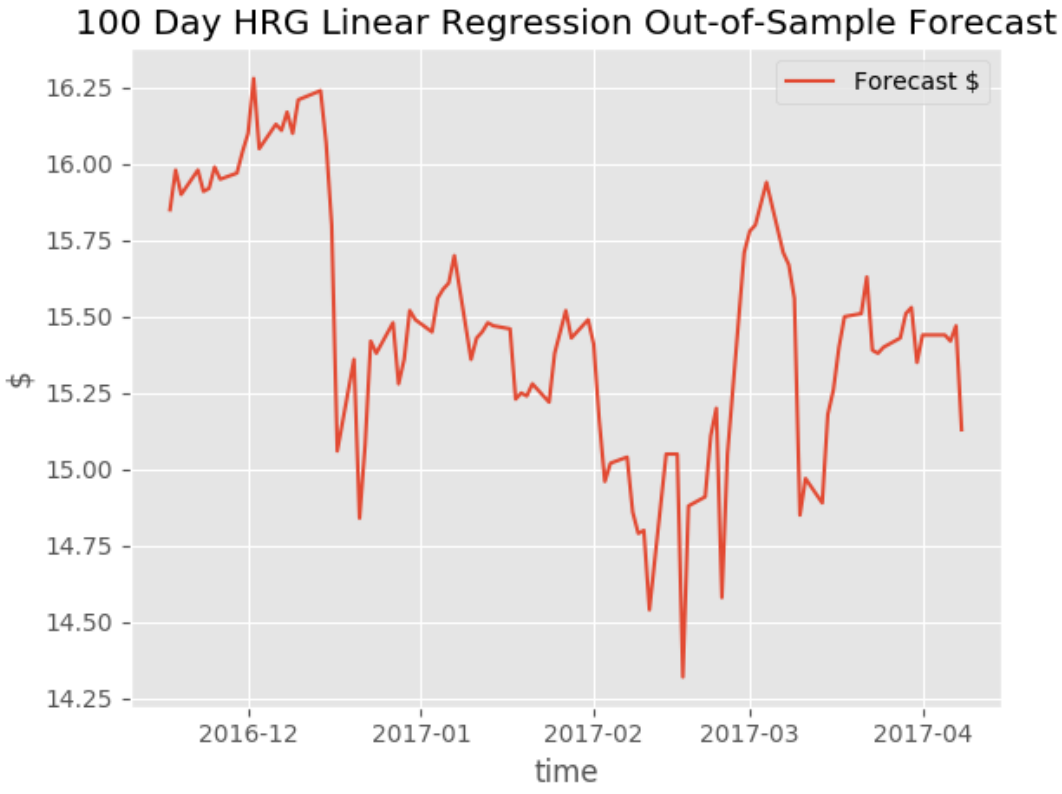
NAVB scored a mean absolute error regression loss of 0.128, and a coefficient of determination of 0.962.



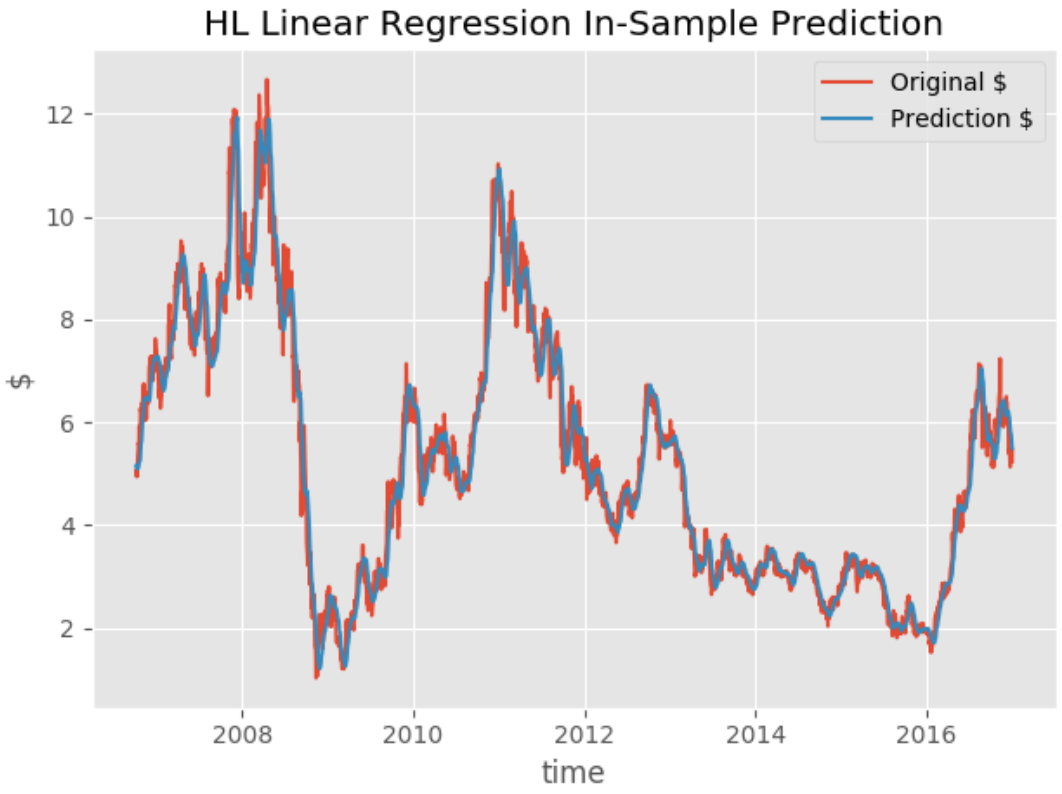


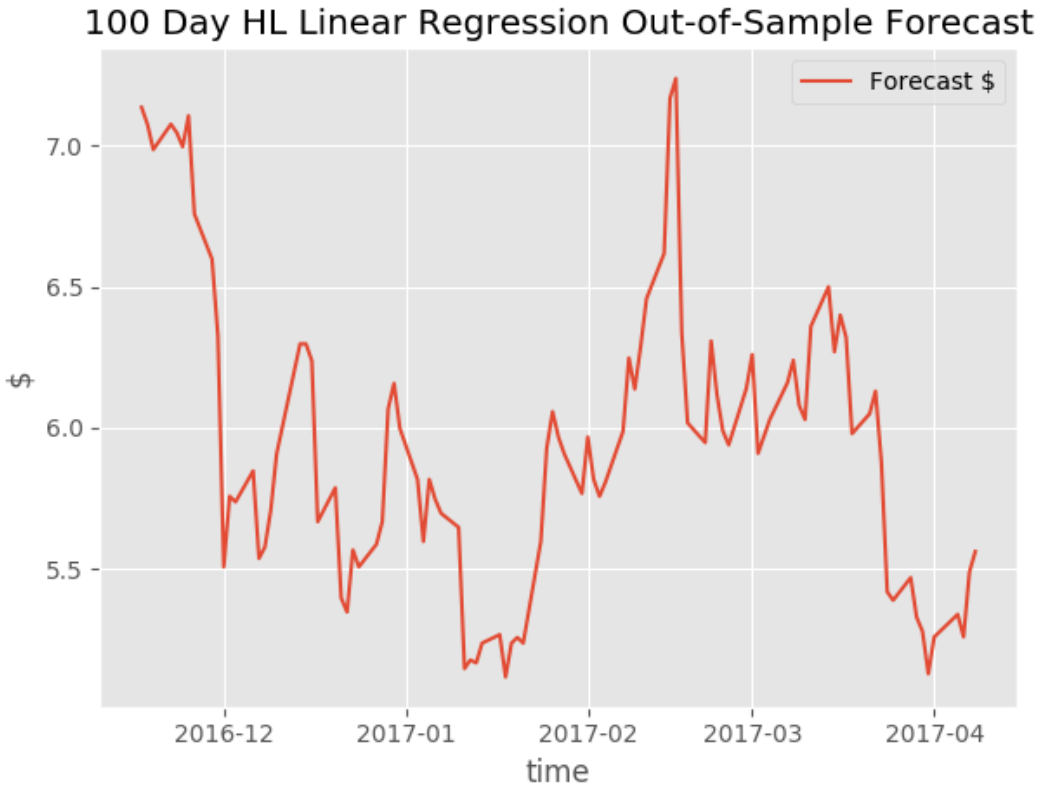
HRG scored a mean absolute error regression loss of 0.324, and a coefficient of determination of 0.984.





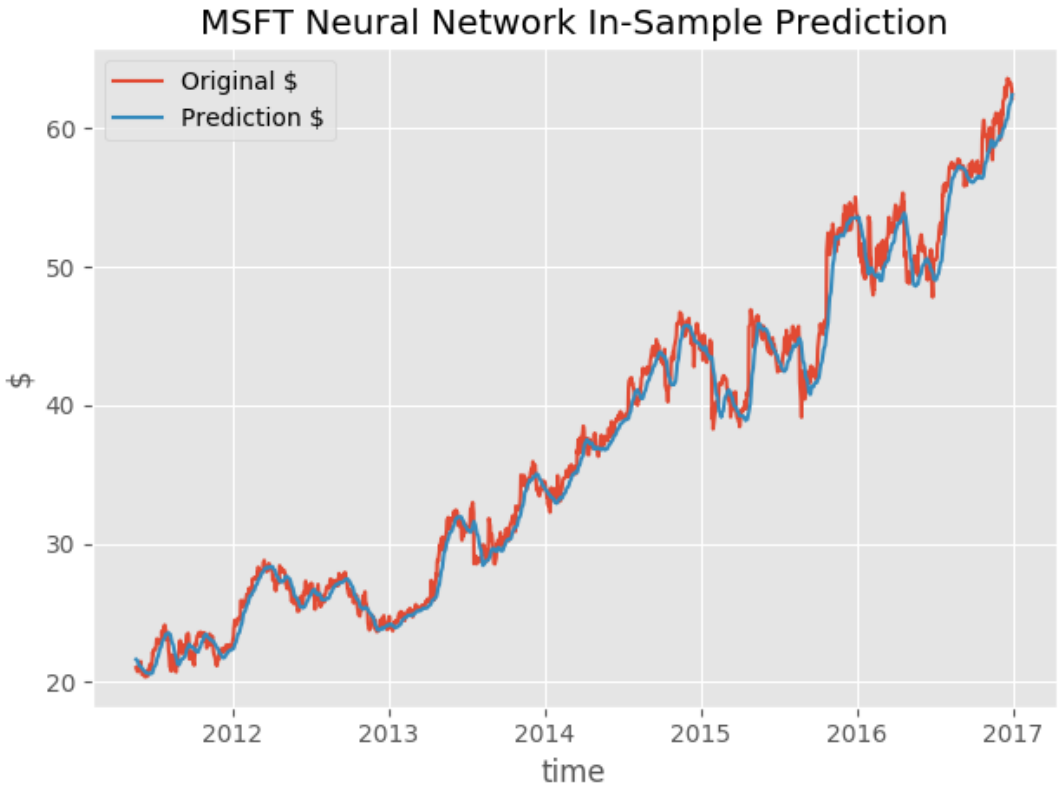
HL scored a mean absolute error regression loss of 0.243, and a coefficient of determination of 0.964.

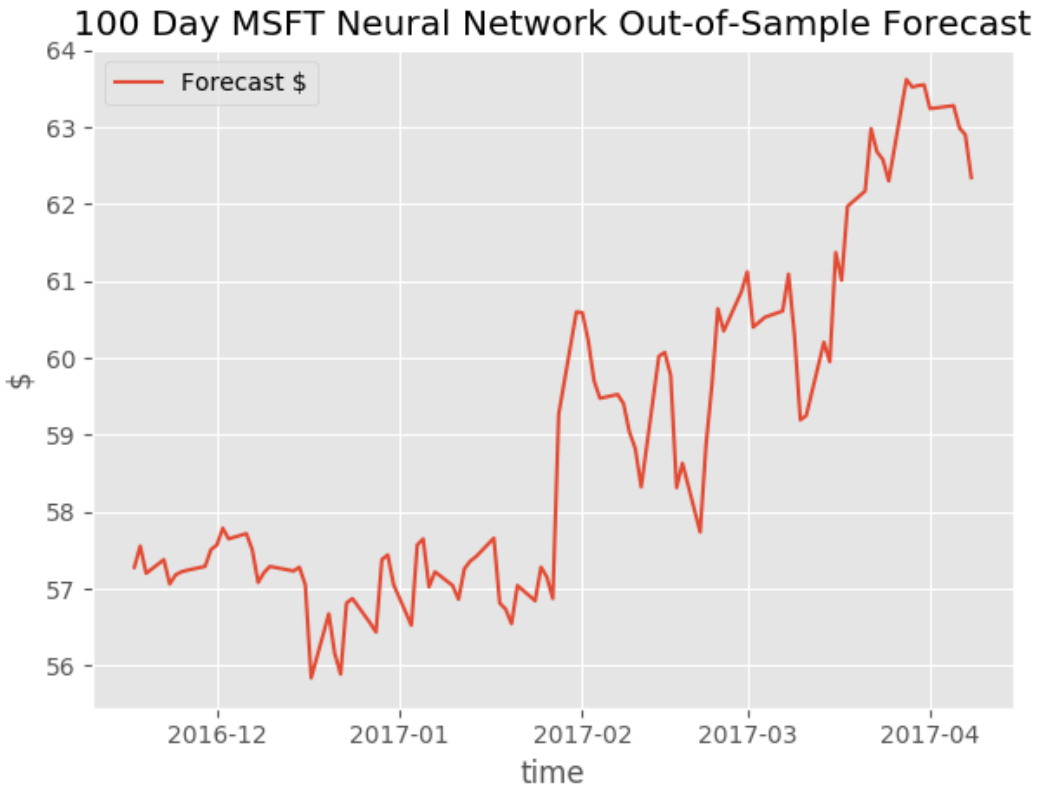

HL Linear Regression In-Sample Prediction
Original $
Prediction $
12
10
8
6
4
2
$
2008
2010
2012
2014
2016
time


100 Day HL Linear Regression Out-of-Sample Forecast
Forecast $
7.0
6.5
6.0
5.5
$
2016-12
2017-01
2017-02
2017-03
2017-04
time

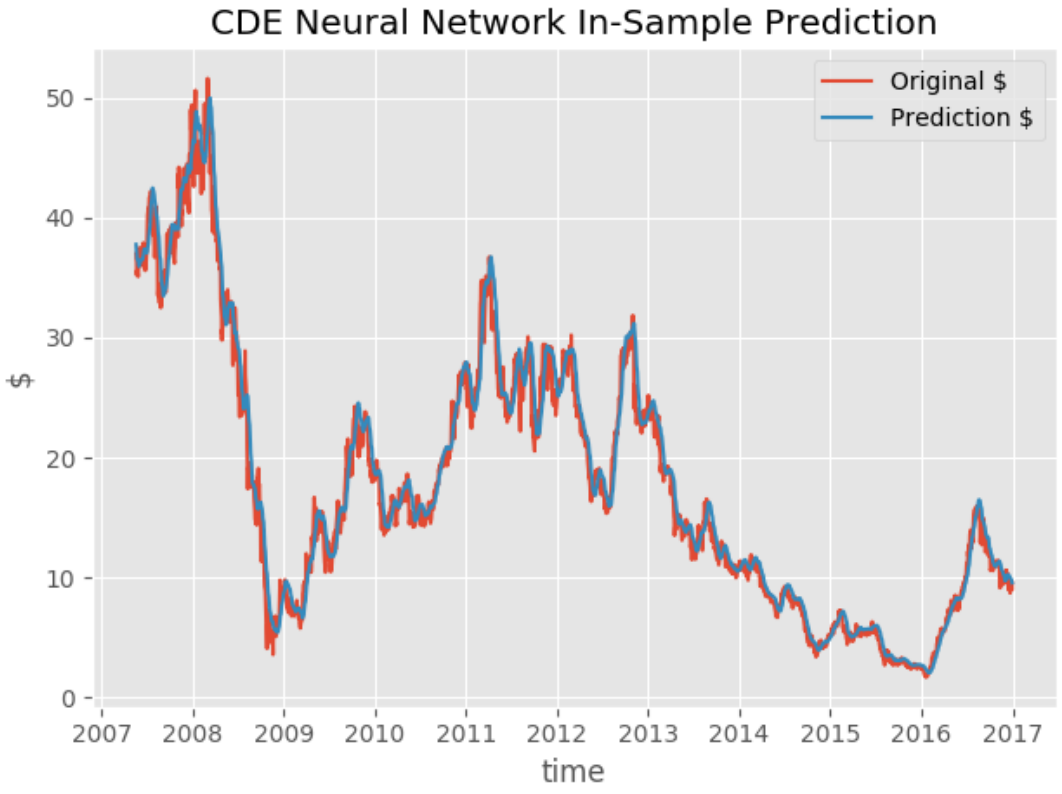
4.2.2.6 Neural Network

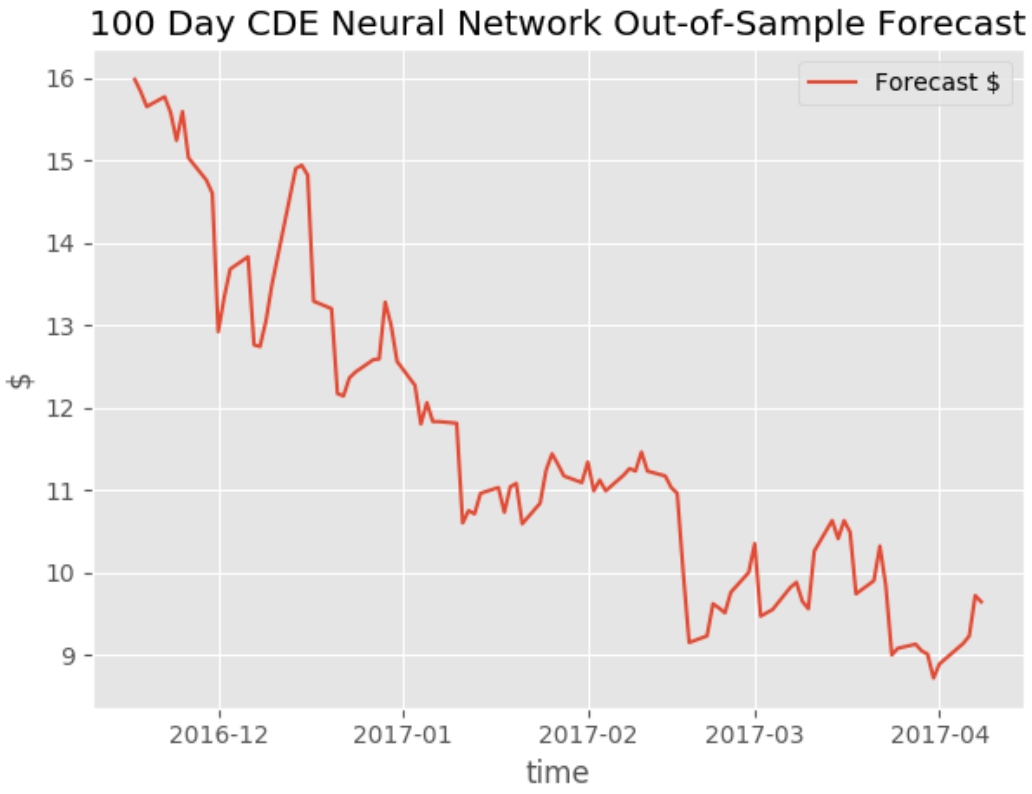
MSFT scored a mean absolute error regression loss of 1.073, and a coefficient of determination of 0.992.



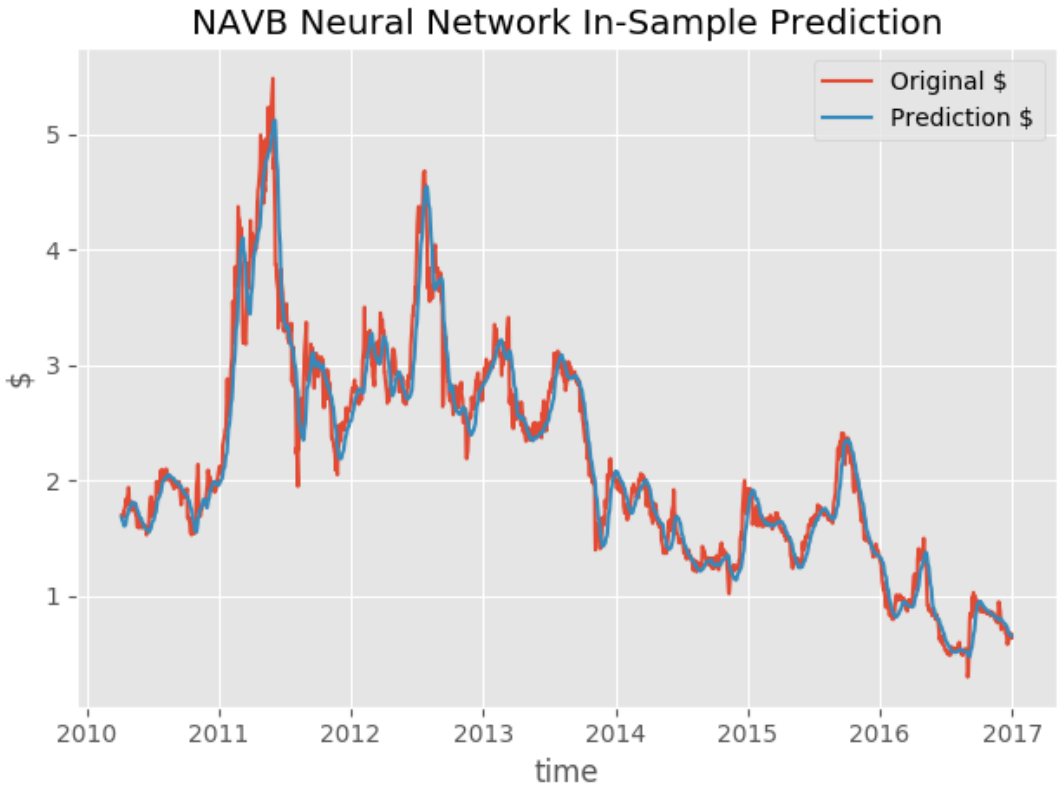


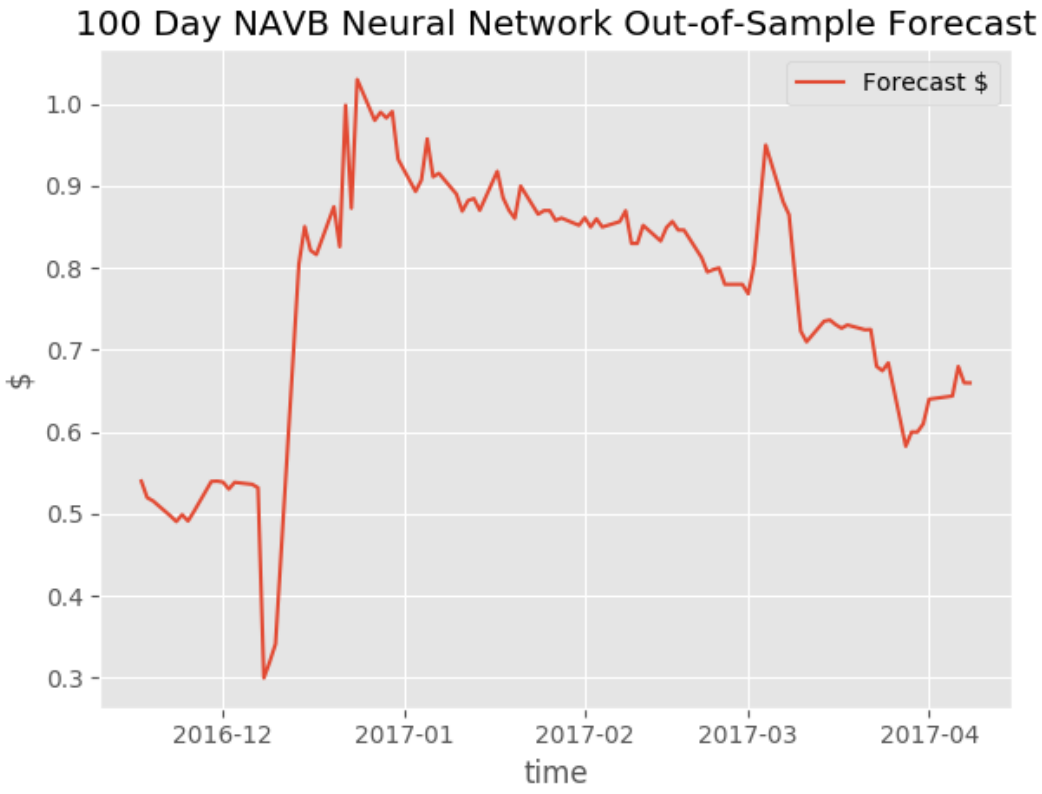
CDE scored a mean absolute error regression loss of 2.906, and a coefficient of determination of 0.816.



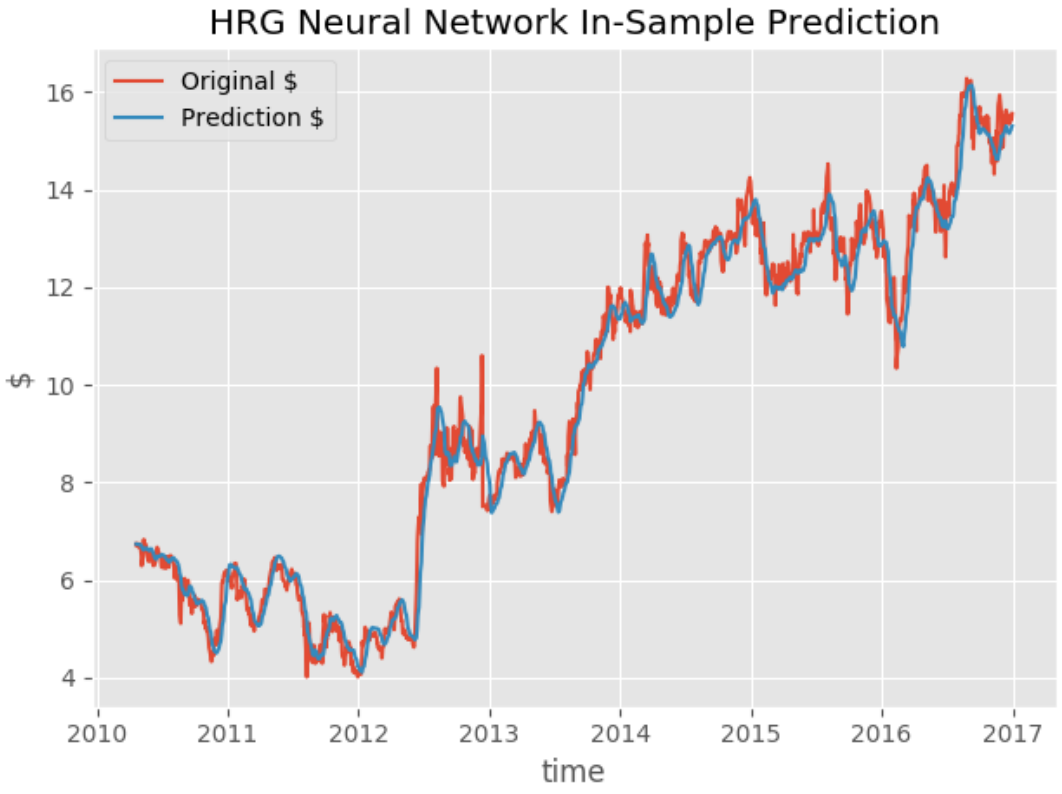


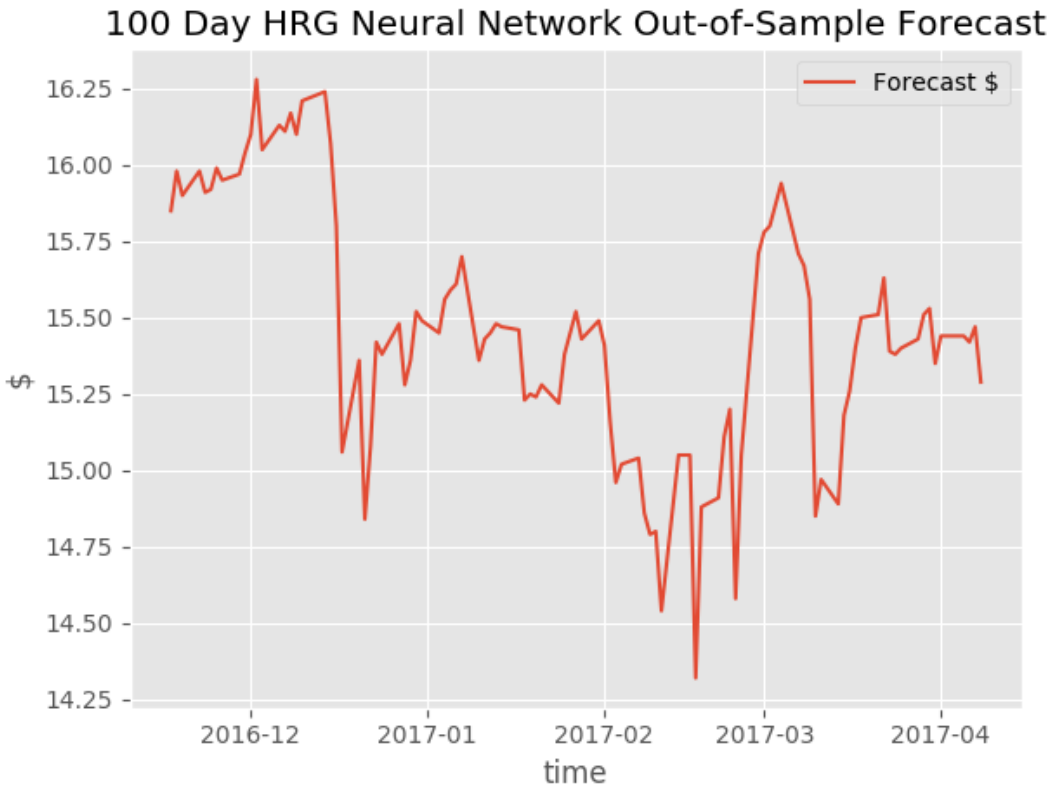
NAVB scored a mean absolute error regression loss of 0.422, and a coefficient of determination of 0.693.



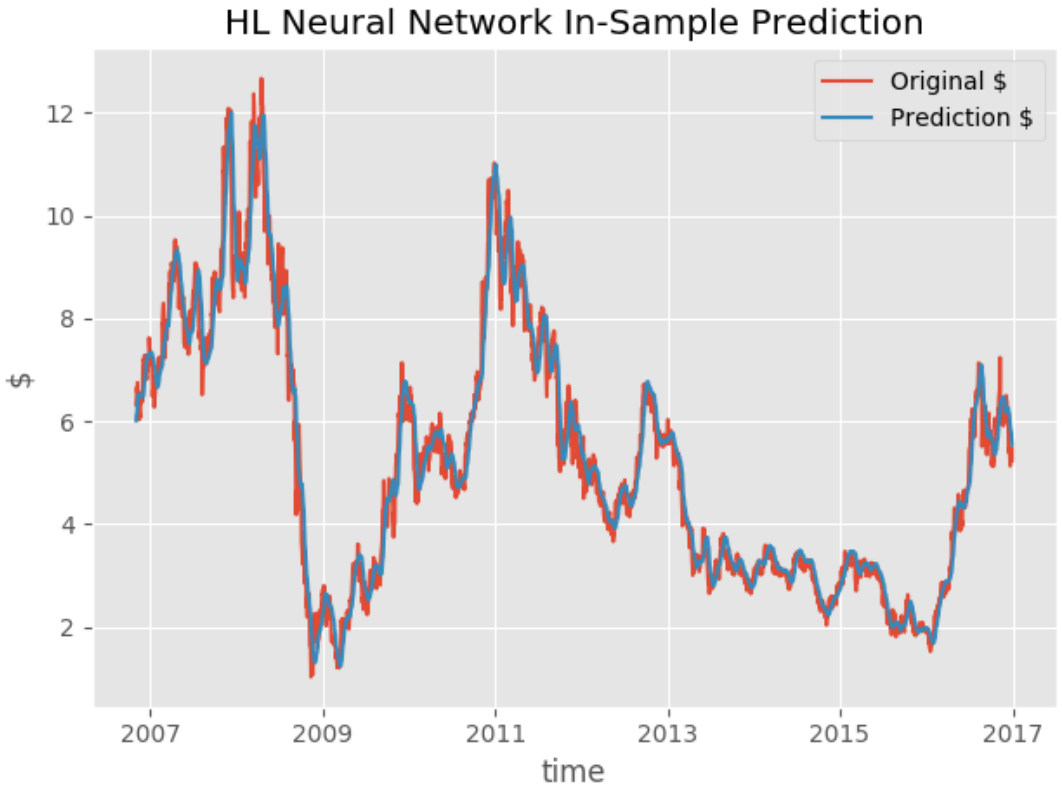


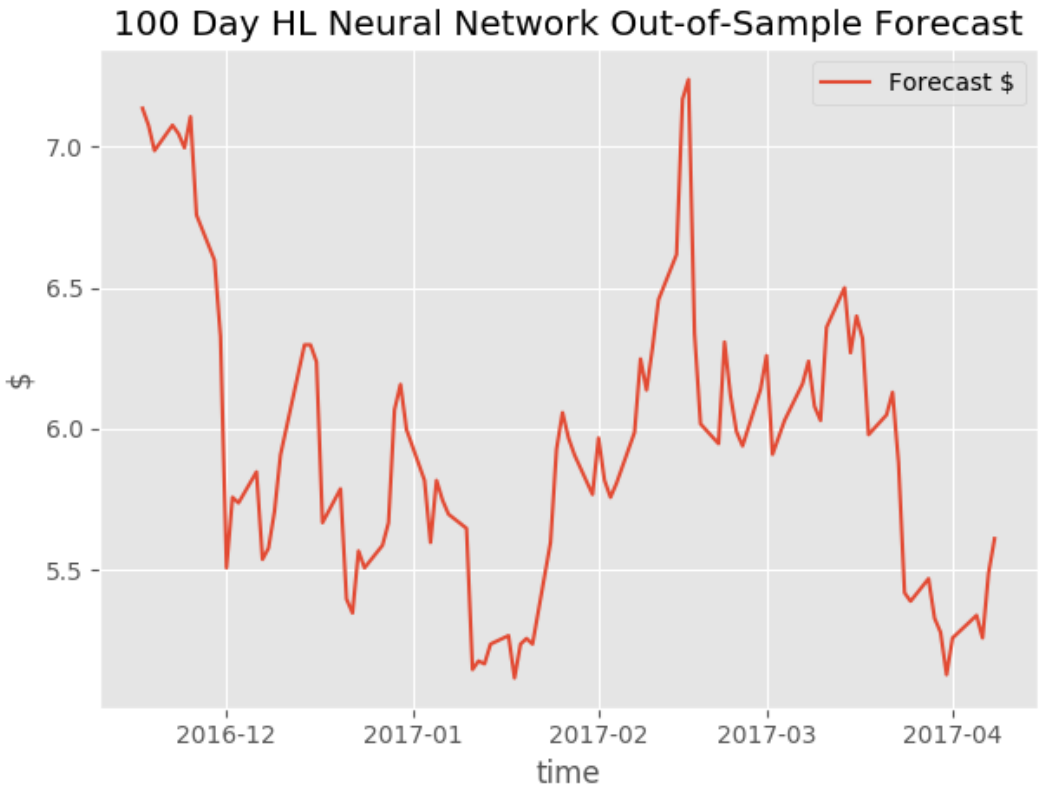
HRG scored a mean absolute error regression loss of 1.415, and a coefficient of determination of 0.464.





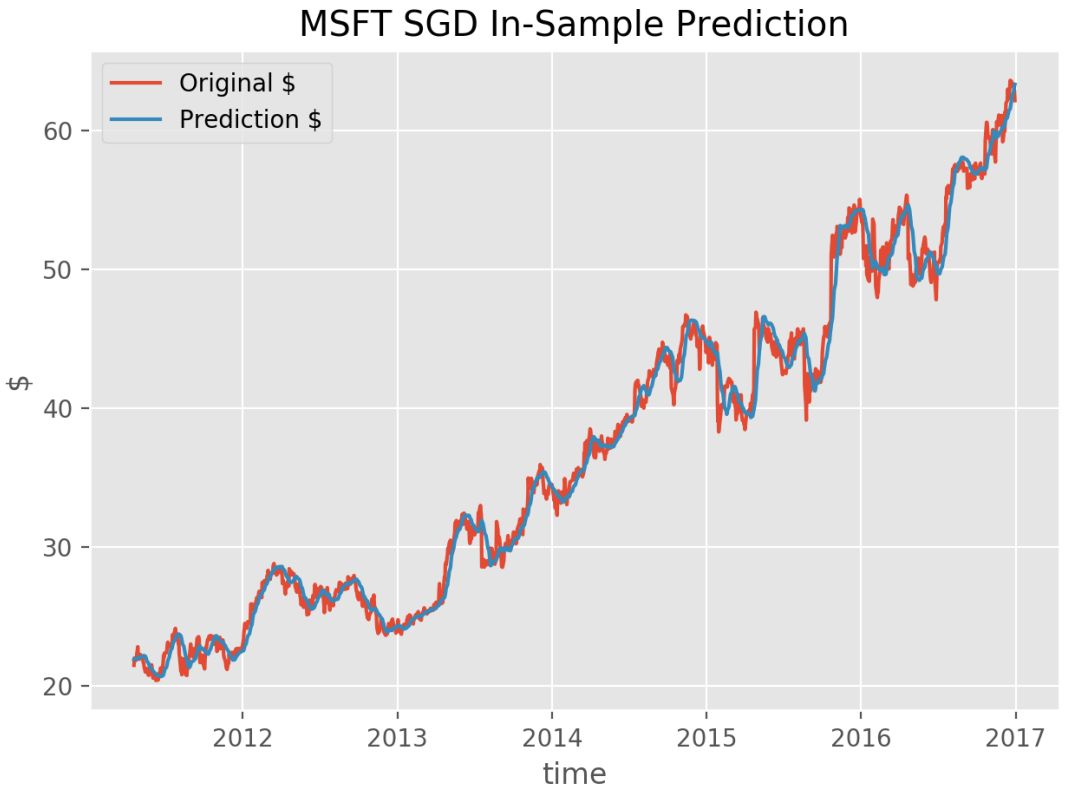
HL scored a mean absolute error regression loss of 0.506, and a coefficient of determination of 0.923.

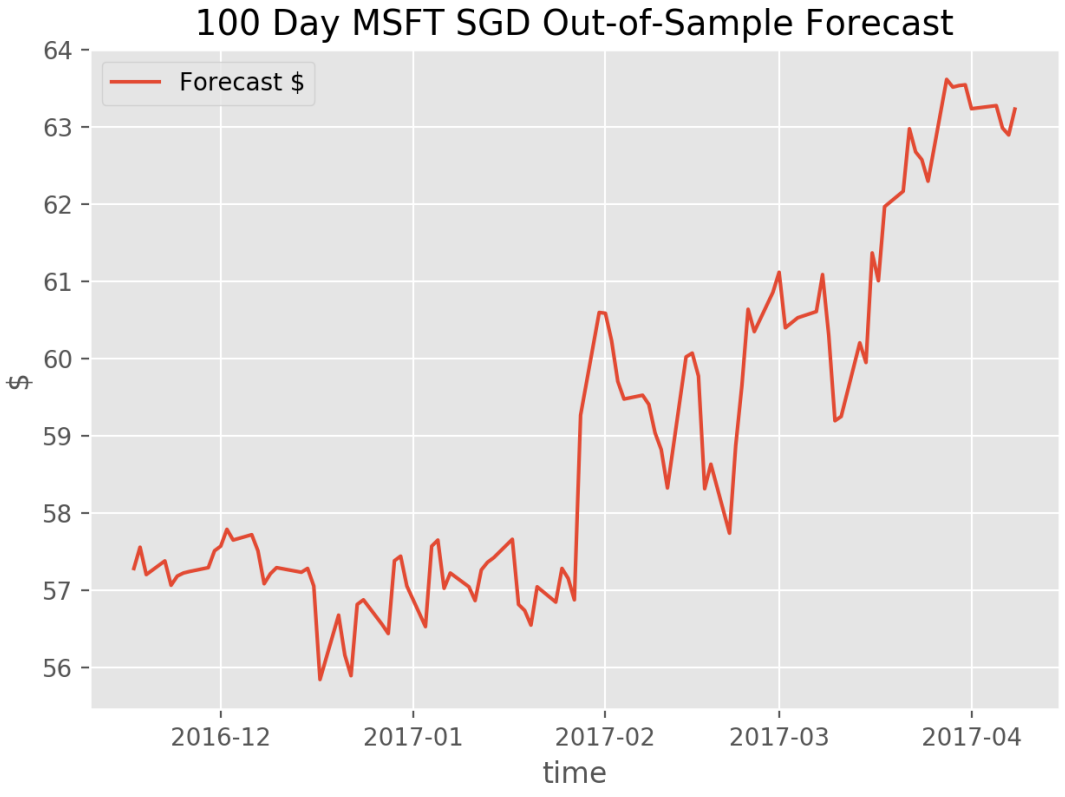




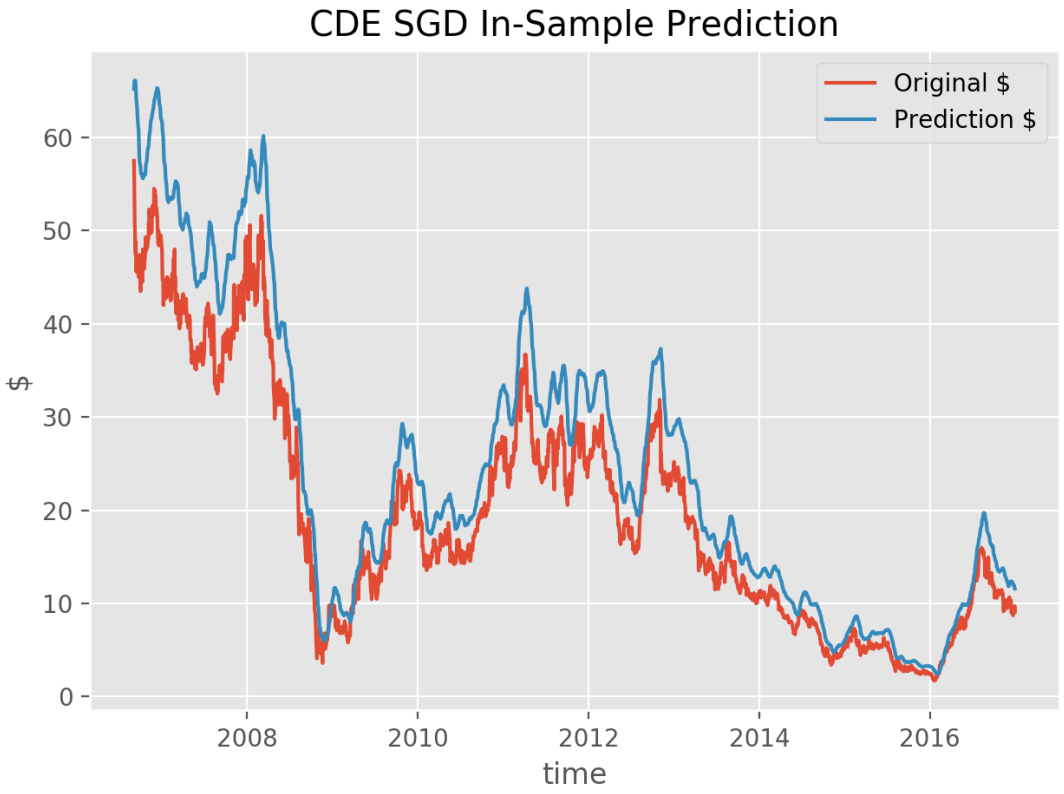
4.2.2.7 Stochastic Gradient Descent

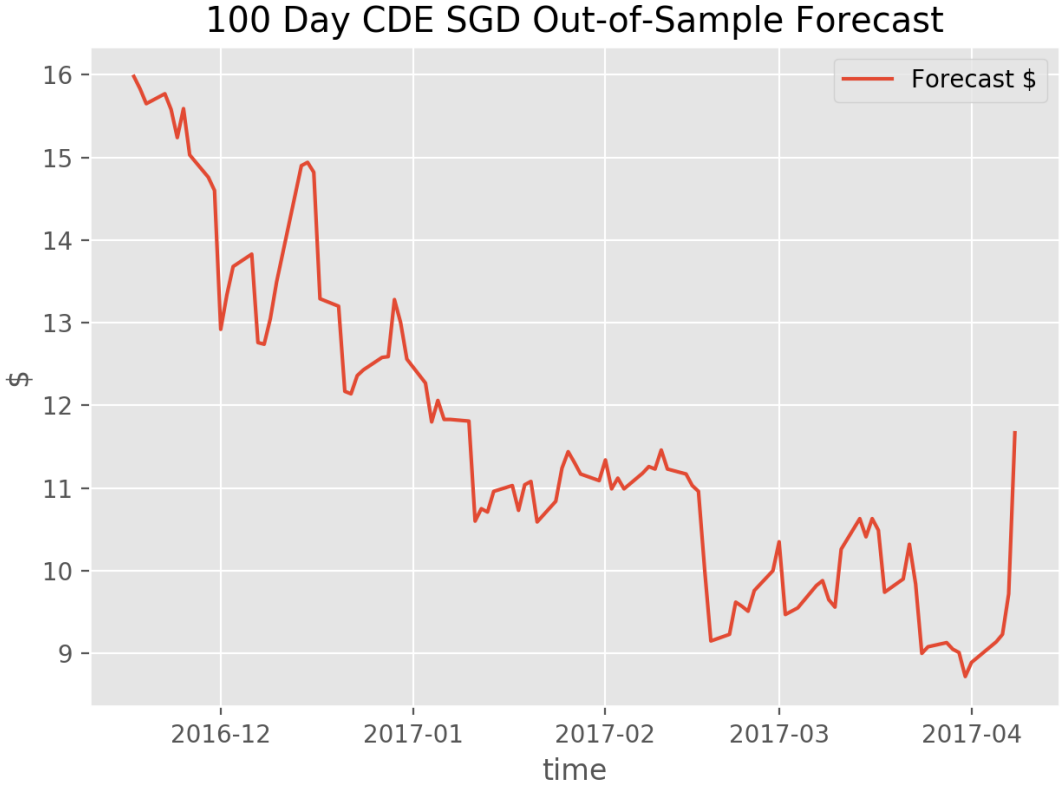
MSFT scored a mean absolute error regression loss of 0.829, and a coefficient of determination of 0.990.



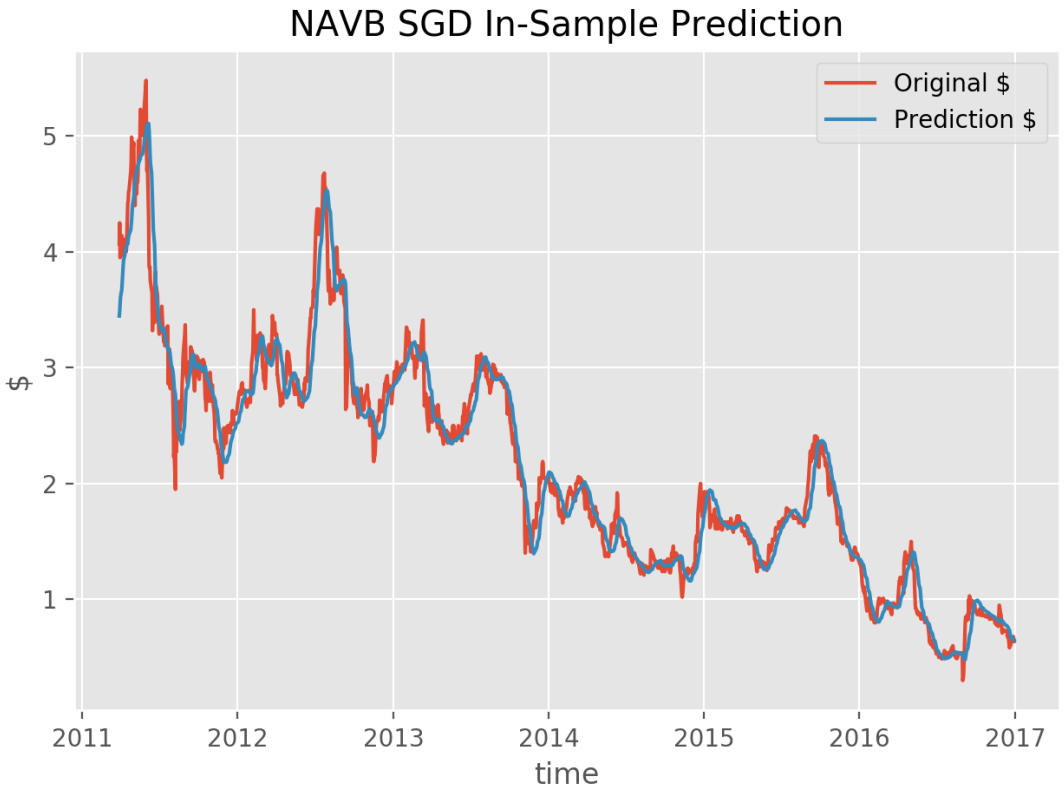


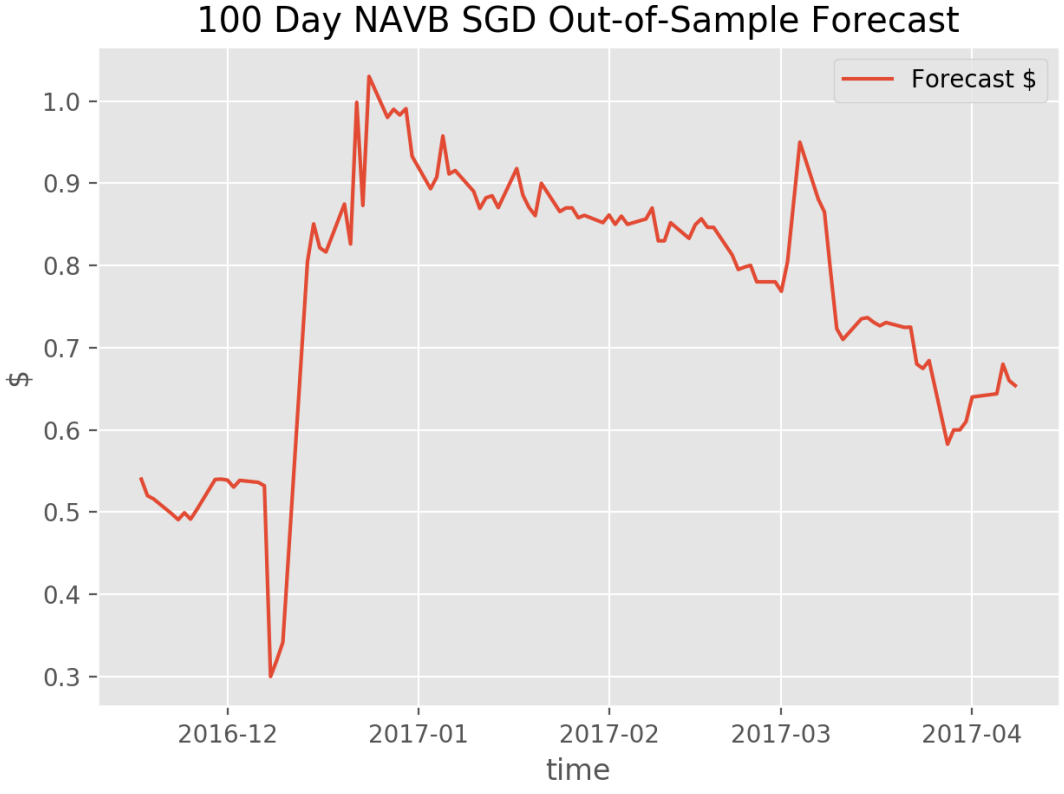
CDE scored a mean absolute error regression loss of 4.724, and a coefficient of determination of 0.788.



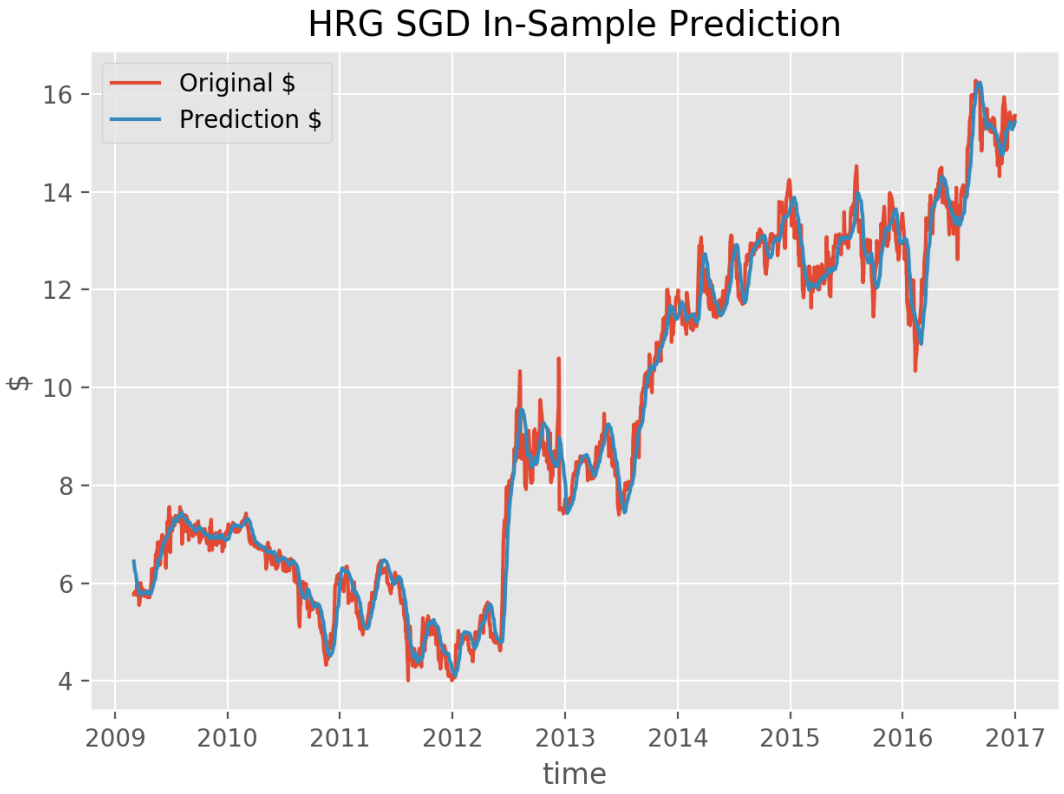


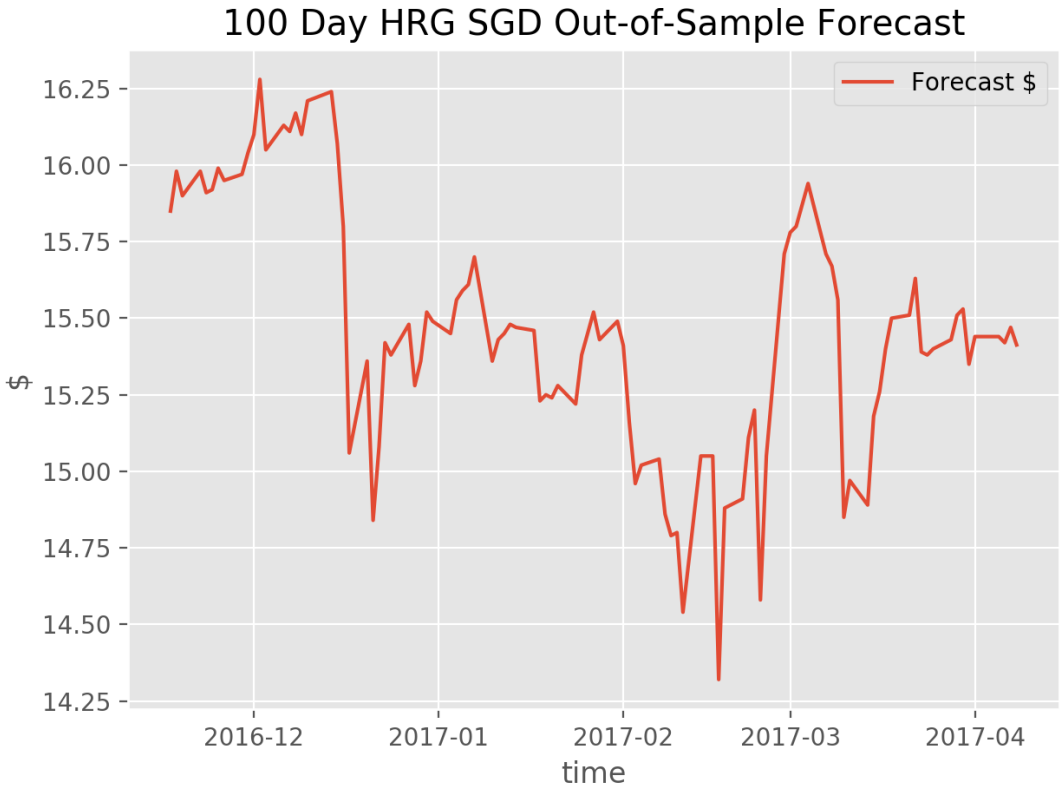
NAVB scored a mean absolute error regression loss of 0.142, and a coefficient of determination of 0.955.



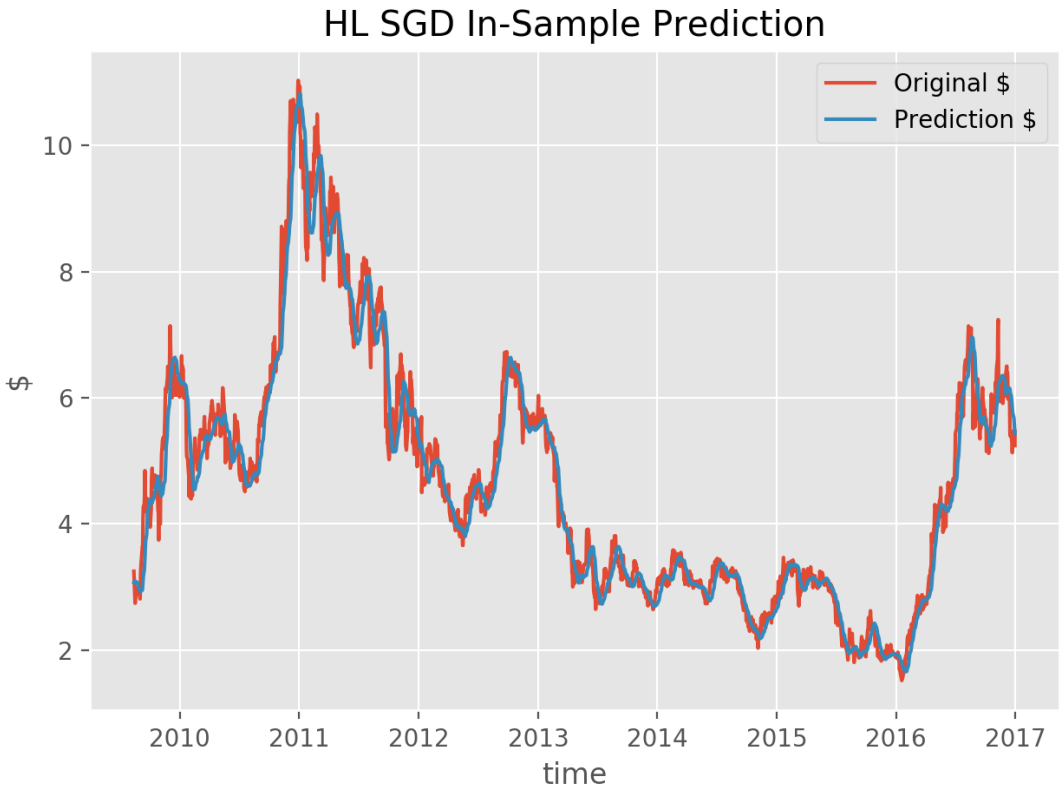


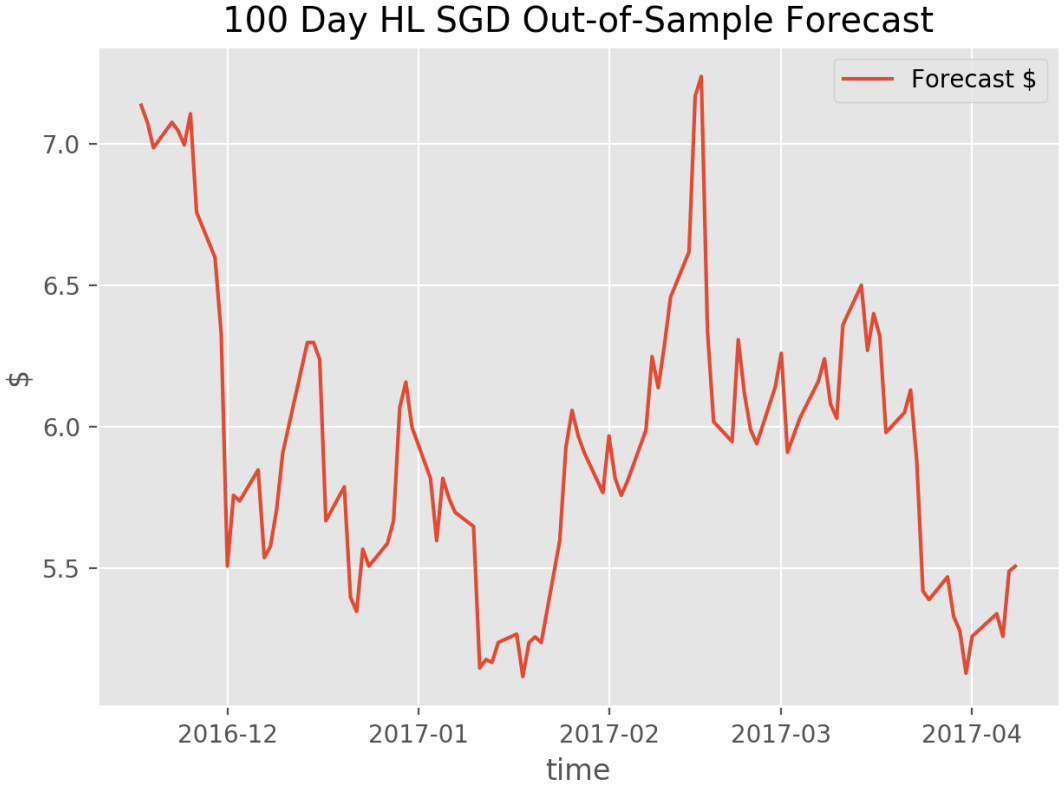
HRG scored a mean absolute error regression loss of 0.292, and a coefficient of determination of 0.987.





HL scored a mean absolute error regression loss of 0.275, and a coefficient of determination of 0.962.

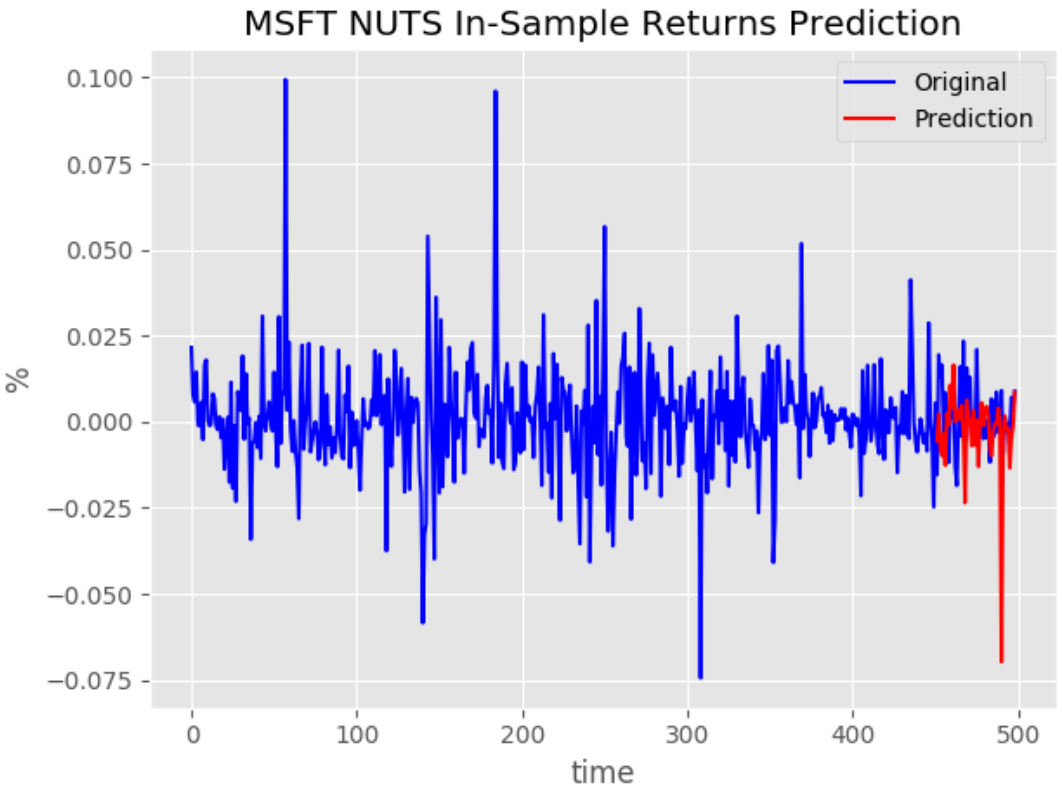

HL SGD In-Sample Prediction
Original $
Prediction $
10
8
6
4
2
$
2010
2011
2012
2013
2014
2015
2016
2017
time

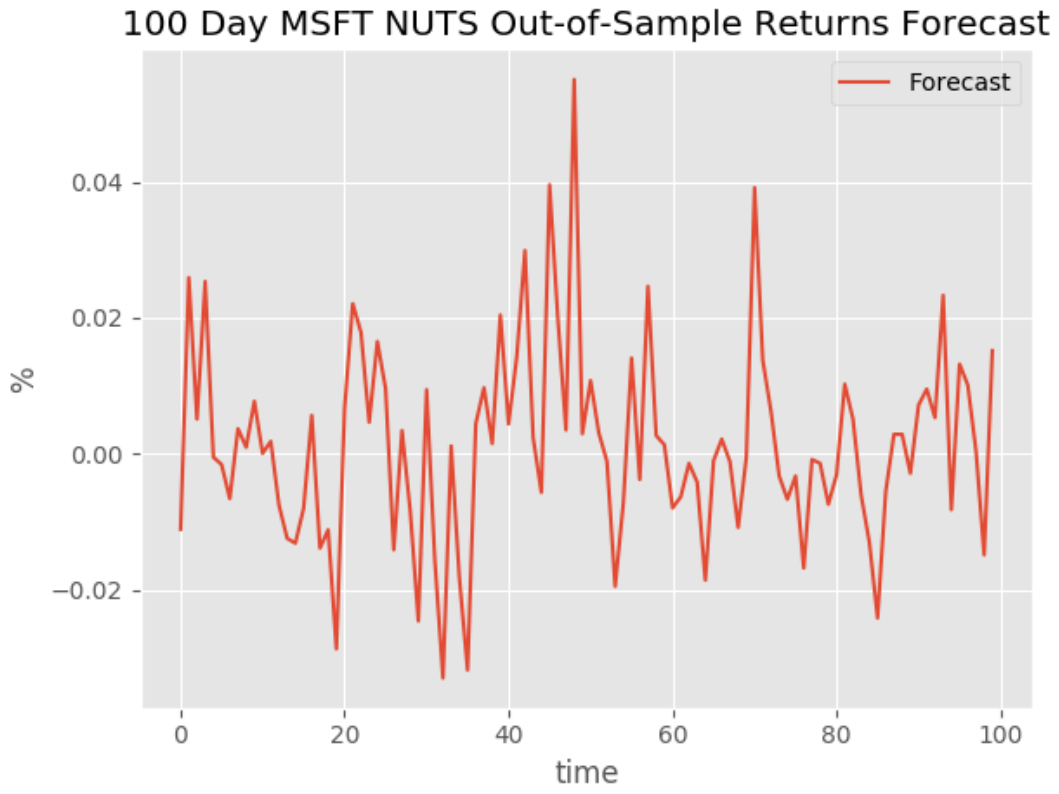

100 Day HL SGD Out-of-Sample Forecast
Forecast $
7.0
6.5
6.0
5.5
$
2016-12
2017-01
2017-02
2017-03
2017-04
time

4.3 Bayesian Statistics

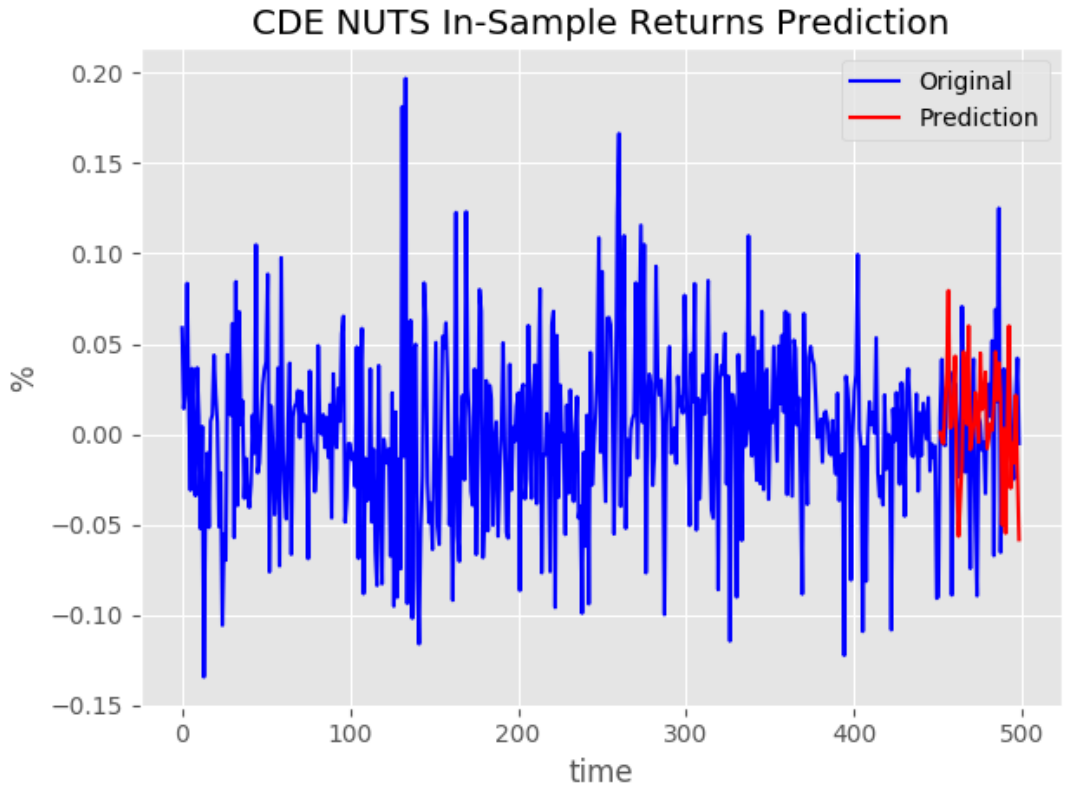
4.3.1 No-U-Turn Sampler (NUTS)

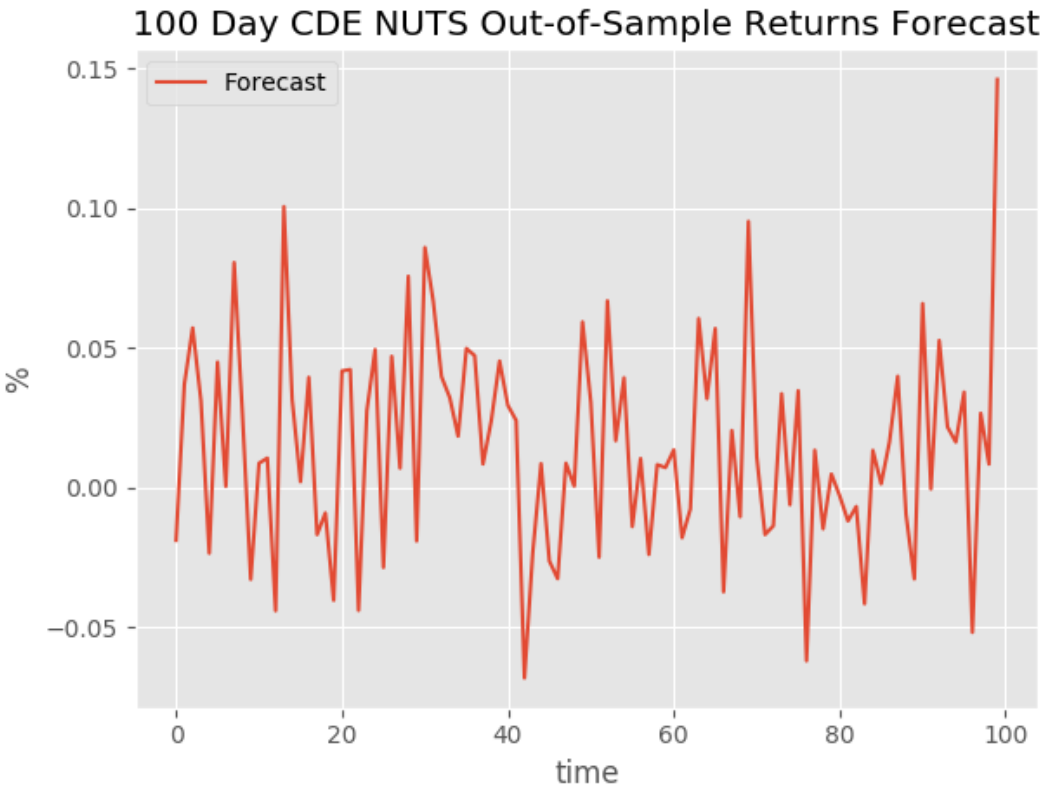
MSFT scored a mean absolute error regression loss of 5.606, and a coefficient of determination of 0.458.



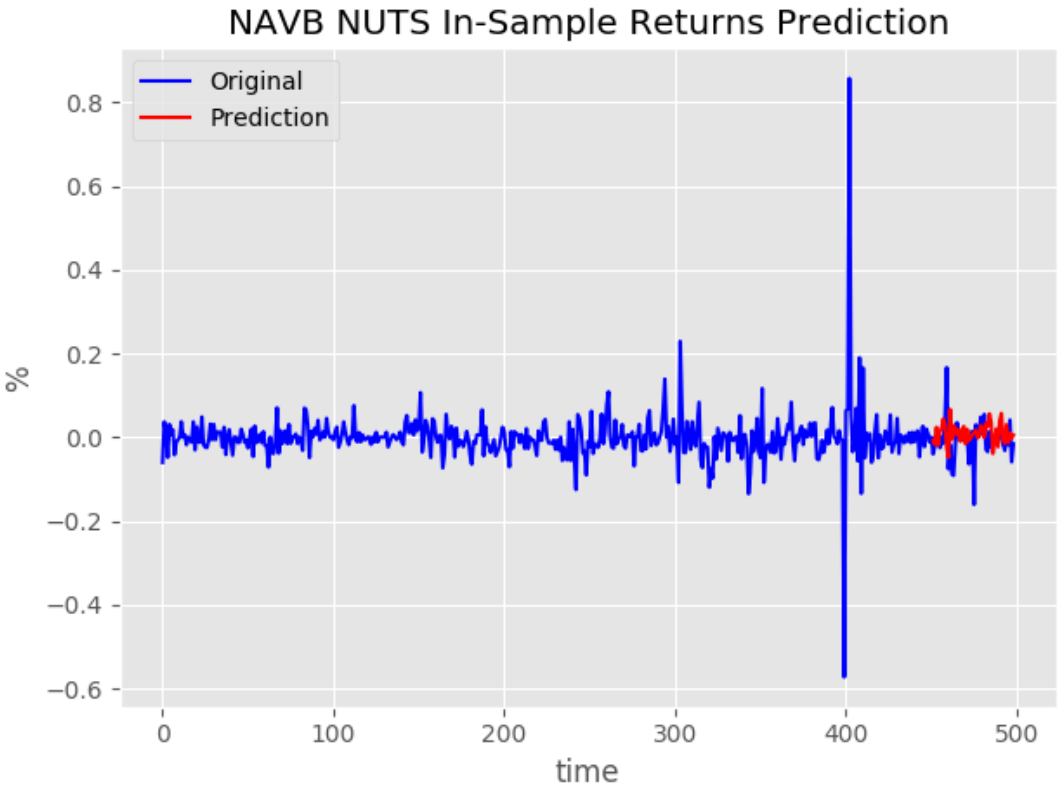


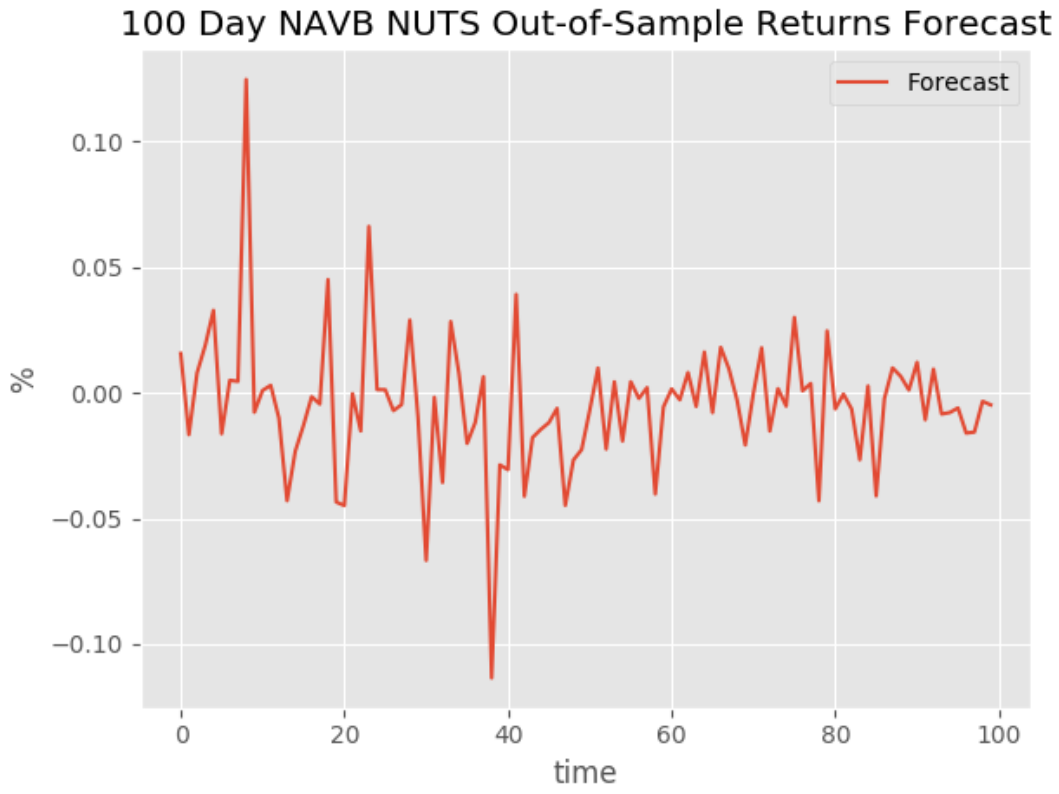
CDE scored a mean absolute error regression loss of 2.906, and a coefficient of determination of 0.816.



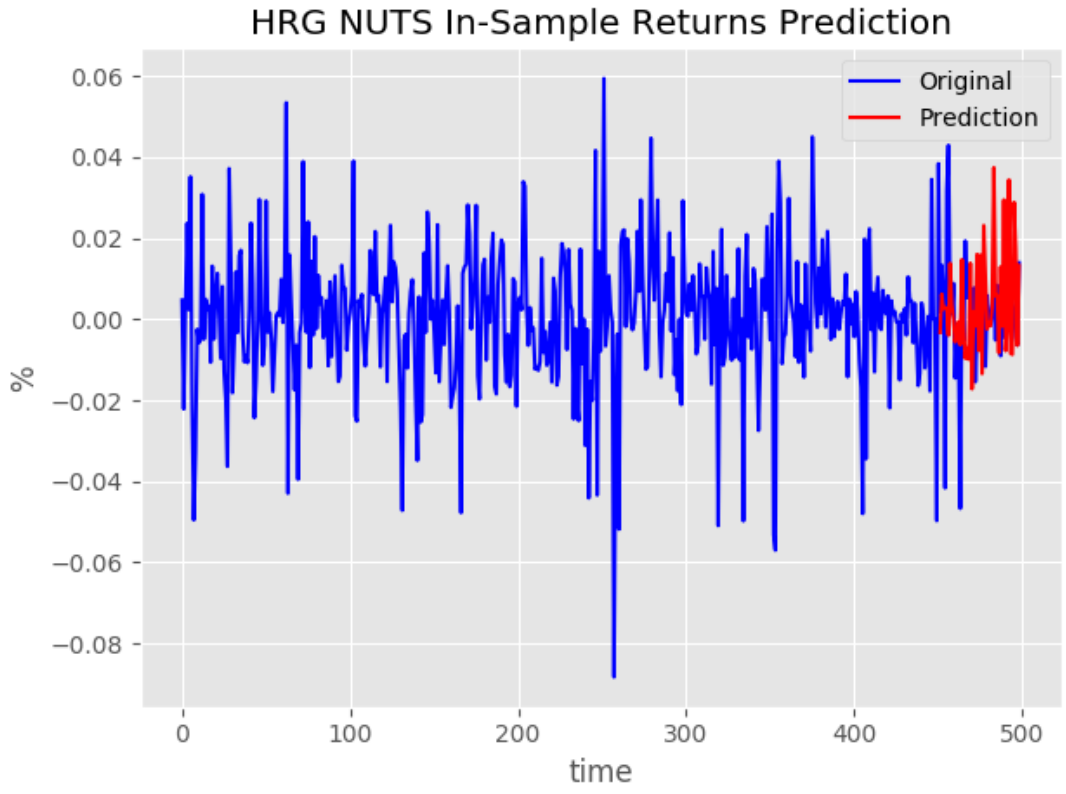


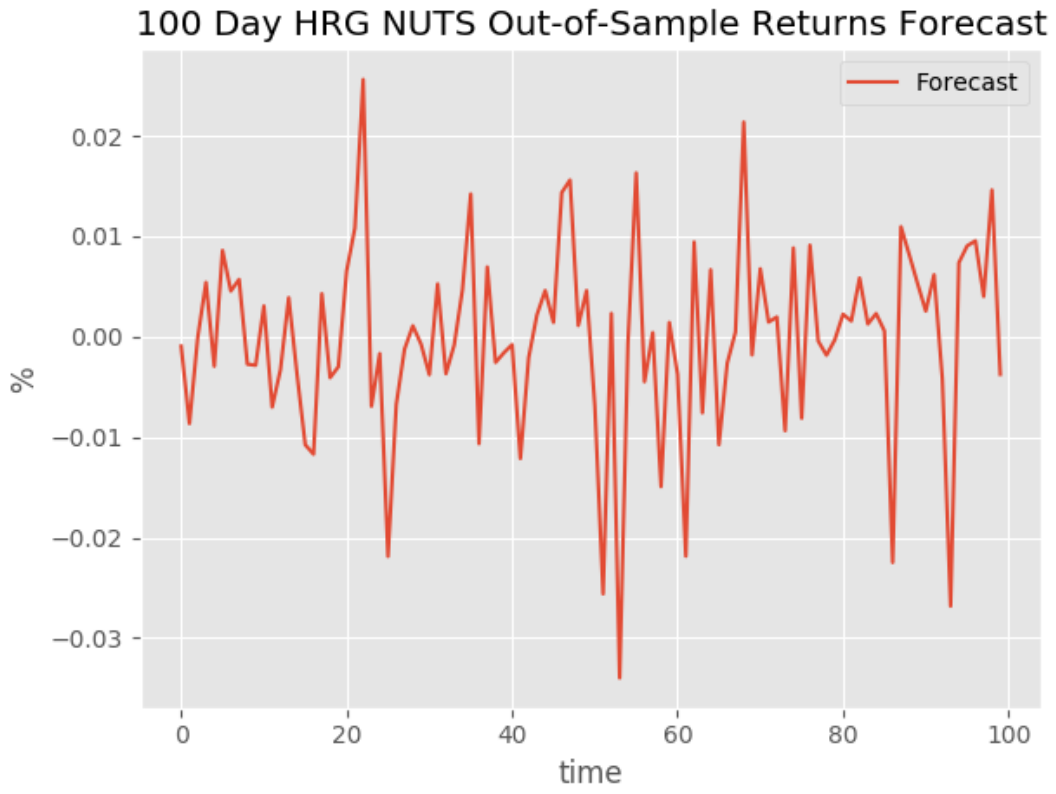
NAVB scored a mean absolute error regression loss of 0.422, and a coefficient of determination of 0.693.



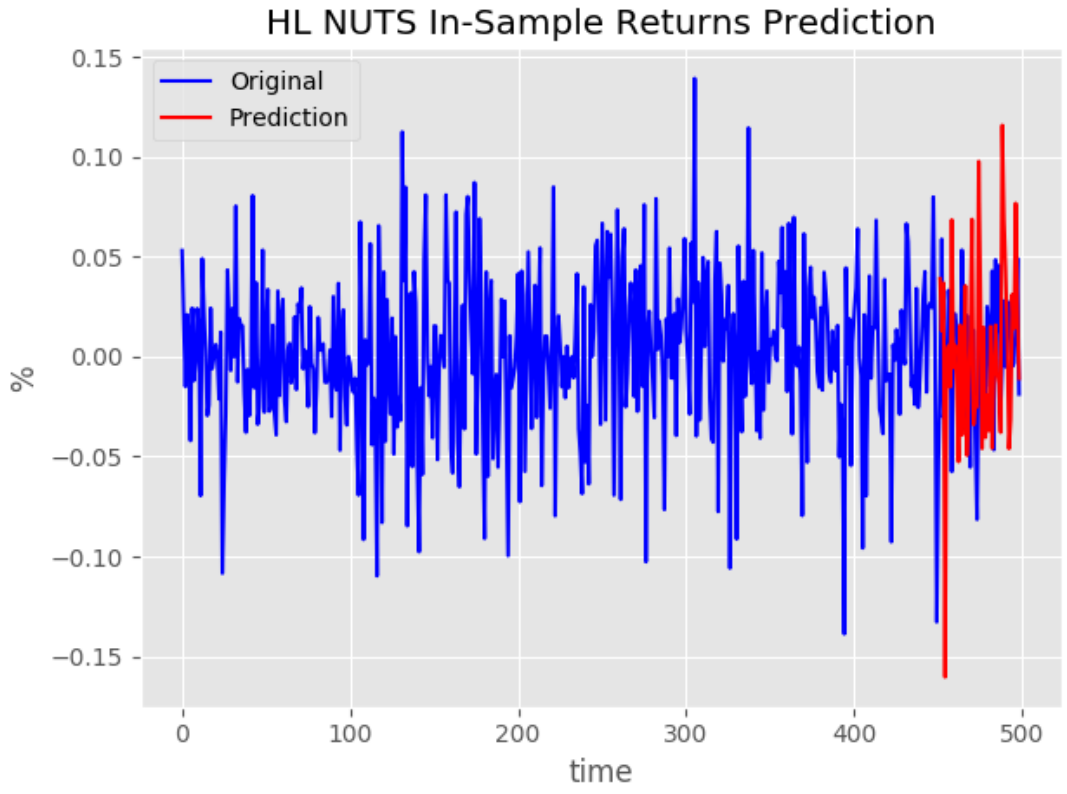


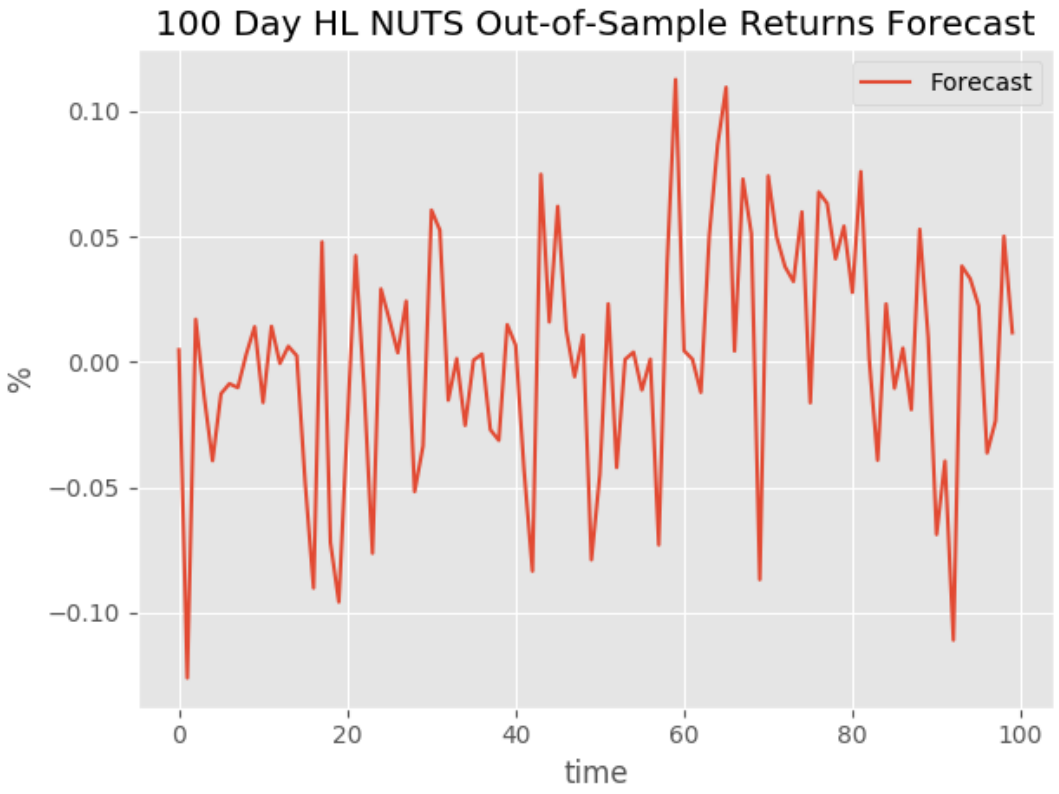
HRG scored a mean absolute error regression loss of 1.415, and a coefficient of determination of 0.464.





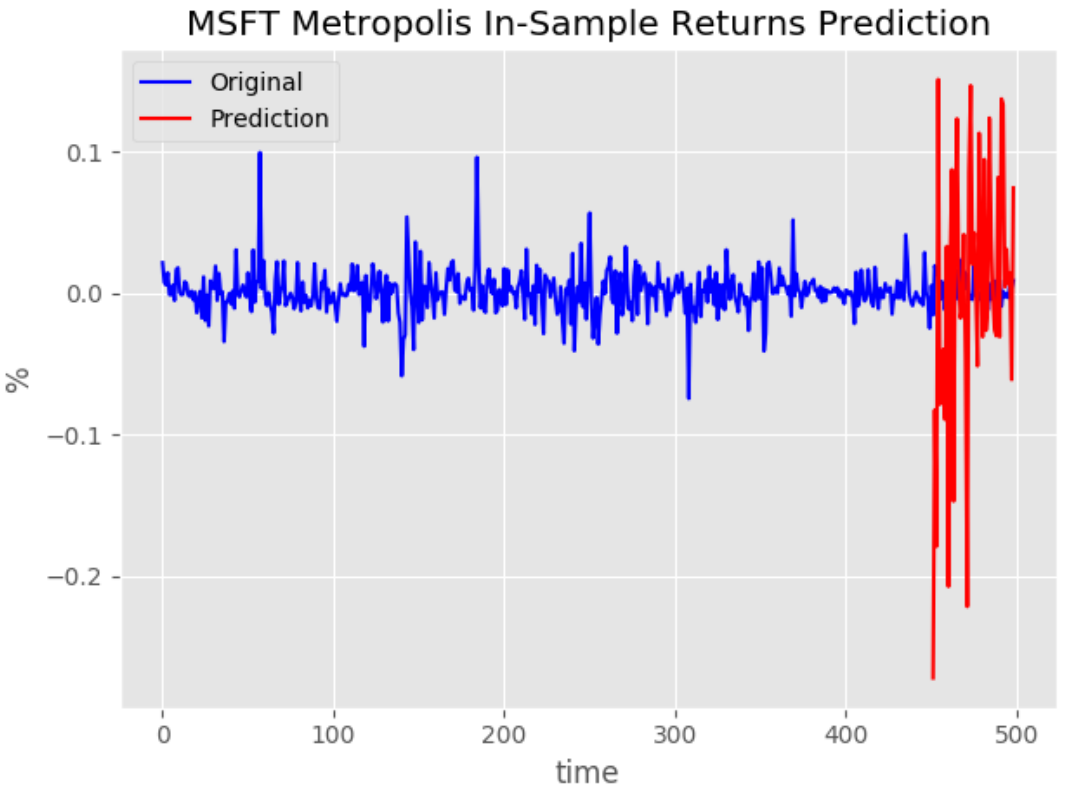
HL scored a mean absolute error regression loss of 0.506, and a coefficient of determination of 0.923.

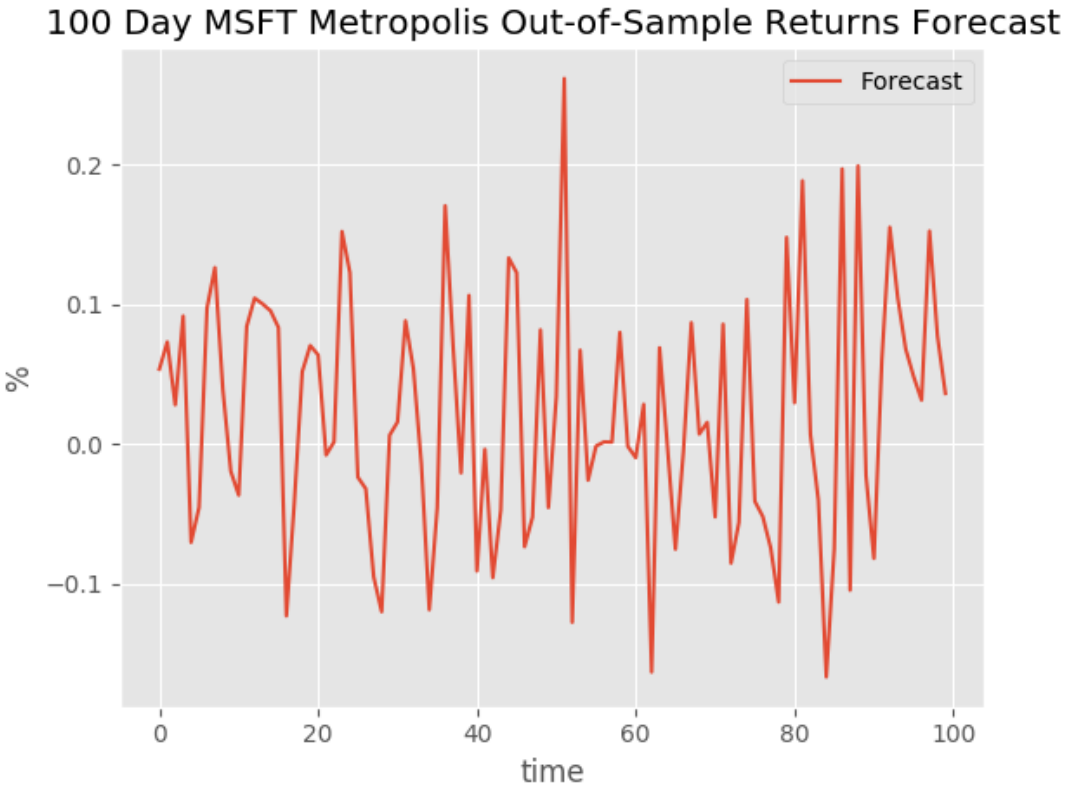




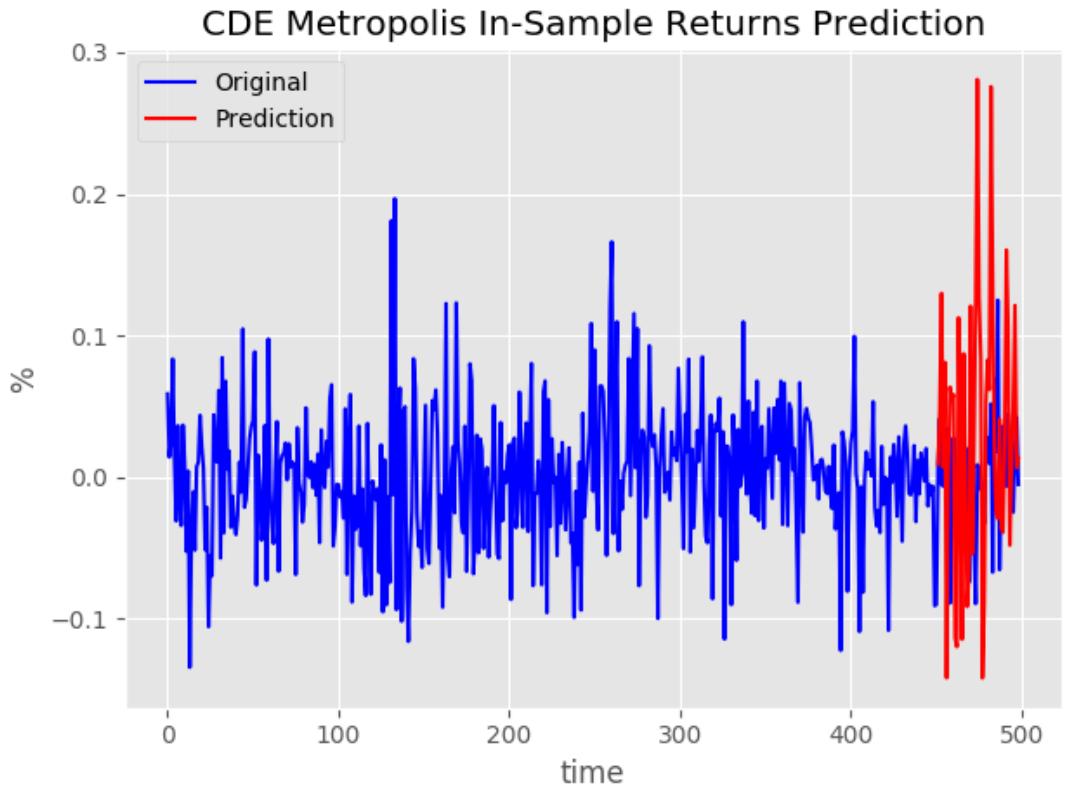
4.3.2 Metropolis-Hastings

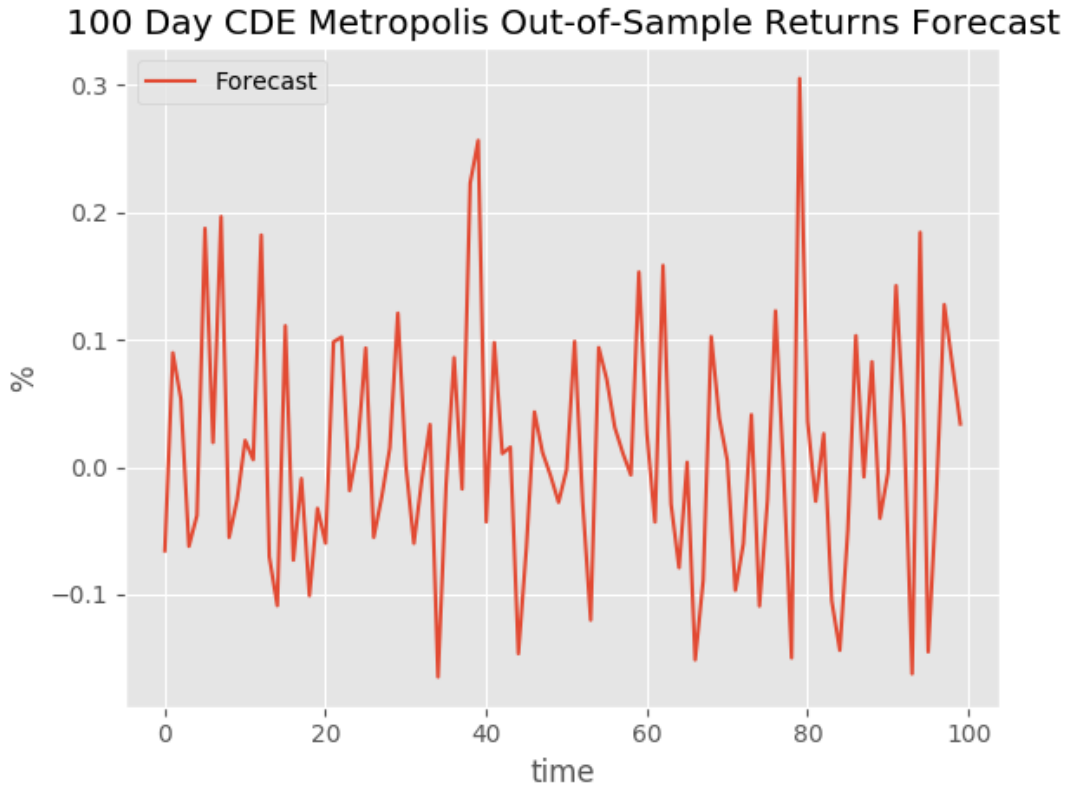
MSFT scored a mean absolute error regression loss of 5.606, and a coefficient of determination of 0.458.



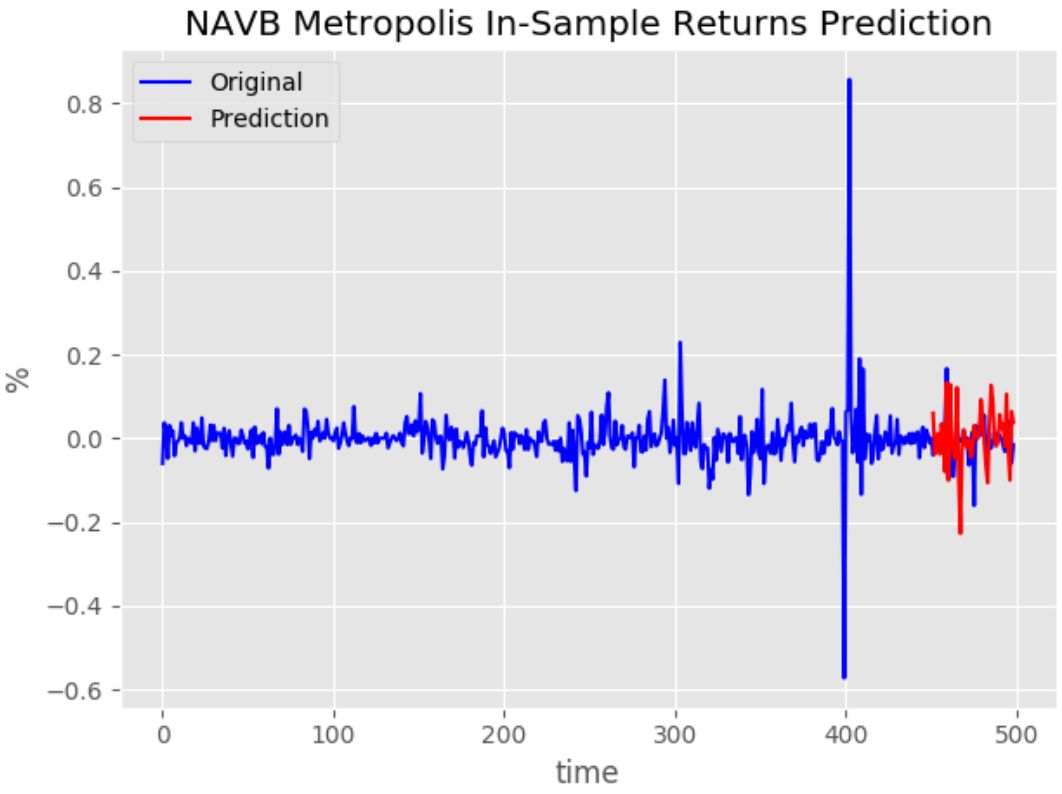


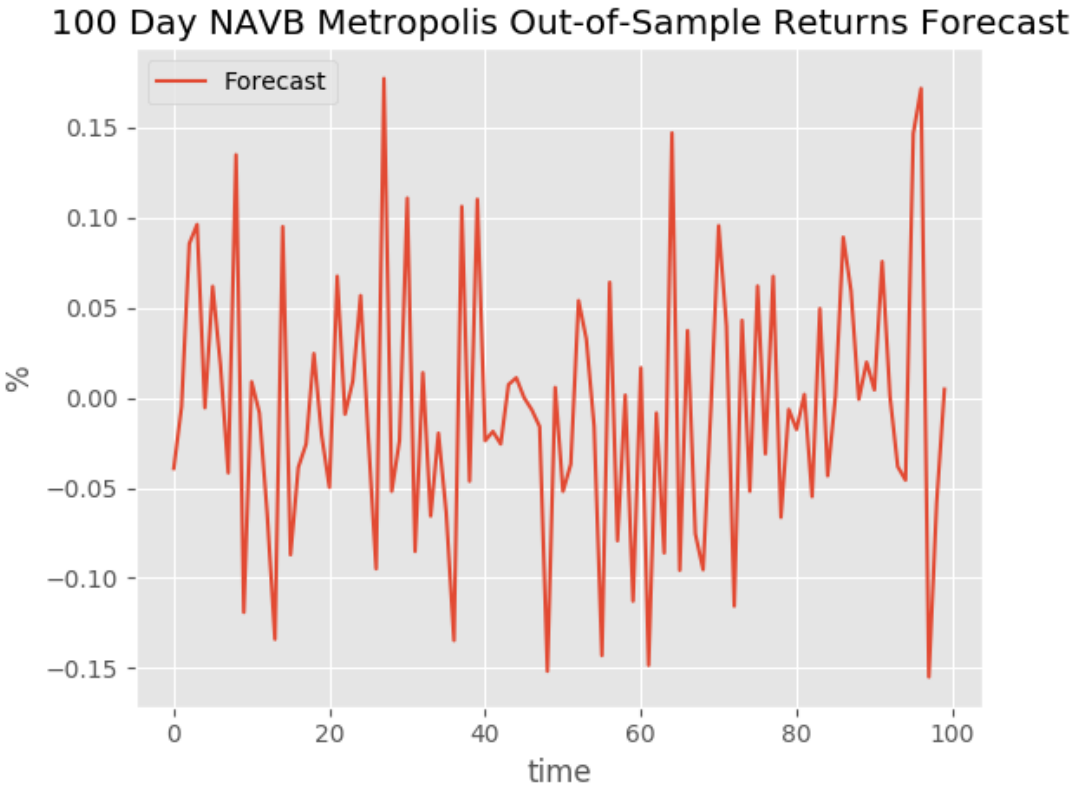
CDE scored a mean absolute error regression loss of 2.906, and a coefficient of determination of 0.816.



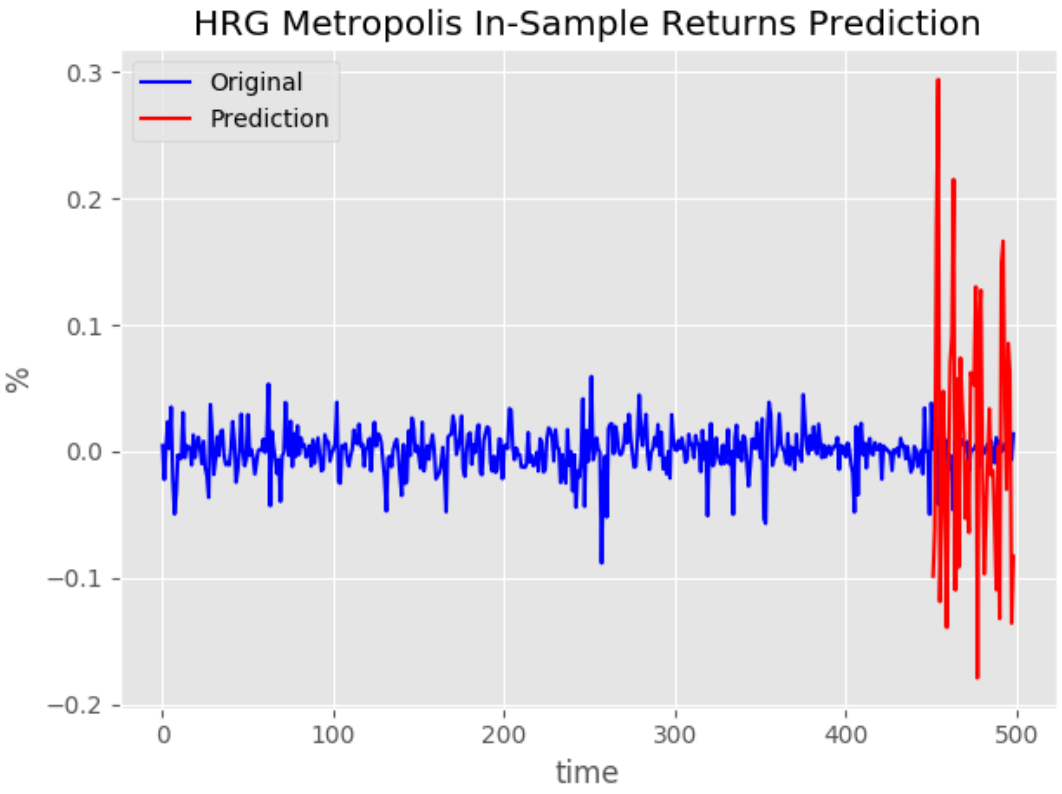


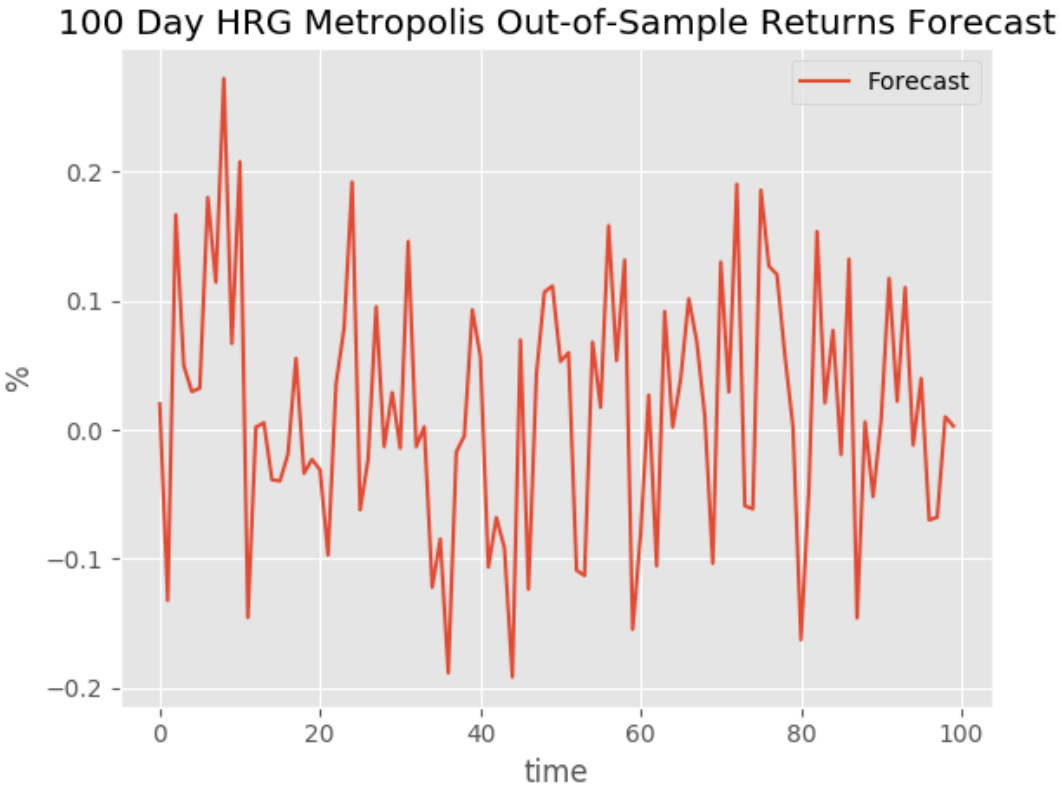
NAVB scored a mean absolute error regression loss of 0.422, and a coefficient of determination of 0.693.



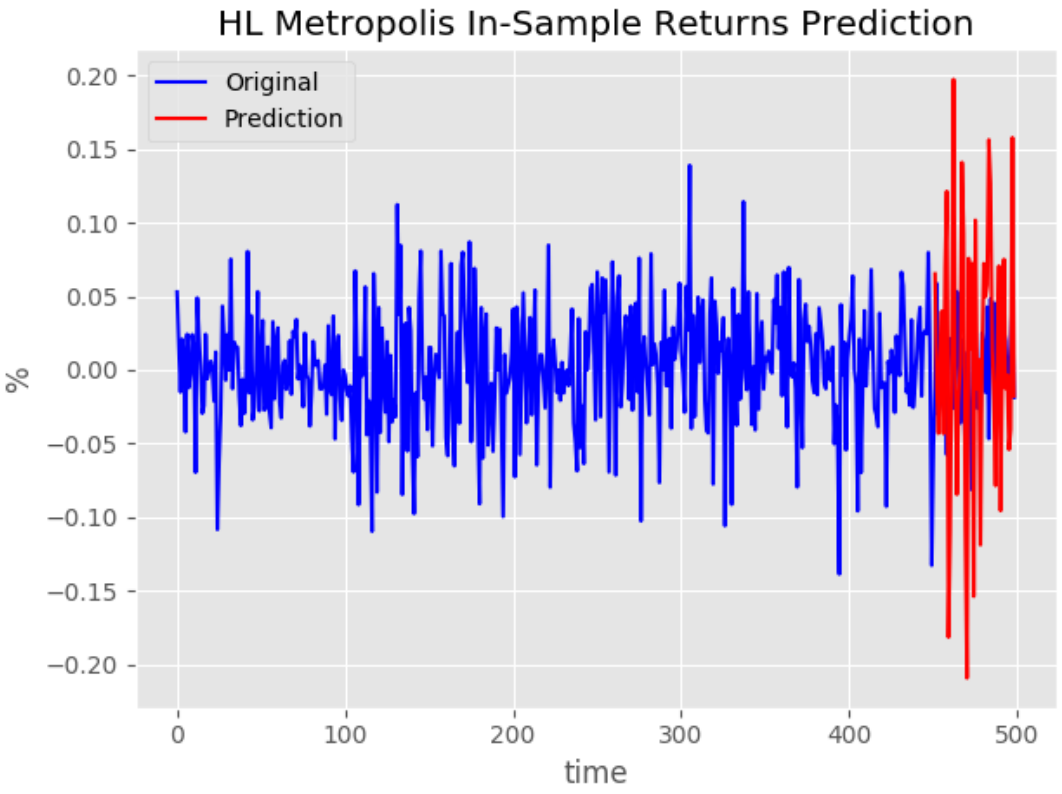


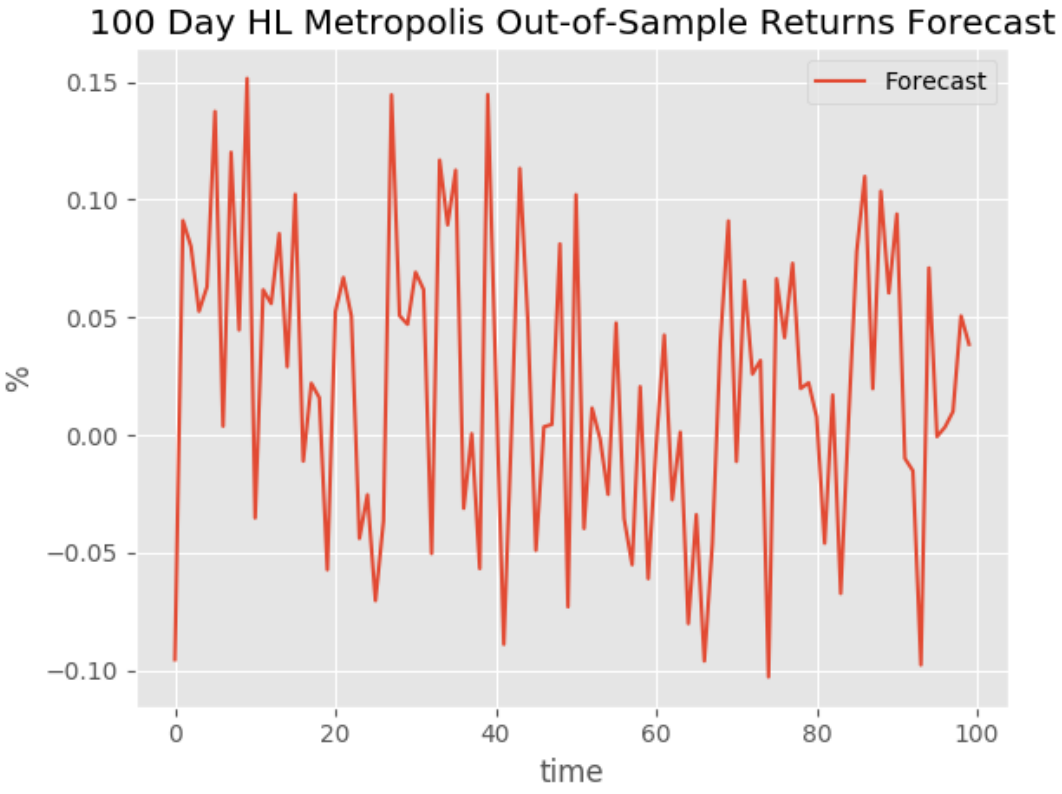
HRG scored a mean absolute error regression loss of 1.415, and a coefficient of determination of 0.464.





HL scored a mean absolute error regression loss of 0.506, and a coefficient of determination of 0.923.


HL Metropolis In-Sample Returns Prediction
Original
Prediction
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
%
0
100
200
300
400
500
time


100 Day HL Metropolis Out-of-Sample Returns Forecast
Forecast
0.15
0.10
0.05
0.00
−0.05
−0.10
%
0
20
40
60
80
100
time

Chapter 5

Discussion and Suggestions for Future Research

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam sodales tortor tortor. Nunc eget cursus dolor, id efficitur arcu. Maecenas posuere dictum nisi, non pulvinar nunc aliquam eu. Sed egestas, leo nec molestie hendrerit, nulla purus mattis lectus, eu rhoncus ipsum nulla vitae lectus. Integer mollis ligula ut risus maximus, ut ultrices tellus aliquet. Ut semper laoreet enim facilisis vulputate. Fusce ullamcorper a nisi iaculis pellentesque. Duis vel magna quis justo cursus fermentum vitae ac libero. Maecenas eget arcu et neque egestas scelerisque. Sed hendrerit at augue id interdum. Phasellus blandit tempus nunc ac feugiat. Vivamus egestas augue nec erat vestibulum elementum. Sed ac metus eu nunc consectetur mattis id ut lorem.

Chapter 6

Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam sodales tortor tortor. Nunc eget cursus dolor, id efficitur arcu. Maecenas posuere dictum nisi, non pulvinar nunc aliquam eu. Sed egestas, leo nec molestie hendrerit, nulla purus mattis lectus, eu rhoncus ipsum nulla vitae lectus. Integer mollis ligula ut risus maximus, ut ultrices tellus aliquet. Ut semper laoreet enim facilisis vulputate. Fusce ullamcorper a nisi iaculis pellentesque. Duis vel magna quis justo cursus fermentum vitae ac libero. Maecenas eget arcu et neque egestas scelerisque. Sed hendrerit at augue id interdum. Phasellus blandit tempus nunc ac feugiat. Vivamus egestas augue nec erat vestibulum elementum. Sed ac metus eu nunc consectetur mattis id ut lorem.

Appendix A

An Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus at pulvinar nisi. Phasellus hendrerit, diam placerat interdum iaculis, mauris justo cursus risus, in viverra purus eros at ligula. Ut metus justo, consequat a tristique posuere, laoreet nec nibh. Etiam et scelerisque mauris. Phasellus vel massa magna. Ut non neque id tortor pharetra bibendum vitae sit amet nisi. Duis nec quam quam, sed euismod justo. Pellentesque eu tellus vitae ante tempus malesuada. Nunc accumsan, quam in congue consequat, lectus lectus dapibus erat, id aliquet urna neque at massa. Nulla facilisi. Morbi ullamcorper eleifend posuere. Donec libero leo, faucibus nec bibendum at, mattis et urna. Proin consectetur, nunc ut imperdiet lobortis, magna neque tincidunt lectus, id iaculis nisi justo id nibh. Pellentesque vel sem in erat vulputate faucibus molestie ut lorem.

Quisque tristique urna in lorem laoreet at laoreet quam congue. Donec dolor turpis, blandit non imperdiet aliquet, blandit et felis. In lorem nisi, pretium sit amet vestibulum sed, tempus et sem. Proin non ante turpis. Nulla imperdiet fringilla convallis. Vivamus vel bibendum nisl. Pellentesque justo lectus, molestie vel luctus sed, lobortis in libero. Nulla facilisi. Aliquam erat volutpat. Suspendisse vitae nunc nunc. Sed aliquet est suscipit sapien rhoncus non adipiscing nibh consequat. Aliquam metus urna, faucibus eu vulputate non, luctus eu justo.

Donec urna leo, vulputate vitae porta eu, vehicula blandit libero. Phasellus eget massa et leo condimentum mollis. Nullam molestie, justo at pellentesque vulputate, sapien velit ornare diam, nec gravida lacus augue non diam. Integer mattis lacus id libero ultrices sit amet mollis neque molestie. Integer ut leo eget mi volutpat congue. Vivamus sodales, turpis id venenatis placerat, tellus purus adipiscing magna, eu aliquam nibh dolor id nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed cursus convallis quam nec vehicula. Sed vulputate neque eget odio fringilla ac sodales urna feugiat.

Phasellus nisi quam, volutpat non ullamcorper eget, congue fringilla leo. Cras et erat et nibh placerat commodo id ornare est. Nulla facilisi. Aenean pulvinar scelerisque eros eget interdum. Nunc pulvinar magna ut felis varius in hendrerit dolor accumsan. Nunc pellentesque magna quis magna bibendum non laoreet erat tincidunt. Nulla facilisi.

Duis eget massa sem, gravida interdum ipsum. Nulla nunc nisl, hendrerit sit amet commodo vel, varius id tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor est. Suspendisse ultrices tincidunt metus eget accumsan. Nullam facilisis, justo vitae convallis sollicitudin, eros augue malesuada metus, nec sagittis diam nibh ut sapien. Duis blandit lectus vitae lorem aliquam nec euismod nisi volutpat. Vestibulum ornare dictum tortor, at faucibus justo tempor non. Nulla facilisi. Cras non massa nunc, eget euismod purus. Nunc metus ipsum, euismod a consectetur vel, hendrerit nec nunc.

Bibliography

[1] Harry Markowitz. Portfolio selection. *American Finance Association*, 1952.

[2] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2016.

[3] Tobias Moskowitz, Yao Hua Ooi, and Lasse H. Pedersen. Time series momentum. *Chicago Booth Research*, September 2011.

[4] S. Radha and M. Thenmozhi. Forecasting short term interest rates using arma, arma-garch and arma-egarch models. *Indian Institute of Capital Markets 9th Capital Markets Conference Paper*, January 2006.

[5] Natalia Abrosimova, Gishan Dissanaike, and Dirk Linowski. Testing weak-form efficiency of the russian stock market. In *EFA 2002 Berlin Meetings*, 2002.

[6] Ali F. Darrat and Maosen Zhong. On testing the random walk hypothesis: A model-comparison approach, 2001.

[7] Thomas H. Cormen and Chales E. Leiserson. *Introduction to Algorithms*. The MIT Press, 2009.

[8] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[9] Punniyamoorthy Murugesan and Jose Joy Thoppan. Detection of stock price manipulation using discriminant analysis, June 2012.

[10] Manish Kumar and M. Thenmozhi. Forecasting stock index movement: A comparison of support vector machines and random forest, June 2016.

[11] Zura Kakushadze. Mean-reversion and optimization mean-reversion and optimization. *Journal of Asset Management*, 16(1):14–40, 2015.

[12] Germán G. Creamer and Yoav Freund. Automated trading with boosting and expert weighting. *Quantitative Finance*, Vol. 4(No. 10):pp. 401–420, April 2010.

[13] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. CRC Press, 2014.

[14] Savvakis C. Savvides. Risk analysis in investment appraisal risk analysis in investment appraisal risk analysis in investment appraisal. *Project Appraisal Journal*, Vol. 9(No. 1), March 1994.

[15] Leif B. G. Andersen. Efficient simulation of the heston stochastic volatility model, 2007.

[16] David Blanchett, Michael S. Finke, and Wade D. Pfau. Asset valuations and safe portfolio withdrawal rates, 2013.

[17] Andrea Gamba. Real options valuation: A monte carlo approach, 2003.