

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

June 2017

Declaration of Authorship

I, Gabriel Gauci Maistre, declare that this thesis titled, ‘Data Driven Automated Algorithmic Trading’ 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

Various existing stock market price forecasting methods were analysed in this report. Three methods were applied towards the problem making use of Technical Analysis, these were Time Series Analysis, Machine Learning, and Bayesian Statistics. Through the results of this report, it was found that the Efficient Market Hypothesis remains true, that past data does not contain enough useful information to forecast future prices and gain an advantage over the market. However, the results proved that Technical Analysis and Machine Learning could still be used to guide an investors decision. It was also found that the Random Walk Hypothesis was not necessarily true, as some stocks showed signs of auto and partial correlation. A common application of technical analysis was demonstrated and shown to produce limited useful information in beating the market. Based on the findings, a number of automated trading algorithms were developed using machine learning and backtested to determine their effectiveness.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor, Alan Gatt, for the patient guidance, encouragement, and advice he has provided throughout my time as his student.

I would also like to thank Luke Vella Critien, for guiding me towards the right path in the early stages of my research and for recommending Alan Gatt as my tutor.

My gratitude is also extended to Emma Galea and Miguel Attard for their valuable input while carrying out my research.

Completing this work would have been all the more difficult were it not for the support and friendship provided by the other members of the Malta College of Arts, Sciences, and Technology, and the Institute of Information and Technology. I am indebted to them for their help.

Finally, I would like to thank my family who have supported me all throughout the final year of my bachelor's.

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	ix
List of Tables	xiv
Abbreviations	xv
1 Introduction	1
1.1 Project goals and scope	1
1.1.1 Efficient Market Hypothesis	1
1.1.2 Random Walk Hypothesis	2
2 Background Theory	3
2.1 Time Series Analysis	3
2.1.1 Auto Regressive Moving Average (ARMA)	4
2.1.2 Auto Regressive Integrated Moving Average (ARIMA)	4
2.2 Statistical Machine Learning	6
2.2.1 Classification	6
2.2.2 Support Vector Machines	7
2.2.3 Regression	7
2.2.4 Decision Trees	8
2.3 Bayesian Statistics	8
2.3.1 Markov Chain Monte Carlo (MCMC)	9
3 Experimental Setup	11
3.1 Data Tidying	11
3.2 Stock Selection	11
3.3 Time Series Analysis	11
3.3.1 Random Walk	12
3.3.2 Ordinary Least Squares (OLS)	12

3.3.3	Auto Regressive (AR)	12
3.3.4	Moving Average (MA)	12
3.3.5	Auto Regressive Moving Average (ARMA)	12
3.3.6	Auto Regressive Integrated Moving Average (ARIMA)	13
3.4	Machine Learning	13
3.4.1	Classification	13
3.4.1.1	Decision Tree	13
3.4.1.2	Boosted Decision Tree	14
3.4.1.3	Support Vector Machine (SVM)	14
3.4.1.4	Random Forest	14
3.4.1.5	K-Nearest Neighbour	15
3.4.1.6	Logistic Regression	15
3.4.1.7	Bernoulli Naive Bayes	15
3.4.1.8	Gaussian Naive Bayes	16
3.4.1.9	Neural Network	16
3.4.1.10	Stochastic Gradient Descent	16
3.4.2	Regression	17
3.4.2.1	Decision Tree	17
3.4.2.2	Boosted Decision Tree	18
3.4.2.3	Random Forest	18
3.4.2.4	Linear Regression	18
3.4.2.5	Neural Network	18
3.4.2.6	Stochastic Gradient Descent	18
3.5	Bayesian Statistics	19
3.5.1	No-U-Turn Sampler (NUTS)	19
3.5.2	Metropolis-Hastings	19
3.6	Strategy	19
3.6.1	Classification	19
3.6.2	Regression	20
4	Research Findings	21
4.1	Time Series Analysis	22
4.1.1	Random Walk	22
4.1.2	Ordinary Least Squares (OLS)	27
4.1.3	Auto Regressive (AR)	35
4.1.4	Moving Average (MA)	47
4.1.5	Auto Regressive Moving Average (ARMA)	60
4.1.6	Auto Regressive Integrated Moving Average (ARIMA)	72
4.2	Machine Learning	85
4.2.1	Classification	85
4.2.1.1	Decision Tree	85
4.2.1.2	Boosted Decision Tree	85
4.2.1.3	Support Vector Machine (SVM)	85
4.2.1.4	Random Forest	86
4.2.1.5	K-Nearest Neighbour	86
4.2.1.6	Logistic Regression	86
4.2.1.7	Gaussian Naive Bayes	87

4.2.1.8	Bernoulli Naive Bayes	87
4.2.1.9	Neural Network	87
4.2.1.10	Stochastic Gradient Descent	88
4.2.2	Regression	88
4.2.2.1	Decision Tree	88
4.2.2.2	Boosted Decision Tree	95
4.2.2.3	K-Nearest Neighbour	103
4.2.2.4	Random Forest	110
4.2.2.5	Linear Regression	118
4.2.2.6	Neural Network	126
4.2.2.7	Stochastic Gradient Descent	134
4.3	Bayesian Statistics	142
4.3.1	Metropolis-Hastings	142
4.3.2	No-U-Turn Sampler (NUTS)	149
4.4	Strategy	157
4.4.1	Classification	157
4.4.2	Regression	159
5	Discussion and Suggestions for Future Research	163
5.1	Time Series Analysis	163
5.1.1	Random Walk	163
5.1.2	Ordinary Least Squares (OLS)	163
5.1.3	Auto Regressive (AR)	164
5.1.4	Moving Average (MA)	164
5.1.5	Auto Regressive Moving Average (ARMA)	164
5.1.6	Auto Regressive Integrated Moving Average (ARIMA)	164
5.2	Machine Learning	165
5.2.1	Classification	165
5.2.1.1	Decision Tree	165
5.2.1.2	Boosted Decision Tree	165
5.2.1.3	Support Vector Machine (SVM)	165
5.2.1.4	Random Forest	165
5.2.1.5	K-Nearest Neighbour	165
5.2.1.6	Logistic Regression	165
5.2.1.7	Bernoulli Naive Bayes	165
5.2.1.8	Gaussian Naive Bayes	166
5.2.1.9	Neural Network	166
5.2.1.10	Stochastic Gradient Descent	166
5.2.2	Regression	166
5.2.2.1	Decision Tree	166
5.2.2.2	Boosted Decision Tree	166
5.2.2.3	K-Nearest Neighbour	167
5.2.2.4	Random Forest	167
5.2.2.5	Linear Regression	167
5.2.2.6	Neural Network	167
5.2.2.7	Stochastic Gradient Descent	167
5.3	Bayesian Statistics	167

5.3.0.1	Metropolis-Hastings	167
5.3.0.2	No-U-Turn-Sampler (NUTS)	168
5.4	Strategy	168
5.4.1	Classification	168
5.4.2	Regression	168
6	Conclusion	169
A	An Appendix	170
	Bibliography	172

List of Figures

4.1	Basket of stocks	21
4.2	MSFT time series analysis	22
4.3	MSFT histogram of returns	23
4.4	CDE time series analysis	23
4.5	CDE histogram of returns	24
4.6	NAV B time series analysis	24
4.7	NAV B histogram of returns	25
4.8	HRG time series analysis	25
4.9	HRG histogram of returns	26
4.10	HL time series analysis	26
4.11	HL histogram of returns	27
4.12	MSFT OLS in-sample prediction	28
4.13	100 day MSFT OLS in-sample forecast	28
4.14	CDE OLS in-sample prediction	29
4.15	100 Day CDE OLS out of sample forecast	30
4.16	NAV B OLS in-sample prediction	31
4.17	100 day NAV B OLS in-sample forecast	31
4.18	HRG OLS in-sample prediction	32
4.19	100 day HRG OLS in-sample forecast	33
4.20	HL OLS in-sample prediction	34
4.21	100 day HL OLS in-sample forecast	34
4.22	MSFT AR time series analysis	35
4.23	MSFT AR histogram of returns	36
4.24	MSFT AR in-sample returns prediction	36
4.25	100 day MSFT AR in-sample returns forecast	37
4.26	CDE AR time series analysis	38
4.27	CDE AR histogram of returns	38
4.28	CDE AR in-sample returns prediction	39
4.29	100 day CDE AR in-sample returns forecast	39
4.30	NAV B AR time series analysis	40
4.31	NAV B AR histogram of returns	41
4.32	NAV B AR in-sample returns prediction	41
4.33	100 day NAV B AR in-sample returns forecast	42
4.34	HRG AR time series analysis	43
4.35	HRG AR histogram of returns	43
4.36	HL AR in-sample returns prediction	44
4.37	100 day HRG AR in-sample returns forecast	44

4.38	HL AR time series analysis	45
4.39	HL AR histogram of returns	46
4.40	HL AR in-sample returns prediction	46
4.41	100 day HL AR in-sample returns forecast	47
4.42	MSFT MA time series analysis	48
4.43	MSFT MA histogram of returns	48
4.44	MSFT MA in-sample returns prediction	49
4.45	100 day MSFT MA in-sample returns forecast	49
4.46	CDE MA time series analysis	50
4.47	CDE MA histogram of returns	51
4.48	CDE MA in-sample returns prediction	51
4.49	100 day CDE MA in-sample returns forecast	52
4.50	NAV B MA time series analysis	53
4.51	NAV B MA histogram of returns	53
4.52	NAV B MA in-sample returns prediction	54
4.53	100 day NAV B MA in-sample returns forecast	54
4.54	HRG MA time series analysis	55
4.55	HRG MA histogram of returns	56
4.56	HRG MA in-sample returns prediction	56
4.57	100 day HRG MA in-sample returns forecast	57
4.58	HL MA time series analysis	58
4.59	HL MA histogram of returns	58
4.60	HL MA in-sample returns prediction	59
4.61	100 day HL MA in-sample returns forecast	59
4.62	MSFT ARMA time series analysis	60
4.63	MSFT ARMA histogram of returns	61
4.64	MSFT ARMA in-sample returns prediction	61
4.65	100 day MSFT ARMA in-sample returns forecast	62
4.66	CDE ARMA time series analysis	63
4.67	CDE ARMA histogram of returns	63
4.68	CDE ARMA in-sample returns prediction	64
4.69	100 day CDE ARMA in-sample returns forecast	64
4.70	NAV B ARMA time series analysis	65
4.71	NAV B ARMA histogram of returns	66
4.72	NAV B ARMA in-sample returns prediction	66
4.73	100 day NAV B ARMA in-sample returns forecast	67
4.74	HRG ARMA time series analysis	68
4.75	HRG ARMA histogram of returns	68
4.76	HRG ARMA in-sample returns prediction	69
4.77	100 day HRG ARMA in-sample returns forecast	69
4.78	HL ARMA time series analysis	70
4.79	HL ARMA histogram of returns	71
4.80	HL ARMA in-sample returns prediction	71
4.81	100 day HL ARMA in-sample returns forecast	72
4.82	MSFT ARIMA time series analysis	73
4.83	MSFT ARIMA histogram of returns	73
4.84	MSFT ARIMA in-sample returns prediction	74

4.85 100 day MSFT ARIMA in-sample returns forecast	74
4.86 CDE ARIMA time series analysis	75
4.87 CDE ARIMA histogram of returns	76
4.88 CDE ARIMA in-sample returns prediction	76
4.89 100 day CDE ARIMA in-sample returns forecast	77
4.90 NAVB ARIMA time series analysis	78
4.91 NAVB ARIMA histogram of returns	78
4.92 NAVB ARIMA in-sample returns prediction	79
4.93 100 day NAVB ARIMA in-sample returns forecast	79
4.94 HRG ARIMA time series analysis	80
4.95 HRG ARIMA histogram of returns	81
4.96 HRG ARIMA in-sample returns prediction	81
4.97 100 day HRG ARIMA in-sample returns forecast	82
4.98 HL ARIMA time series analysis	83
4.99 HL ARIMA histogram of returns	83
4.100HL ARIMA in-sample returns prediction	84
4.101100 day HL ARIMA in-sample returns forecast	84
4.102MSFT Decision Trees in-sample prediction	88
4.103100 day MSFT Decision Trees out-of-sample forecast	89
4.104CDE Decision Trees in-sample prediction	90
4.105100 day CDE Decision Trees out-of-sample forecast	90
4.106NAVB Decision Trees in-sample prediction	91
4.107100 day NAVB Decision Trees out-of-sample forecast	92
4.108HRG Decision Trees in-sample prediction	93
4.109100 day HRG Decision Trees out-of-sample forecast	93
4.110HL Decision Trees in-sample prediction	94
4.111100 day HL Decision Trees out-of-sample forecast	95
4.112MSFT Boosted Decision Trees in-sample prediction	96
4.113100 day MSFT Boosted Decision Trees out-of-sample forecast	96
4.114CDE Boosted Decision Trees in-sample prediction	97
4.115100 day CDE Boosted Decision Trees out-of-sample forecast	98
4.116NAVB Boosted Decision Trees in-sample prediction	99
4.117100 day NAVB Boosted Decision Trees out-of-sample forecast	99
4.118HRG Boosted Decision Trees in-sample prediction	100
4.119100 day HRG Boosted Decision Trees out-of-sample forecast	101
4.120HL Boosted Decision Trees in-sample prediction	102
4.121100 day HL Boosted Decision Trees out-of-sample forecast	102
4.122MSFT K-Nearest Neighbour in-sample prediction	103
4.123100 day MSFT K-Nearest Neighbour out-of-sample forecast	104
4.124CDE K-Nearest Neighbour in-sample prediction	105
4.125100 day CDE K-Nearest Neighbour out-of-sample forecast	105
4.126NAVB K-Nearest Neighbour in-sample prediction	106
4.127100 day NAVB K-Nearest Neighbour out-of-sample forecast	107
4.128HRG K-Nearest Neighbour in-sample prediction	108
4.129100 day HRG K-Nearest Neighbour out-of-sample forecast	108
4.130HL K-Nearest Neighbour in-sample prediction	109
4.131100 day HL K-Nearest Neighbour out-of-sample forecast	110

4.132MSFT Random Forest in-sample prediction	111
4.133100 day MSFT Random Forest out-of-sample forecast	111
4.134CDE Random Forest in-sample prediction	112
4.135100 day CDE Random Forest out-of-sample forecast	113
4.136NAVB Random Forest in-sample prediction	114
4.137100 day NAVB Random Forest out-of-sample forecast	114
4.138HRG Random Forest in-sample prediction	115
4.139100 day HRG Random Forest out-of-sample forecast	116
4.140HL Random Forest in-sample prediction	117
4.141100 day HL Random Forest out-of-sample forecast	117
4.142MSFT Linear Regression in-sample prediction	118
4.143100 day MSFT Linear Regression out-of-sample forecast	119
4.144CDE Linear Regression in-sample prediction	120
4.145100 day CDE Linear Regression out-of-sample forecast	120
4.146CDE Linear Regression in-sample prediction	122
4.147100 day CDE Linear Regression out-of-sample forecast	122
4.148HRG Linear Regression in-sample prediction	123
4.149100 day HRG Linear Regression out-of-sample forecast	124
4.150HL Linear Regression in-sample prediction	125
4.151100 day HL Linear Regression out-of-sample forecast	125
4.152MSFT Neural Network in-sample prediction	126
4.153100 day MSFT Neural Network out-of-sample forecast	127
4.154CDE Neural Network in-sample prediction	129
4.155100 day CDE Neural Network out-of-sample forecast	129
4.156NAVB Neural Network in-sample prediction	130
4.157100 day NAVB Neural Network out-of-sample forecast	131
4.158HRG Neural Network in-sample prediction	132
4.159100 day HRG Neural Network out-of-sample forecast	132
4.160HL Neural Network in-sample prediction	133
4.161100 day HL Neural Network out-of-sample forecast	134
4.162MSFT SGD in-sample prediction	135
4.163100 day MSFT SGD out-of-sample forecast	135
4.164CDE SGD in-sample prediction	136
4.165100 day CDE SGD out-of-sample forecast	137
4.166NAVB SGD in-sample prediction	138
4.167100 day NAVB SGD out-of-sample forecast	138
4.168HRG SGD in-sample prediction	139
4.169100 day HRG SGD out-of-sample forecast	140
4.170HL SGD in-sample prediction	141
4.171100 day HL SGD out-of-sample forecast	141
4.172MSFT Metropolis-Hastings in-sample prediction	142
4.173100 day MSFT Metropolis-Hastings out-of-sample forecast	143
4.174CDE Metropolis-Hastings in-sample prediction	144
4.175100 day CDE Metropolis-Hastings out-of-sample forecast	144
4.176NAVB Metropolis-Hastings in-sample prediction	145
4.177100 day NAVB Metropolis-Hastings out-of-sample forecast	146
4.178HRG Metropolis-Hastings in-sample prediction	147

4.179	100 day HRG Metropolis-Hastings out-of-sample forecast	147
4.180	HL Metropolis-Hastings in-sample prediction	148
4.181	100 day HL Metropolis-Hastings out-of-sample forecast	149
4.182	MSFT NUTS in-sample prediction	150
4.183	100 day MSFT NUTS out-of-sample forecast	150
4.184	CDE NUTS in-sample prediction	151
4.185	100 day CDE NUTS out-of-sample forecast	152
4.186	NAVB NUTS in-sample prediction	153
4.187	100 day NAVB NUTS out-of-sample forecast	153
4.188	HRG NUTS in-sample prediction	154
4.189	100 day HRG NUTS out-of-sample forecast	155
4.190	HL NUTS in-sample prediction	156
4.191	100 day HL NUTS out-of-sample forecast	156
4.192	Machine Learning Classifier strategy with only upwards forecasts	157
4.193	Machine Learning Classifier strategy with upwards and downwards forecasts	158
4.194	Machine Learning Classifier strategy with stop loss	159
4.195	Machine Learning Regression strategy with only upwards forecasts	160
4.196	Machine Learning Regression strategy with upwards and downwards forecasts	161
4.197	Machine Learning Regression strategy with stop loss	162

List of Tables

4.1	Equities Descriptive Statistics	22
4.2	Decision Tree results	85
4.3	Boosted Decision Tree results	85
4.4	Support Vector Machine results	85
4.5	Random Forest results	86
4.6	K-Nearest Neighbour results	86
4.7	Logistic Regression results	86
4.8	Gaussian Naive Bayes results	87
4.9	Bernoulli Naive Bayes results	87
4.10	Neural Network results	87
4.11	Stochastic Gradient Descent results	88
4.12	Machine Learning Classifier strategy with only upwards forecasts	157
4.13	Machine Learning Classifier strategy with upwards and downwards forecasts	158
4.14	Machine Learning Classifier strategy with stop loss	158
4.15	Machine Learning Regression strategy with only upwards forecasts	159
4.16	Machine Learning Regression strategy with upwards and downwards forecasts	160
4.17	Machine Learning Regression strategy with stop loss	161

Abbreviations

EMH	Efficient Market Hypothesis
RWH	Random Walk Hypothesis
OLS	Ordinary Least Squares
AR	Auto Regressive
MA	Moving Average
ARMA	Auto Regressive Moving Average
ARIMA	Auto Regressive Integrated Moving Average
GARCH	Generalised Auto Regressive Conditional Heteroskedasticity
EGARCH	Exponential Generalised Auto Regressive Conditional Heteroskedasticity
ADF	Augmented Dickey Fuller
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent
SMA	Simple Moving Average
AIC	Alkaline Information Criterion
IC	Information Criterion
NUTS	No-U-Turn Sampler
QQ	Quantile-Quantile
ETF	Exchange Traded Fund
ANN	Artificial Neural Network

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

The stock market retains its status as a prime location for investors to invest in the market and earn a profit, however this is not always easy due to the constantly thriving and changing nature which follows the stock market. Investors are constantly presented with numerous profit potential opportunities, however without intensive planning and analysis, these opportunities could easily turn into losses. This means that it is crucial for every investor to carry out stock market analysis prior to any investment by monitoring past price movements in order to forecast future trends. Even though past data is not a clear indication of future movement, it is still proven to provide some useful insight.

1.1 Project goals and scope

This report aims to disprove two hypotheses, the Efficient Market Hypotheses (EMH), and the Random Walk Hypothesis (RWH).

1.1.1 Efficient Market Hypothesis

By definition of the EMH, it is impossible for any investor to beat the market as the markets are efficient and all stock prices always incorporate and reflect all relevant information. This essentially means that stocks are always traded at their fair value, and their prices only fluctuate when new information is released to the public. This essentially makes it impossible for any investors to exploit any inefficiencies in the market and earn a profit. Believers of the EMH claim that it is pointless to try and search for undervalued stocks, or predict market trends.

Although a vast number of academics find the EMH to be true, there are still a number of investors who have consistently beaten the market over long periods of time, something which is impossible by definition according to the EMH. Stock market crashes, such as that of the financial crisis of 1987, in which the market fell 28.3% in a single day, proving that stock prices can seriously deviate from their fair values.

It is the firm belief of proponents of the EMH that investors are better off investing in low risk index funds. It is a known fact that only a few number of active managers successfully manage to outperform passive funds.

1.1.2 Random Walk Hypothesis

According to the RWH, it is not possible to use the past movement or trend of a stock price to predict its future movement. The main idea of the RWH is all stocks take a random and unpredictable path. Strong believers of the RWH believe that it is not possible to beat the market without taking on more risk.

There are however a vast number of critics of this theory, who state that stocks do indeed follow trends over time, making it possible to carry out stock market analysis and carefully select entry and exit points for stock investments.

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 x_t , each one being recorded at a specific time t . A discrete-time time series is one in which the set T_0 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 $T_0 = [0, 1]$.[\[2\]](#)

Noteworthy "time series momentum" has likewise been achieved in equity index, currency, commodity, and bond futures for each of the 58 liquid instruments which were considered.[3] Moskowitz et al. discover persistence in returns ranging from 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 was found to deliver substantial abnormal returns with little exposure to standard asset pricing factors and performs best during extreme markets. After examining the trading activities of speculators and hedgers, speculators were found to profit mostly from time series momentum at the expense of hedgers.

2.1.1 Auto Regressive Moving Average (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 Auto Regressive Integrated Moving Average (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

$$x_t - \alpha_1 X_{t-1} - \dots - \alpha_p X_{t-p} = \varepsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_q \epsilon_{t-q}$$

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.

Darrat et al. set out to investigate with the use of 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] Two different approaches were applied, the standard variance-ratio test, and a model-comparison test that compares the ex post forecasts from a naive model with those obtained from several alternative models such as ARIMA, GARCH, and ANNs. To evaluate ex post forecasts, Darrat et al. made use of several procedures including root-mean-square error (RMSE), mean absolute error (MAE), uncertainty coefficient, and encompassing tests. In contrast to the variance-ratio test, results from the model-comparison approach were quite decisive in rejecting the random-walk hypothesis in both Chinese stock markets. Moreover, the results showed 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 (MCMC)

In statistics, (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.

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.

Hoffman et al. introduce the 'No-U-Turn-Sampler', an extension to the Hamiltonian Monte Carlo (HMC), which is an MCMC algorithm that avoids the random walk behaviour and sensitivity to correlated parameters that plague many MCMC methods by

taking a series of steps informed by first-order gradient information.[\[18\]](#) In their paper, they claim HMCs performance to be highly sensitive to two user-specified parameters: a step size, and a desired number of steps. NUTS is an improvement from HMC as it eliminates the need to set a number of steps. Their algorithm uses a recursive algorithm to build a set of likely candidate points that spans a wide swath of the target distribution, stopping automatically when it starts to double back and retrace its steps.

Chapter 3

Experimental 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 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 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 parameter 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 value 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 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.

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. The sharpe ratio of the original and predicted price returns was calculated to serve as an accuracy score in the in-sample tests.

3.5.1 No-U-Turn Sampler (NUTS)

The model returns were modeled with a Student-t distribution with an unknown degrees of freedom paramater, and a scale paramater determined by a latent process.

3.5.2 Metropolis-Hastings

This model was fit to the data with the same paramaters as the NUTS algorithm except that the model was optimised using the L-BFGS-B algorithm.

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. The allocation amount of stock to invest in was determined based on the size of the portfolio.

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 close 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 was placed on a stock if the algorithm predicted an increase in the following day's price. The stock was shorted if the algorithm predicted a decrease in the following day's price. A stop loss of 80% was placed on each position.

3.6.2 Regression

An ordered dictionary was used to store the day's close price for each stock as the back-tester 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. An order was placed on a stock if the predicted price was higher than the current day's price. The stock was shorted if the predicted price was lower than the current day's price. A stop loss of 80% was placed on each position.

Chapter 4

Research Findings

For the purpose of benchmarking 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.

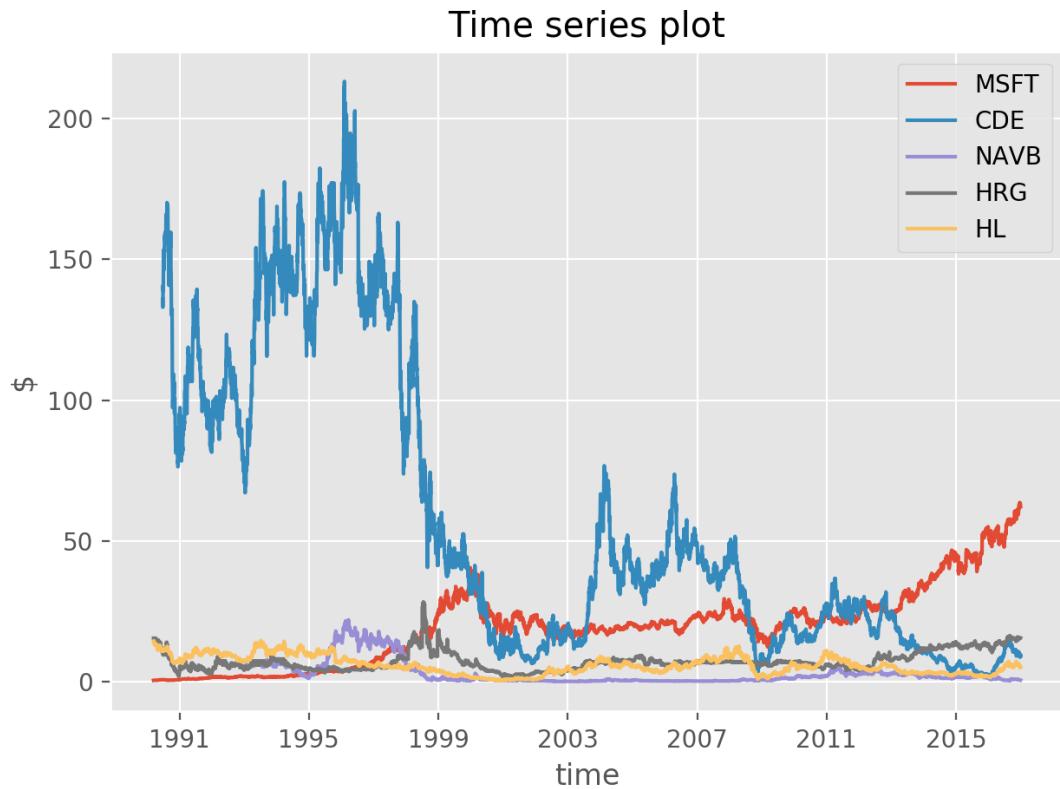


FIGURE 4.1: Basket of stocks

	MSFT	CDE	NAV	HRG	HL
Mean	16.983	57.716	3.000	13.328	7.154
Median	18.656	36.700	1.250	7.190	6.196
Maximum	63.620	213.178	22.000	89.340	23.830
Minimum	0.061	1.730	0.080	1.725	0.486
Variance	204.707	2807.852	18.445	249.105	18.445
Standard Deviation	14.308	52.989	4.29	15.783	4.295
Skewness	0.716	0.918	2.213	2.708	0.651
Kurtosis	0.250	-0.550	4.193	6.865	-0.081

TABLE 4.1: Equities Descriptive Statistics

4.1 Time Series Analysis

4.1.1 Random Walk

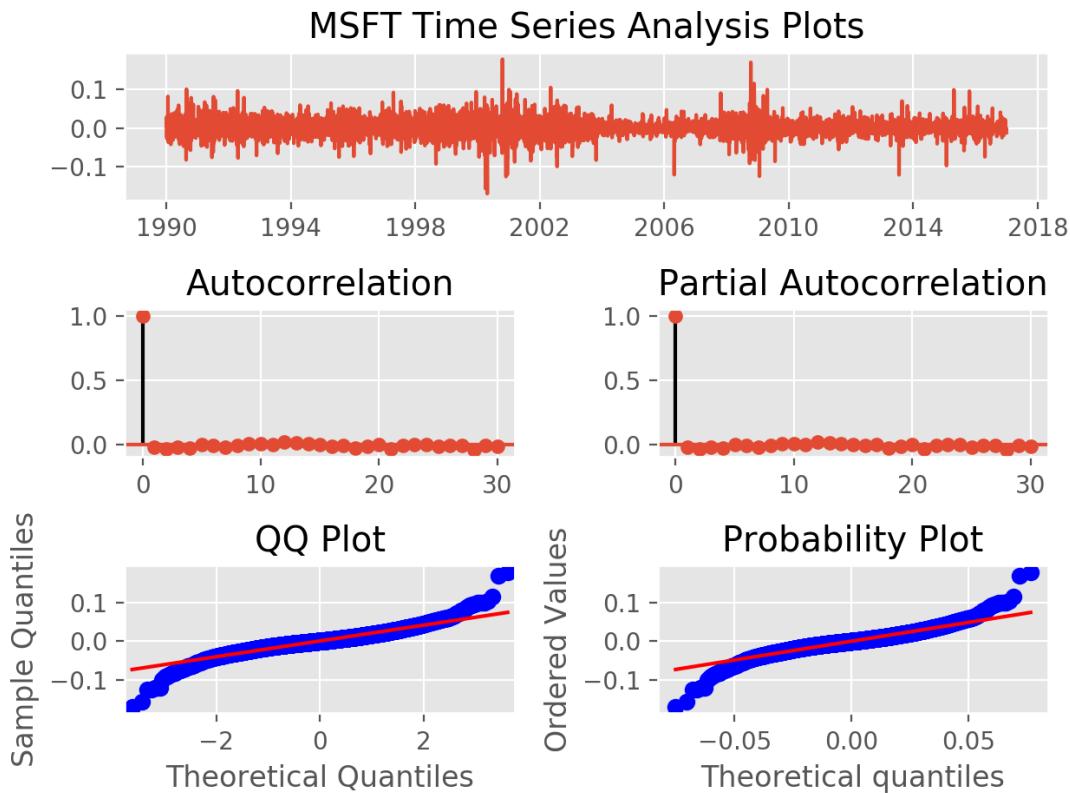


FIGURE 4.2: MSFT time series analysis

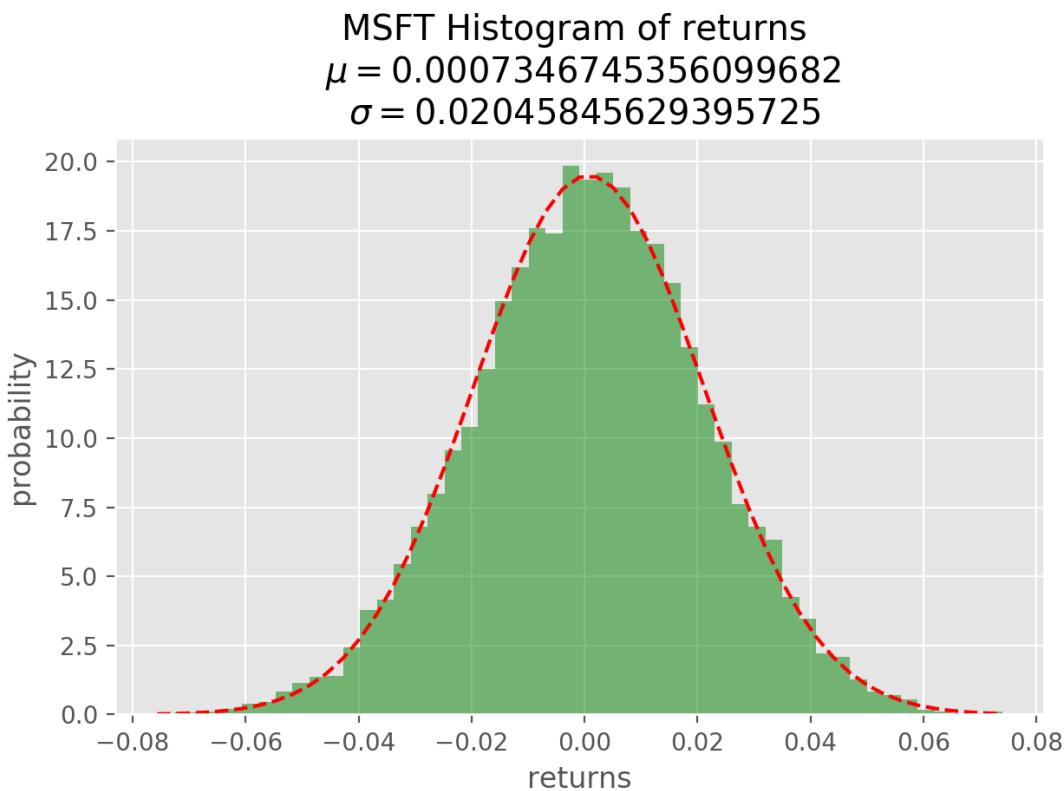


FIGURE 4.3: MSFT histogram of returns

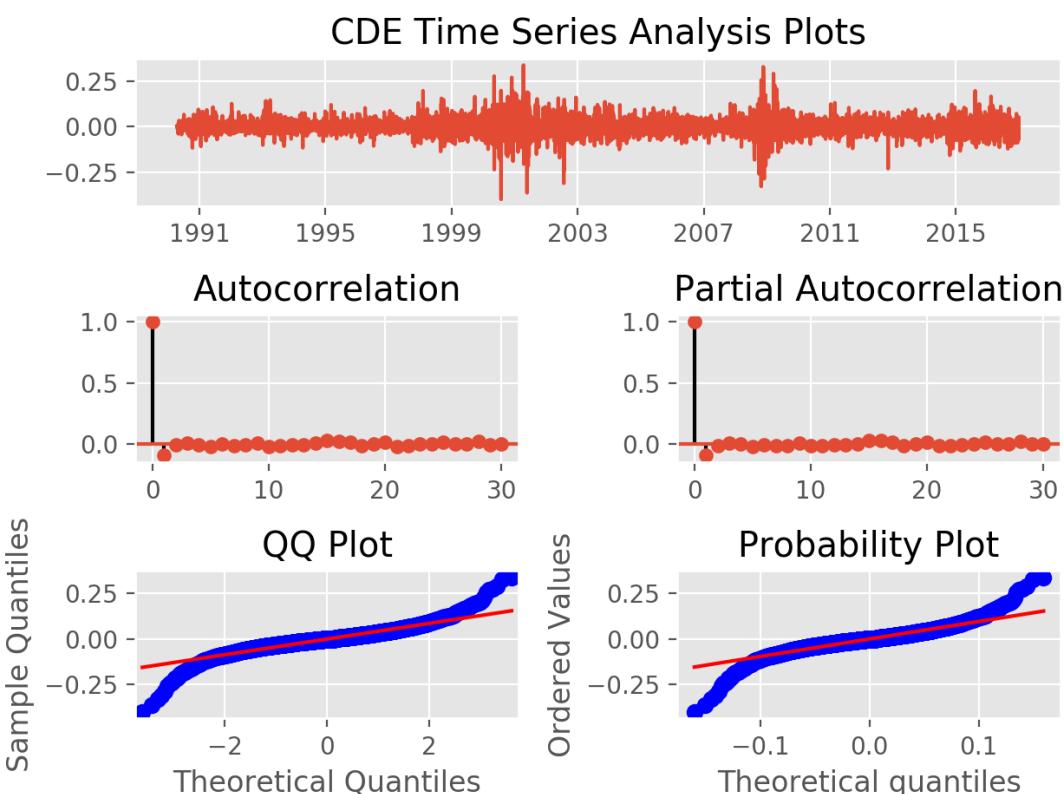


FIGURE 4.4: CDE time series analysis

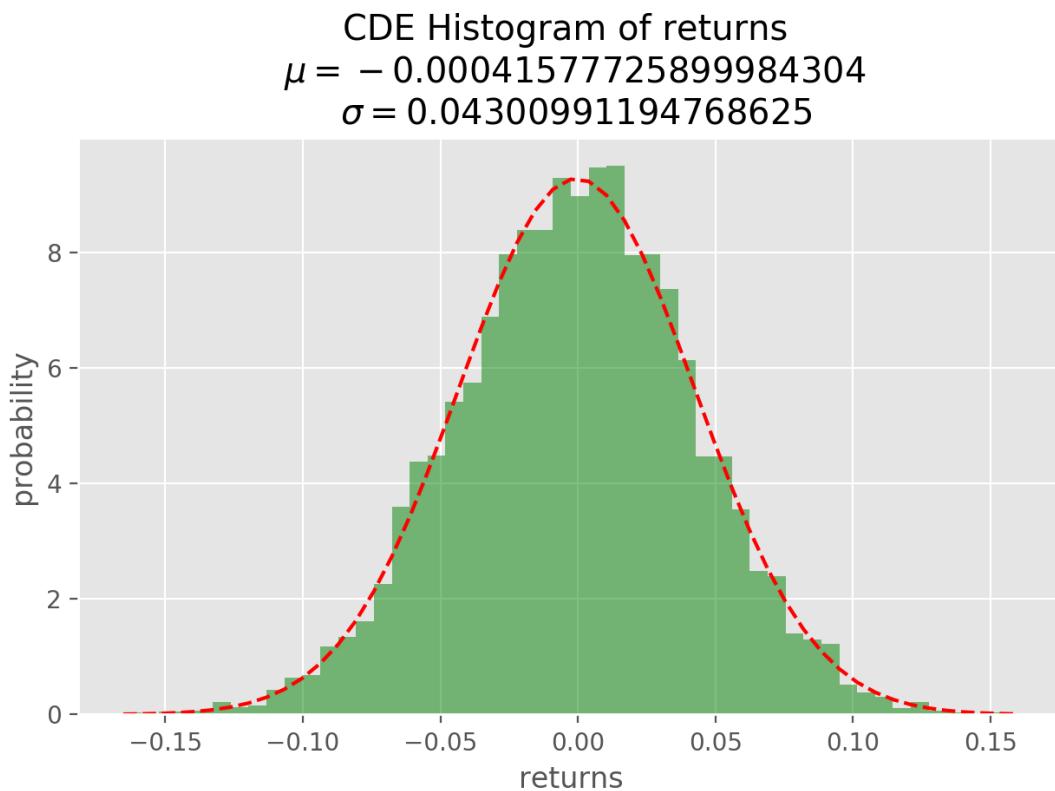


FIGURE 4.5: CDE histogram of returns

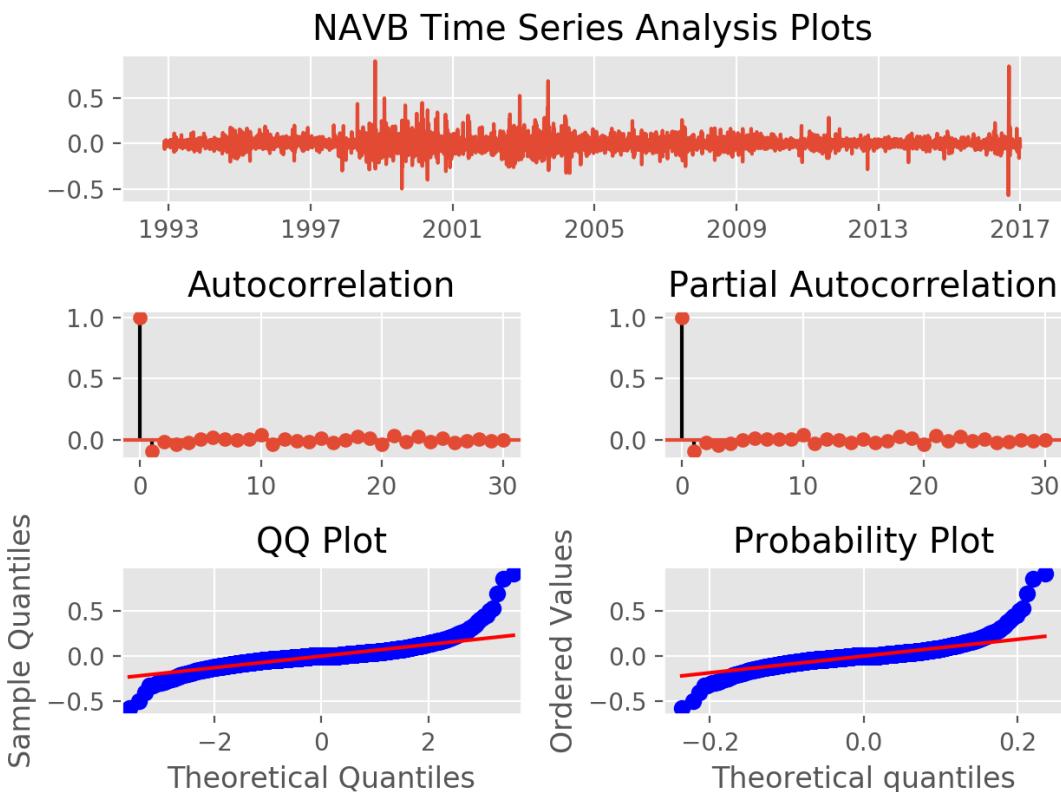


FIGURE 4.6: NAVB time series analysis

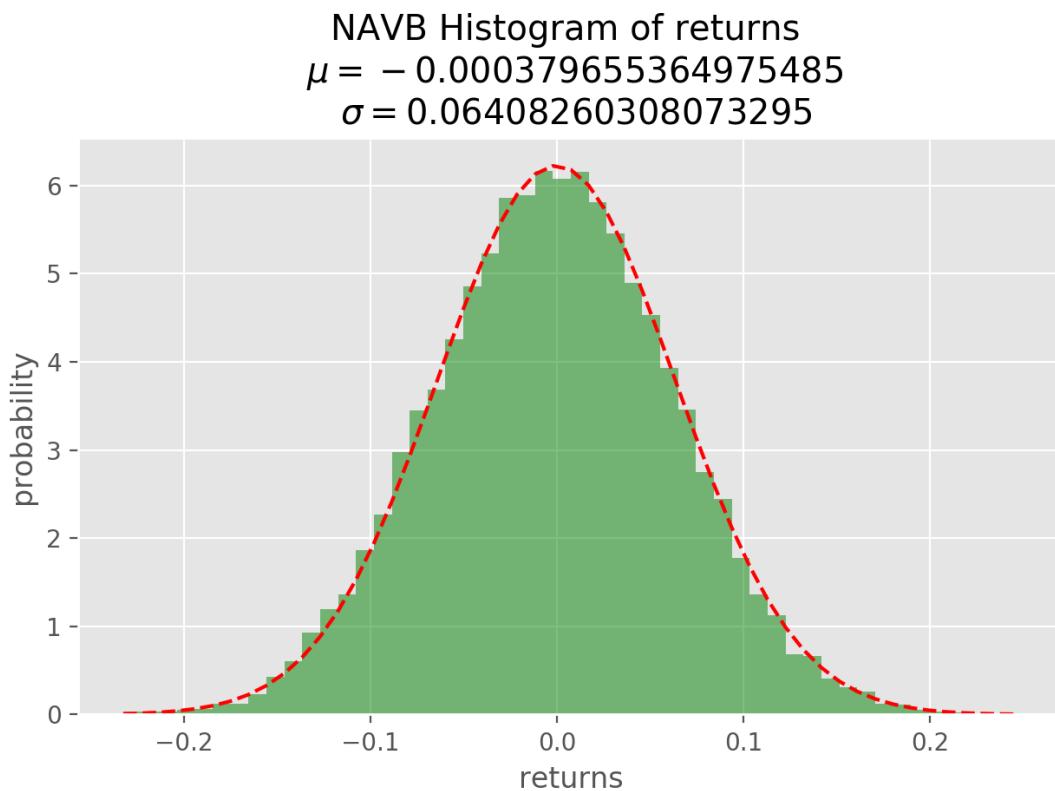


FIGURE 4.7: NAVB histogram of returns

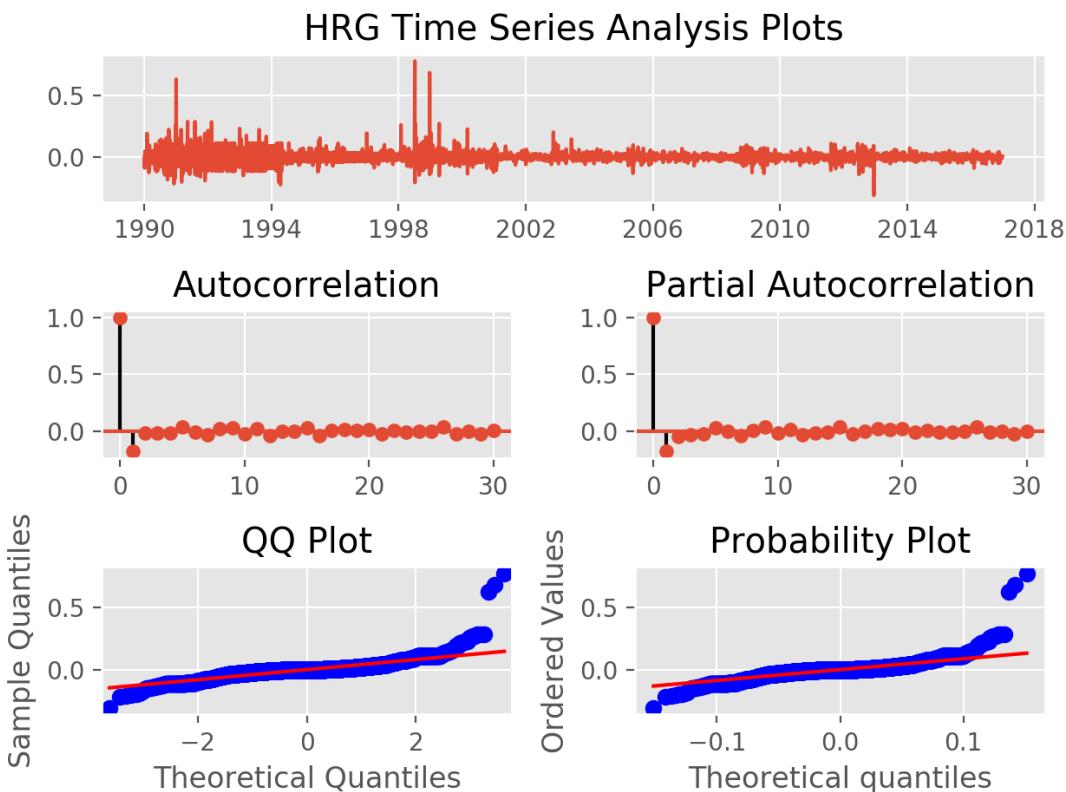


FIGURE 4.8: HRG time series analysis

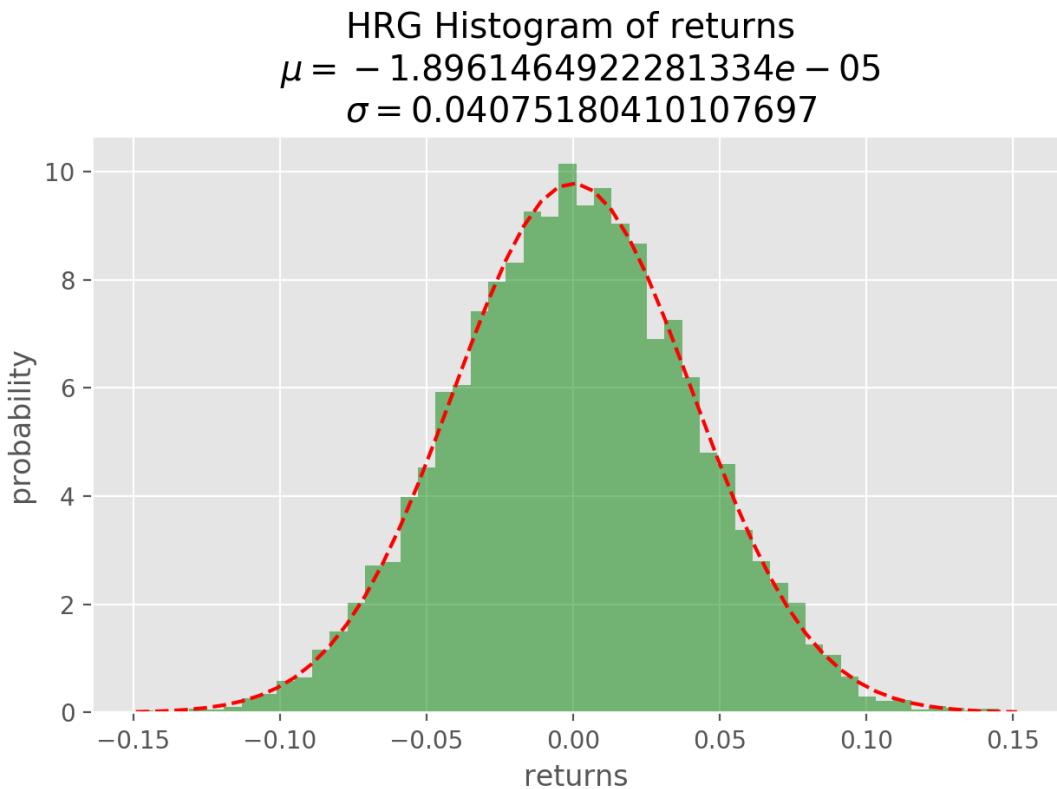


FIGURE 4.9: HRG histogram of returns

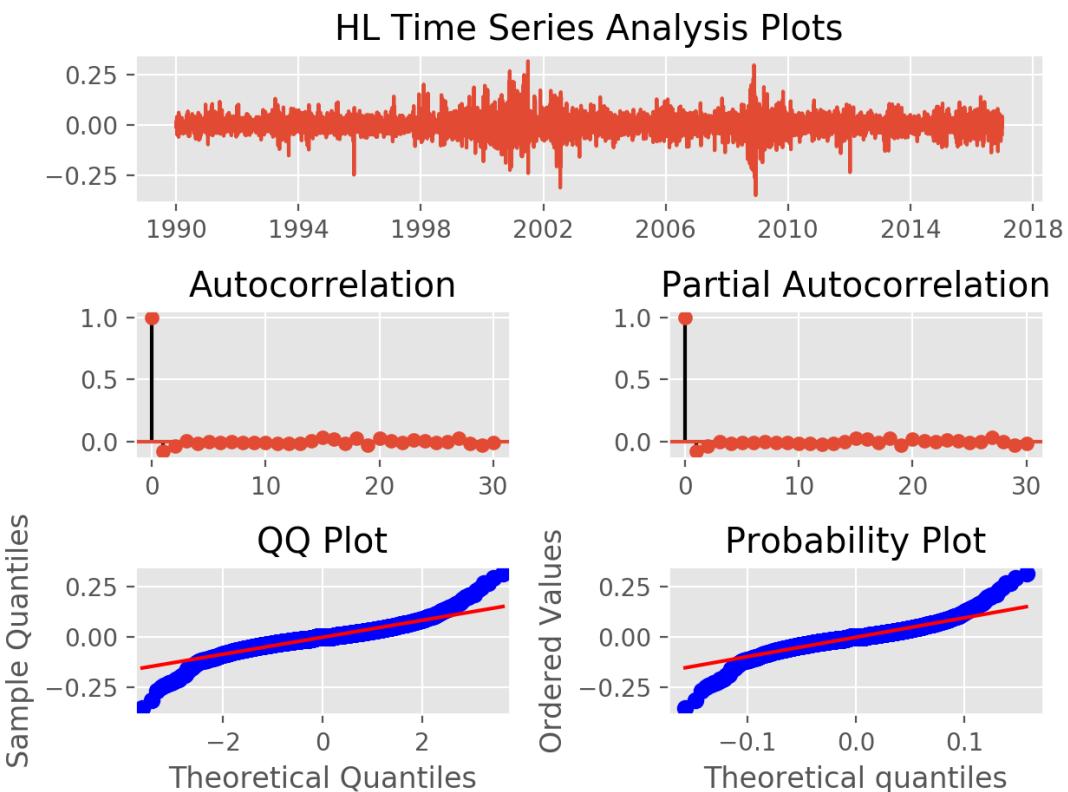


FIGURE 4.10: HL time series analysis

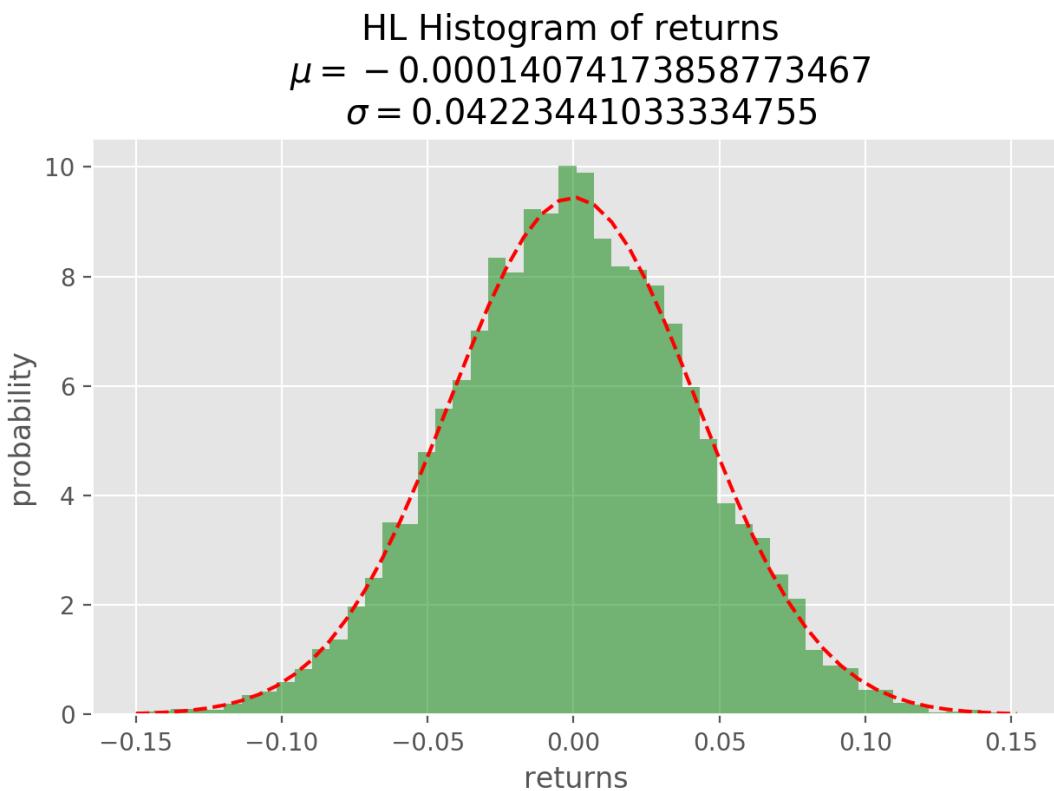


FIGURE 4.11: HL histogram of returns

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.

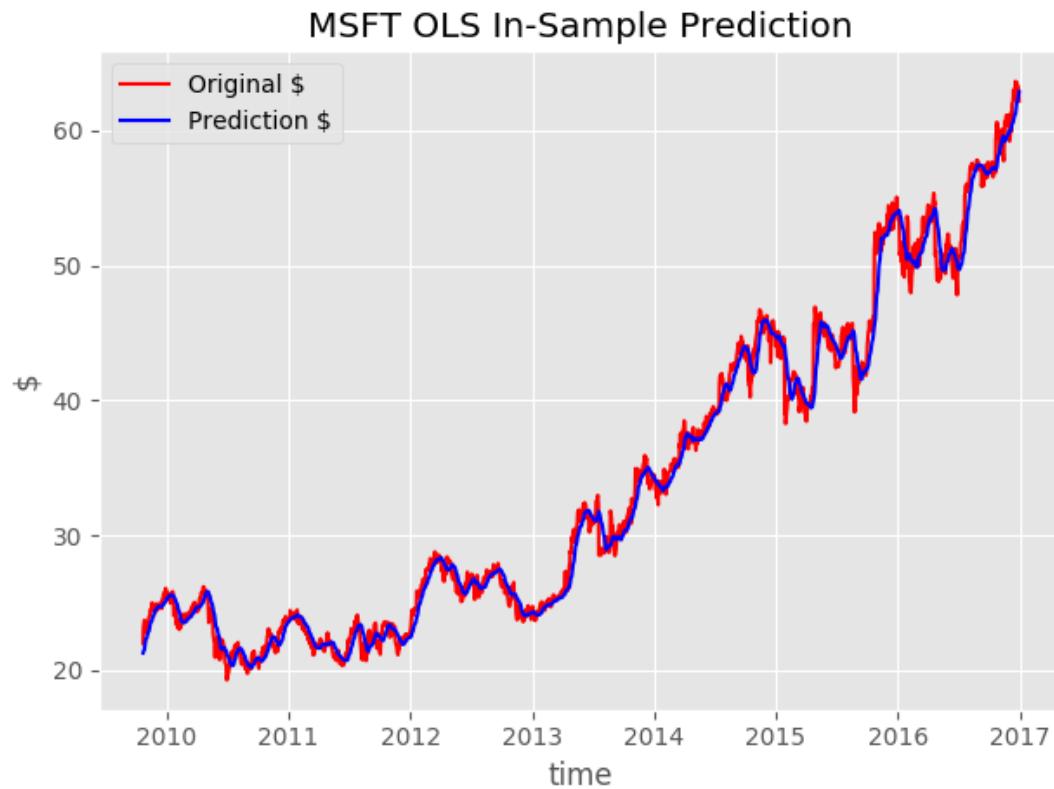


FIGURE 4.12: MSFT OLS in-sample prediction

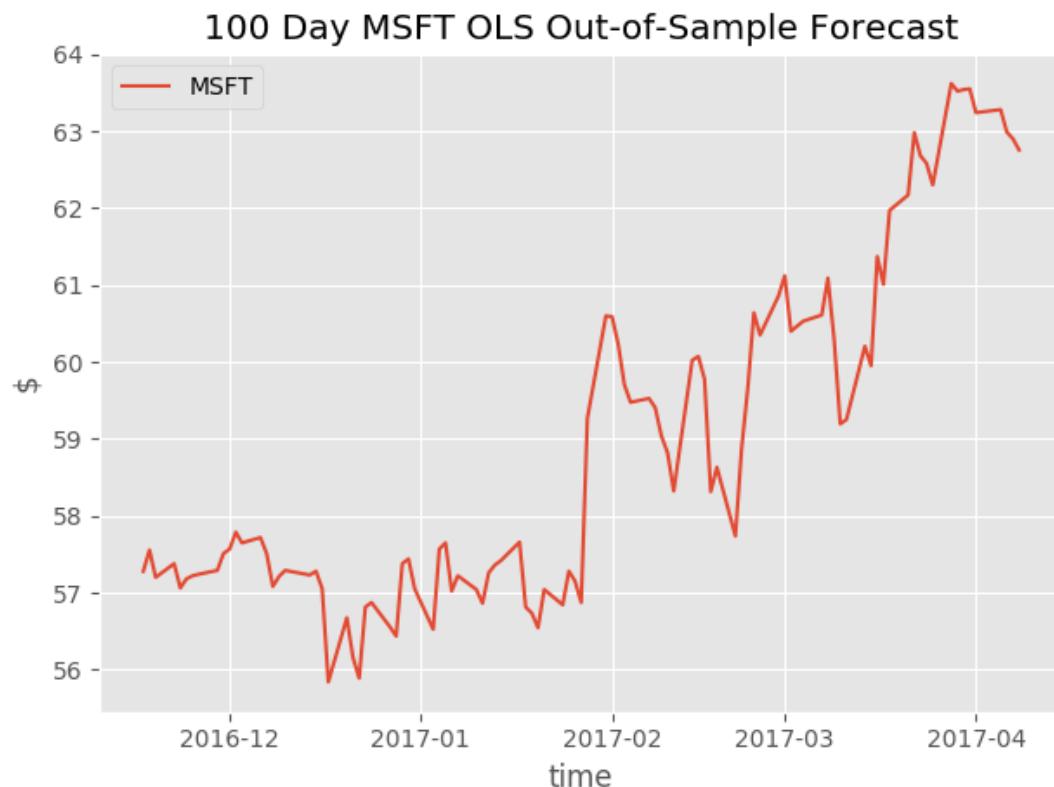


FIGURE 4.13: 100 day MSFT OLS in-sample forecast

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

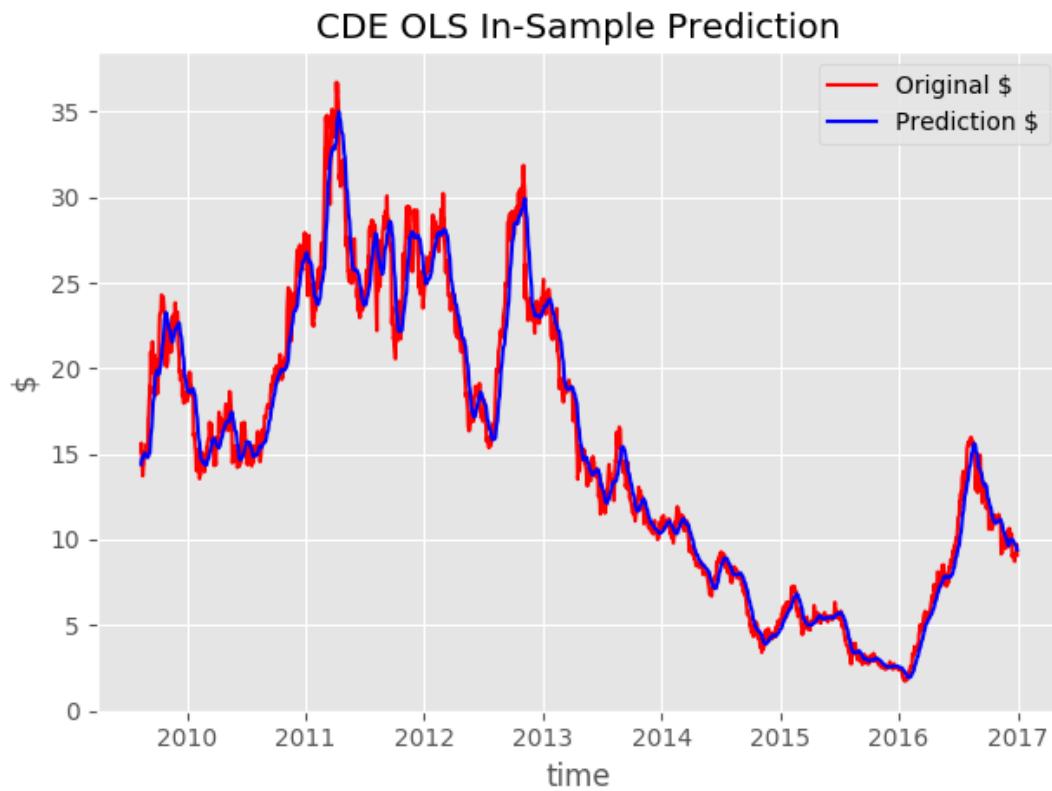


FIGURE 4.14: CDE OLS in-sample prediction

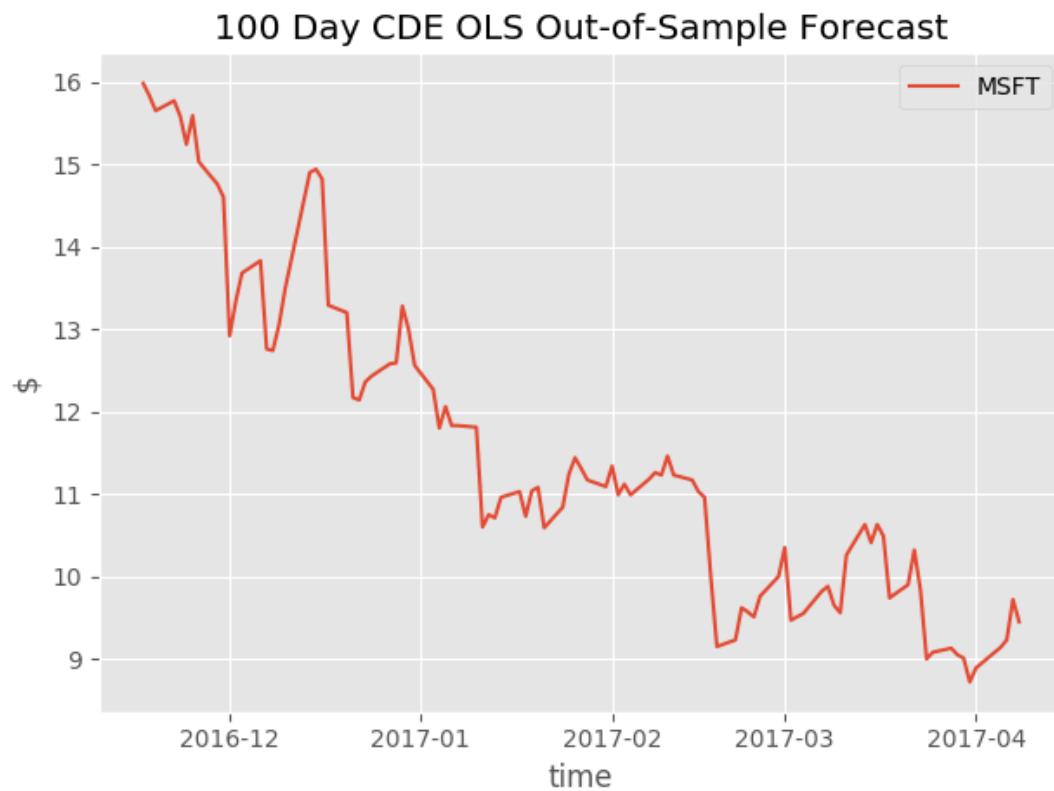


FIGURE 4.15: 100 Day CDE OLS out of sample forecast

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

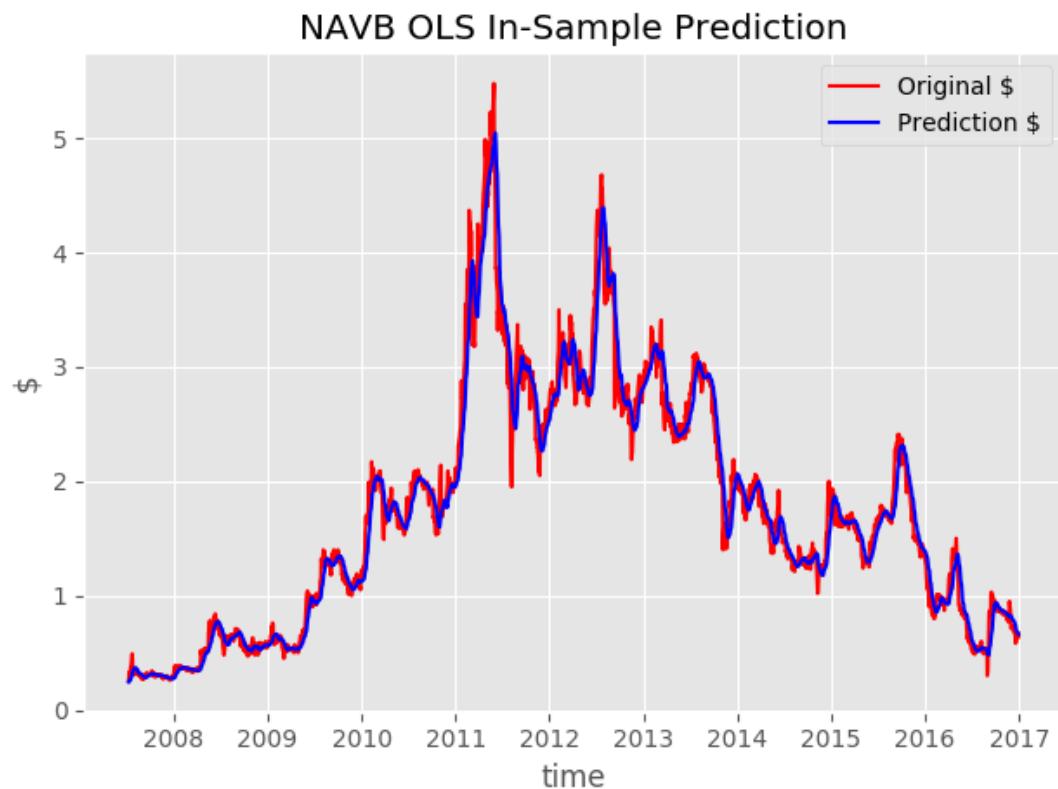


FIGURE 4.16: NAVB OLS in-sample prediction

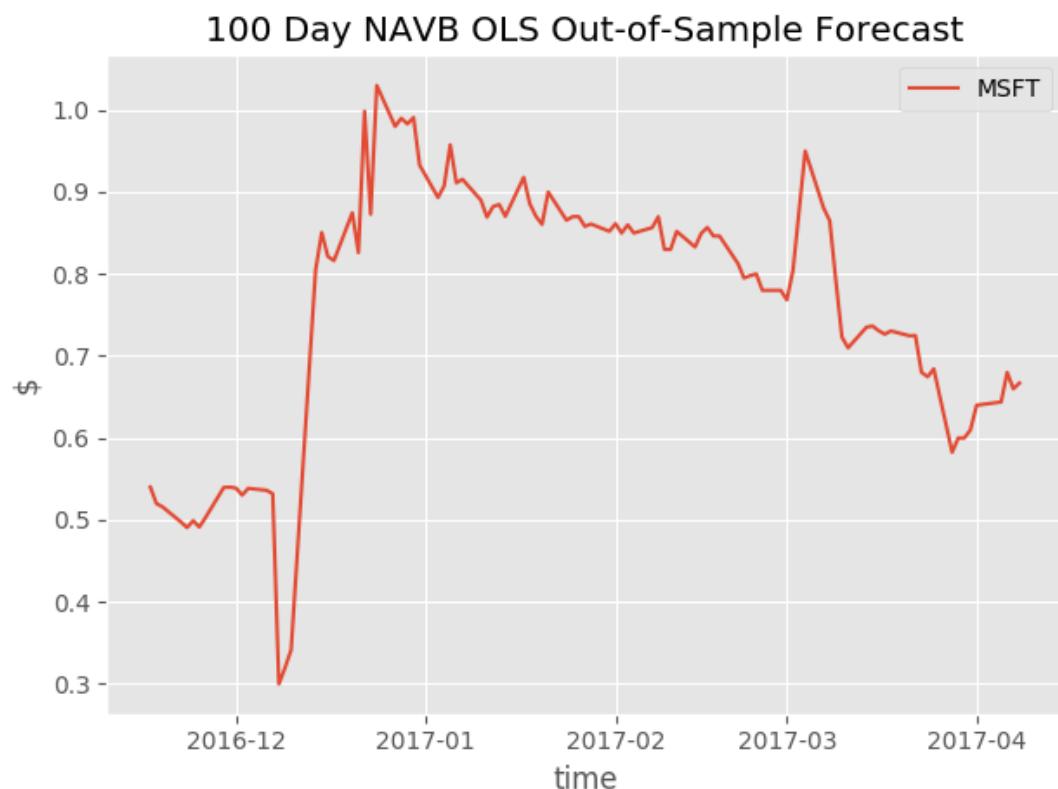


FIGURE 4.17: 100 day NAVB OLS in-sample forecast

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

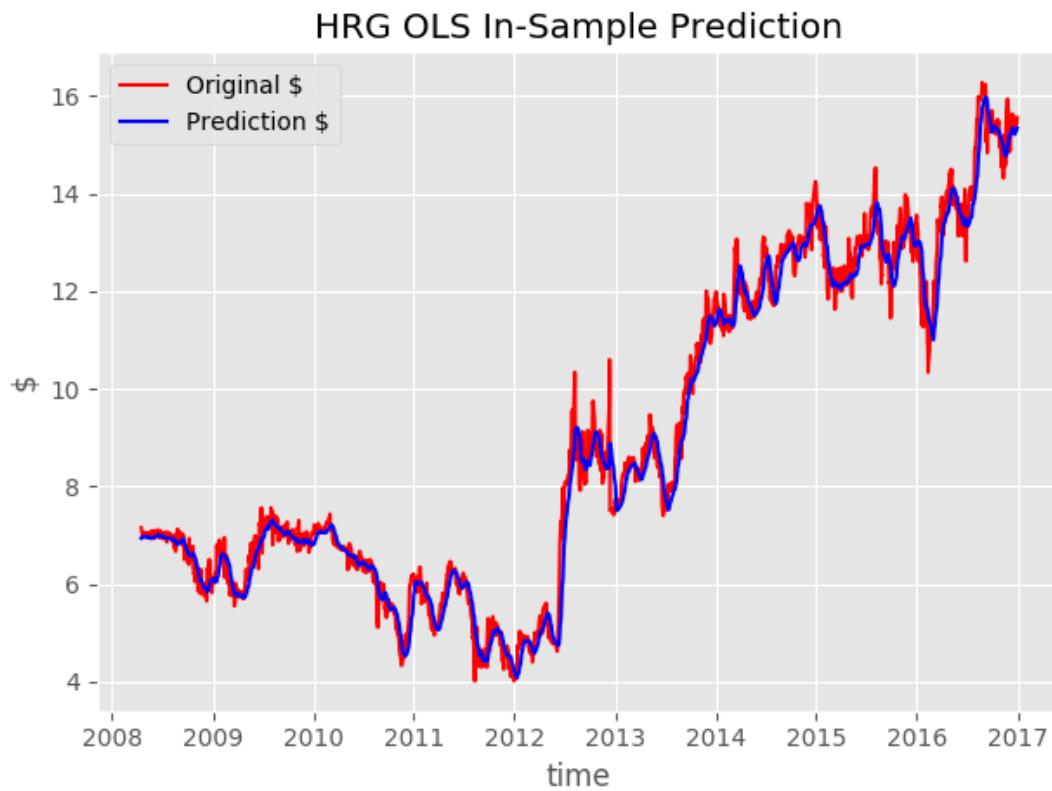


FIGURE 4.18: HRG OLS in-sample prediction

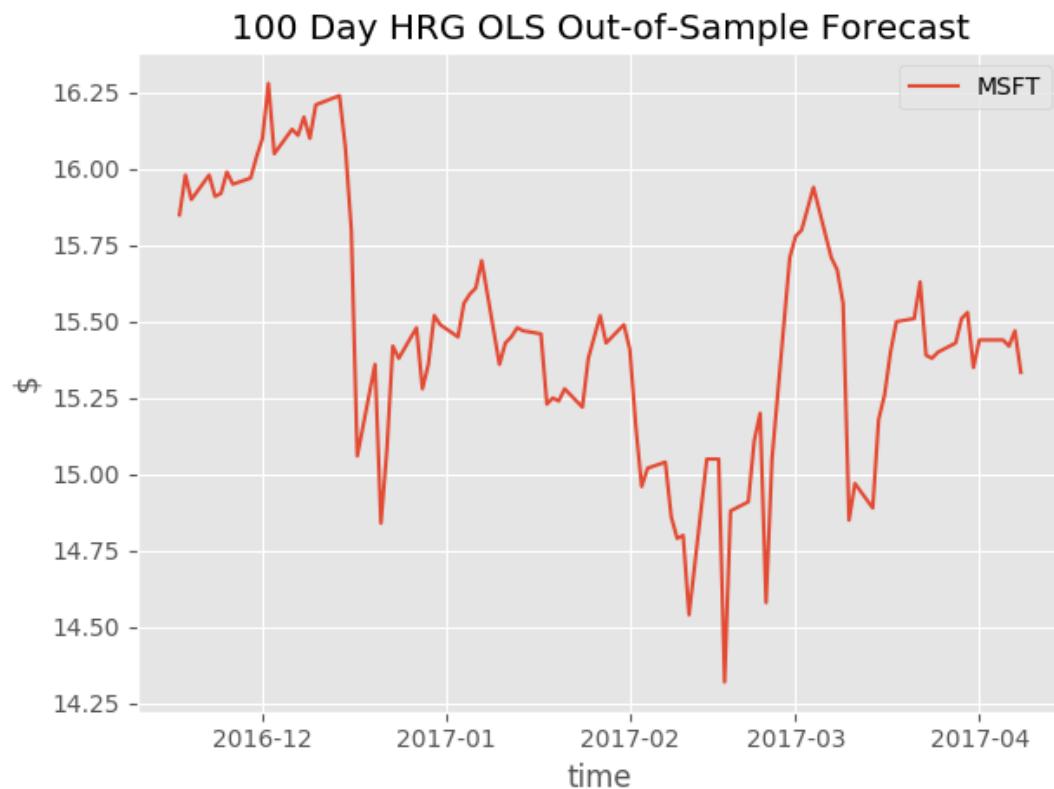


FIGURE 4.19: 100 day HRG OLS in-sample forecast

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

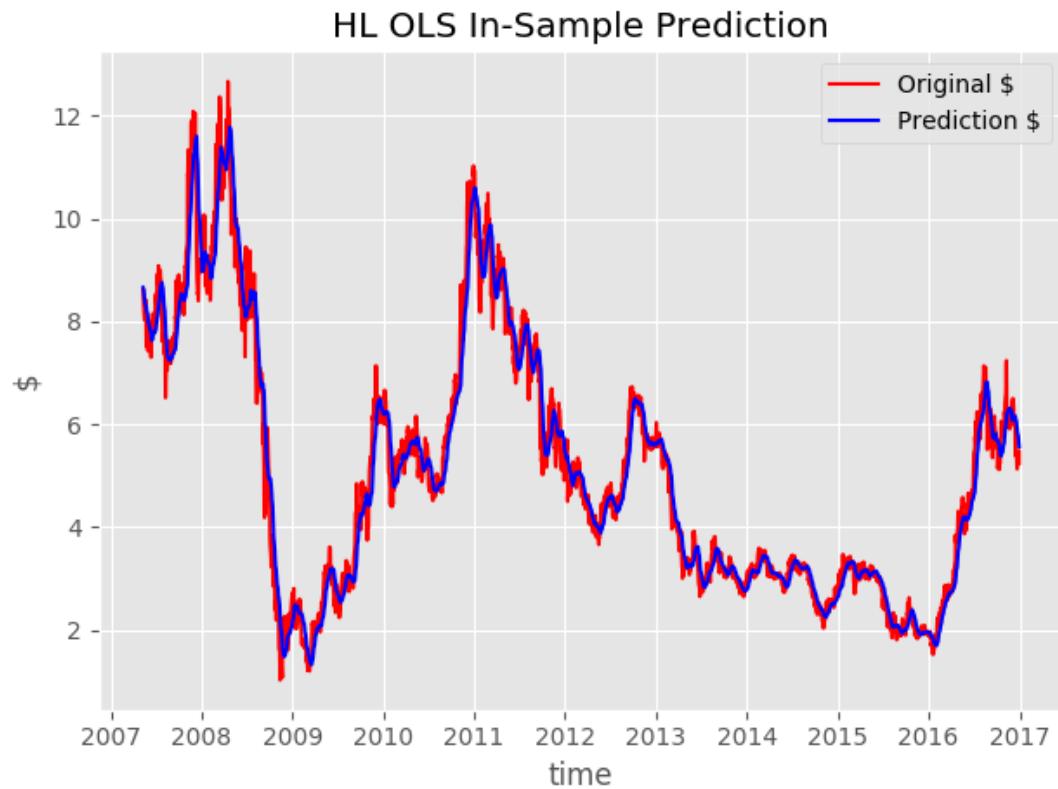


FIGURE 4.20: HL OLS in-sample prediction

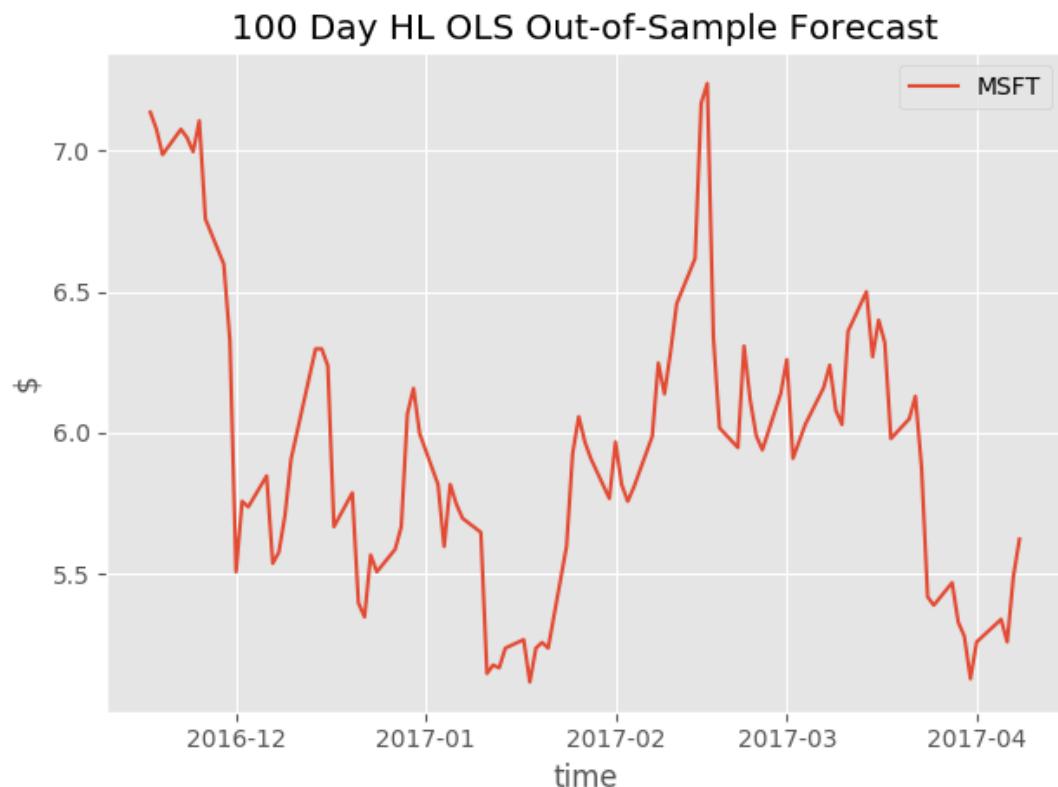


FIGURE 4.21: 100 day HL OLS in-sample forecast

4.1.3 Auto Regressive (AR)

MSFT scored sharpe ratios of 1.152 for the original returns, and -34.641 for the predicted returns in the in-sample test.

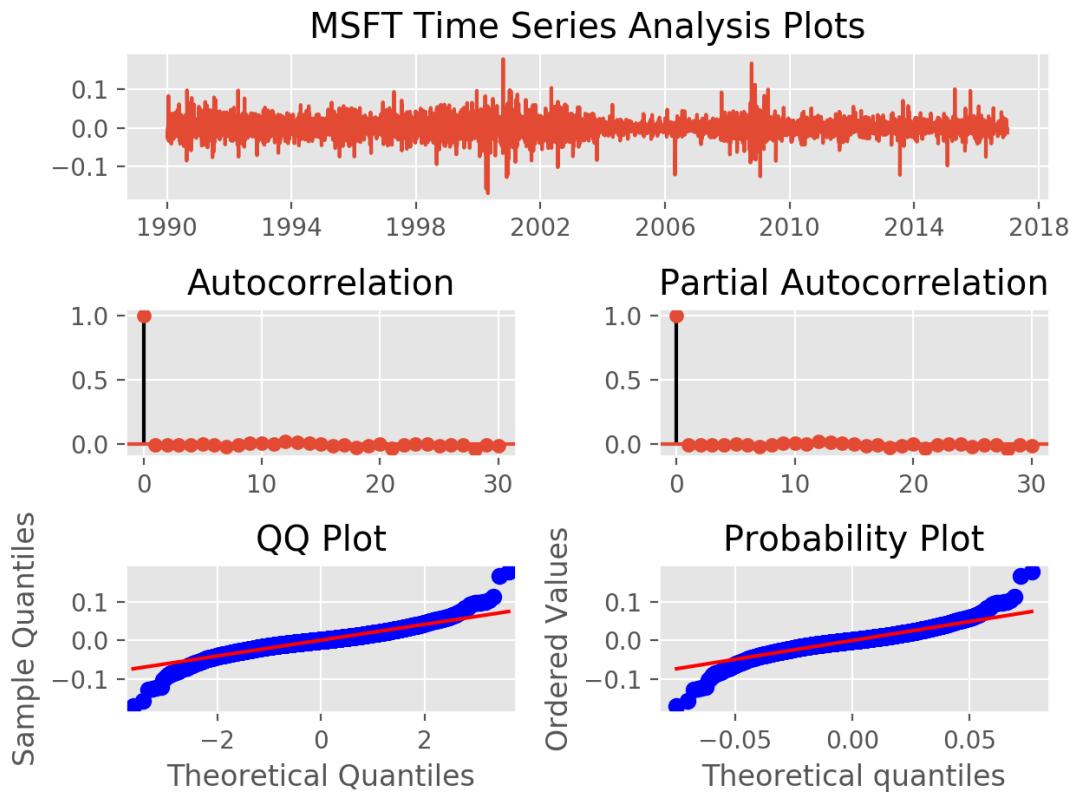


FIGURE 4.22: MSFT AR time series analysis

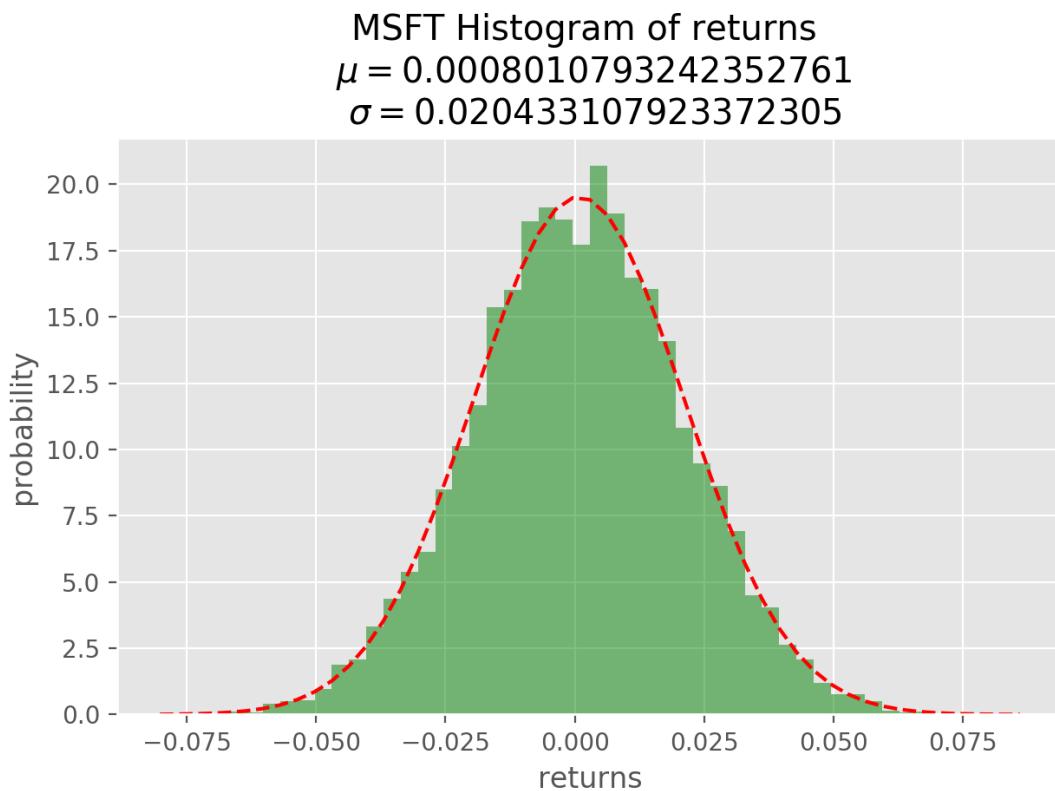


FIGURE 4.23: MSFT AR histogram of returns

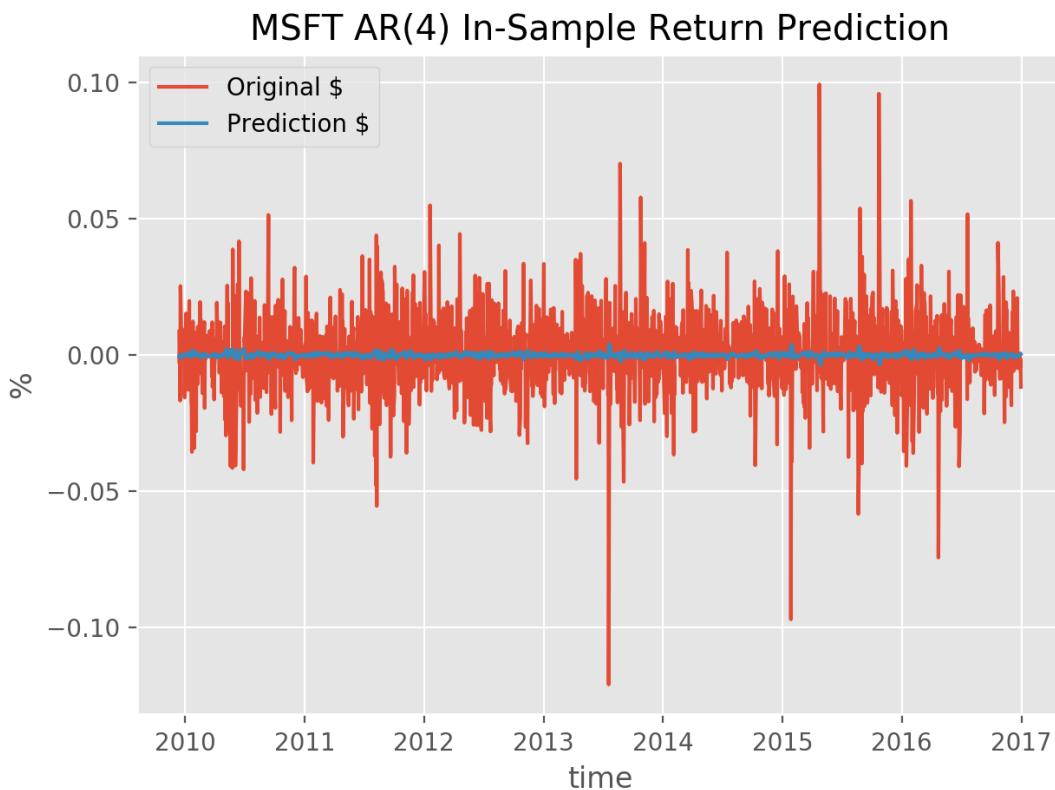


FIGURE 4.24: MSFT AR in-sample returns prediction

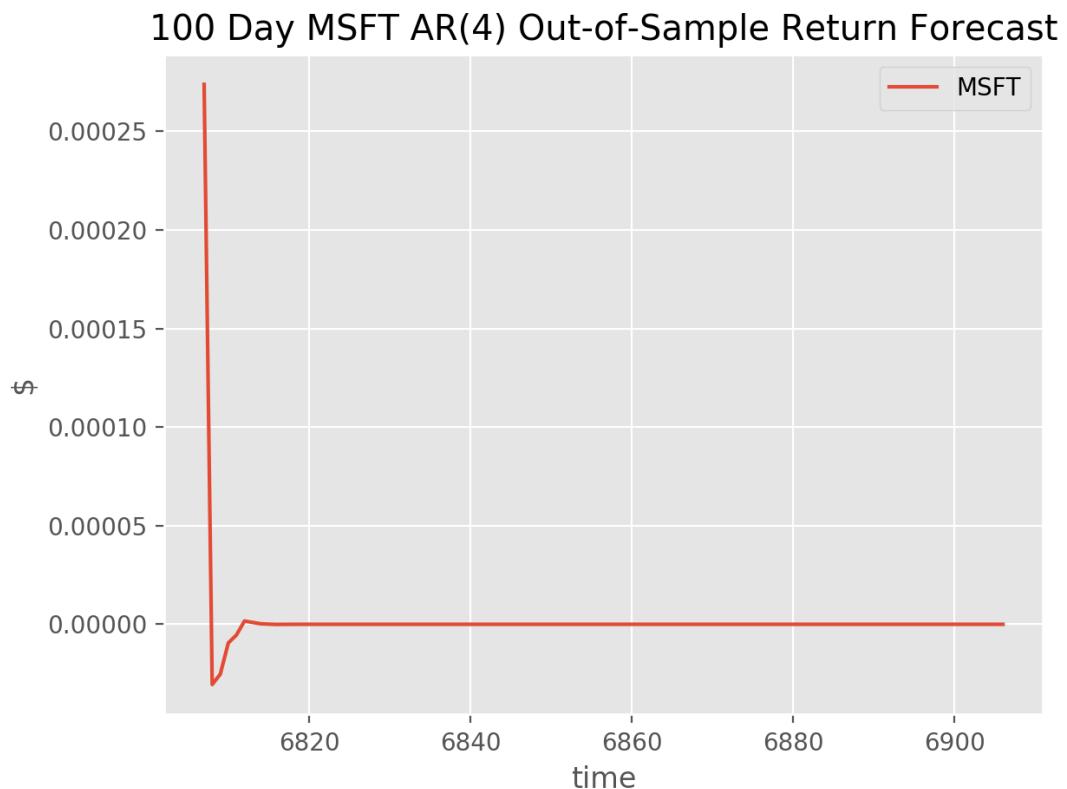


FIGURE 4.25: 100 day MSFT AR in-sample returns forecast

CDE scored sharpe ratios of -1.050 for the original returns, and -0.239 for the predicted returns in the in-sample test.

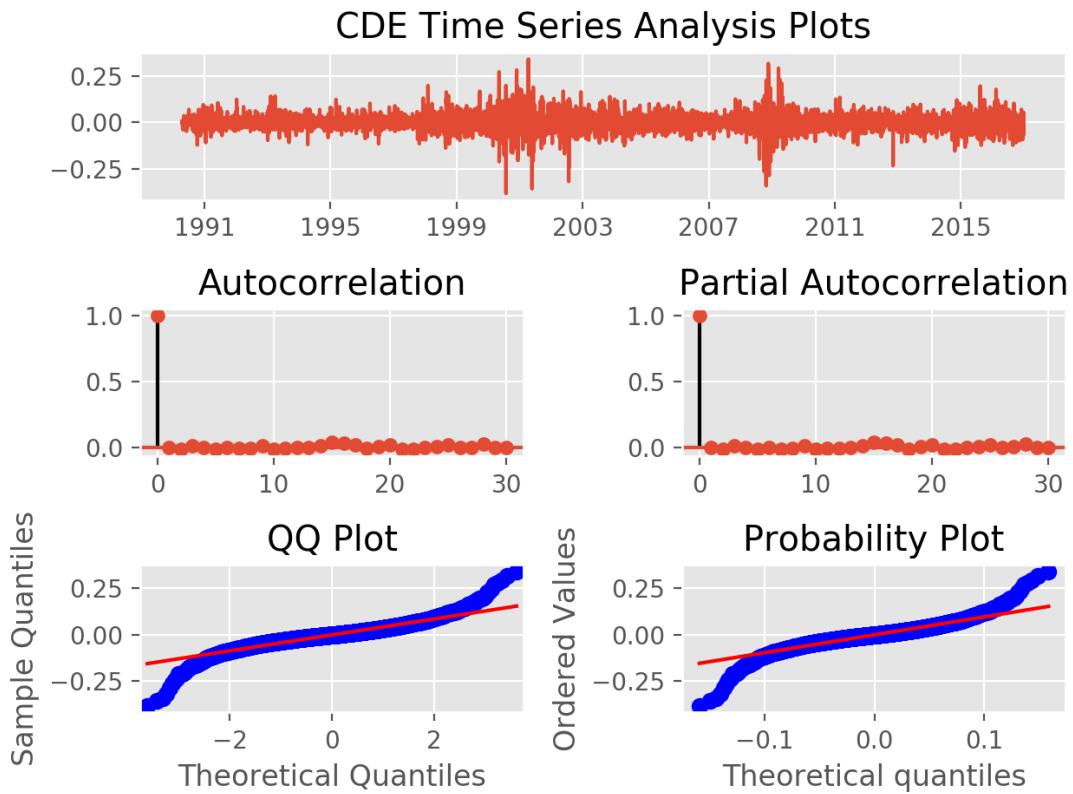


FIGURE 4.26: CDE AR time series analysis

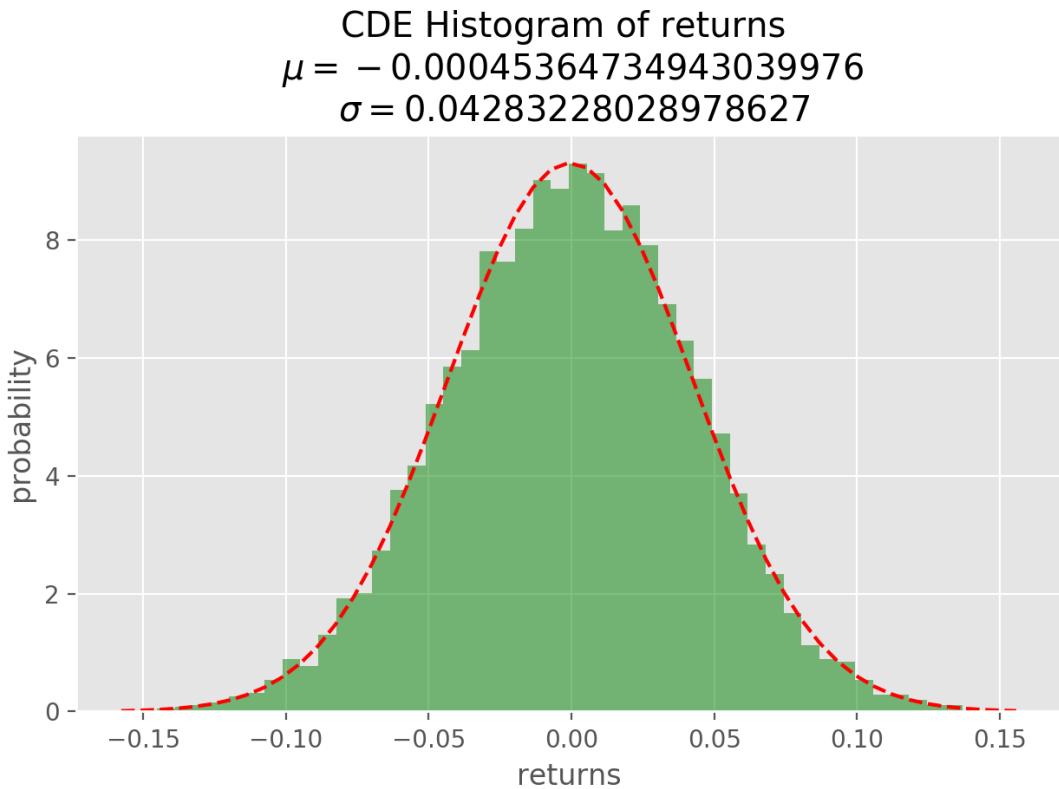


FIGURE 4.27: CDE AR histogram of returns

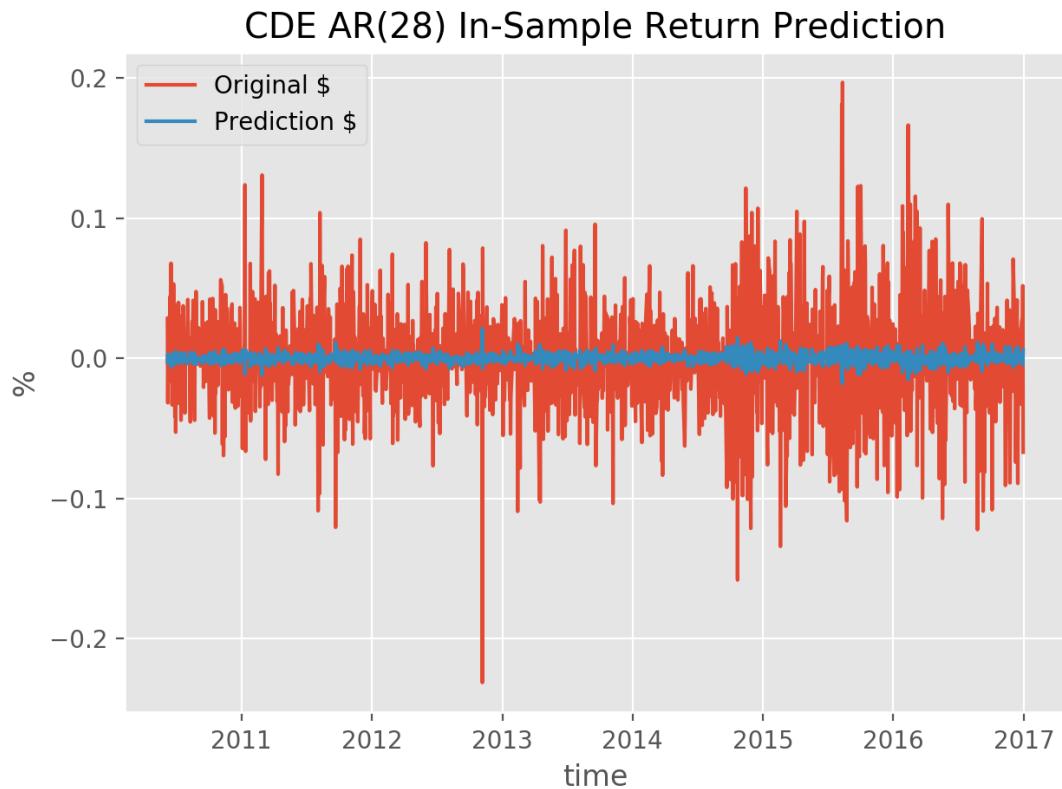


FIGURE 4.28: CDE AR in-sample returns prediction

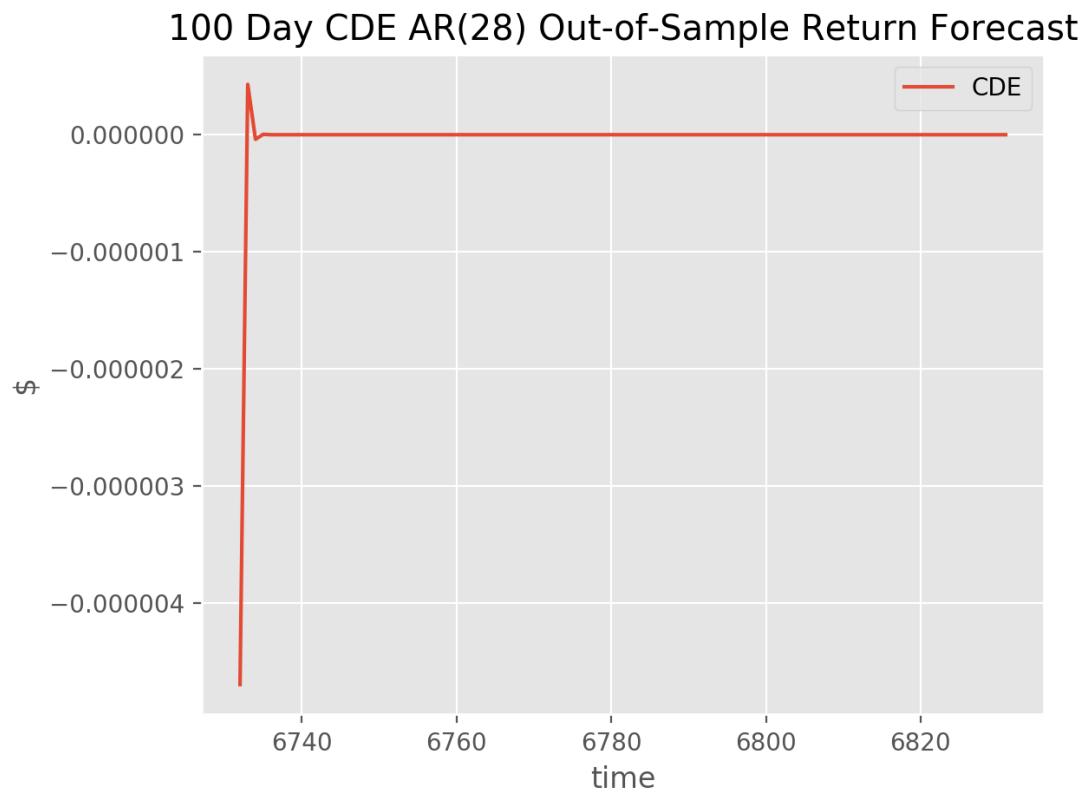


FIGURE 4.29: 100 day CDE AR in-sample returns forecast

NAVB scored sharpe ratios of -0.410 for the original returns, and 0.124 for the predicted returns in the in-sample test.

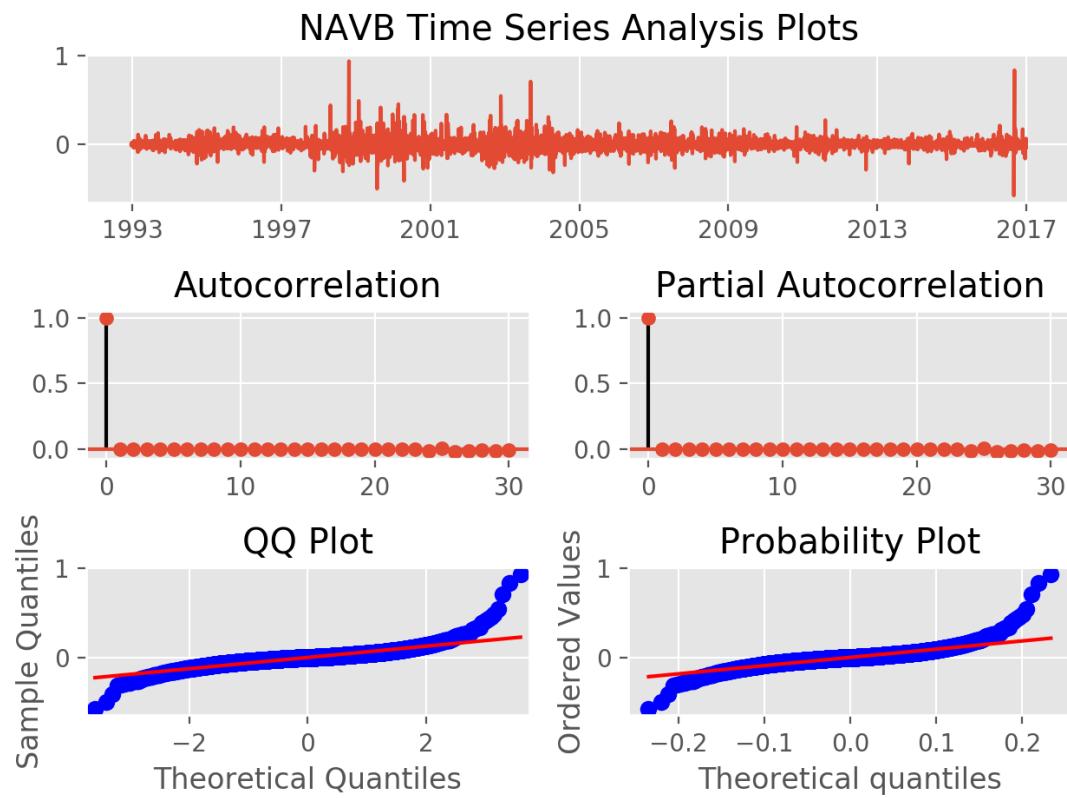


FIGURE 4.30: NAVB AR time series analysis

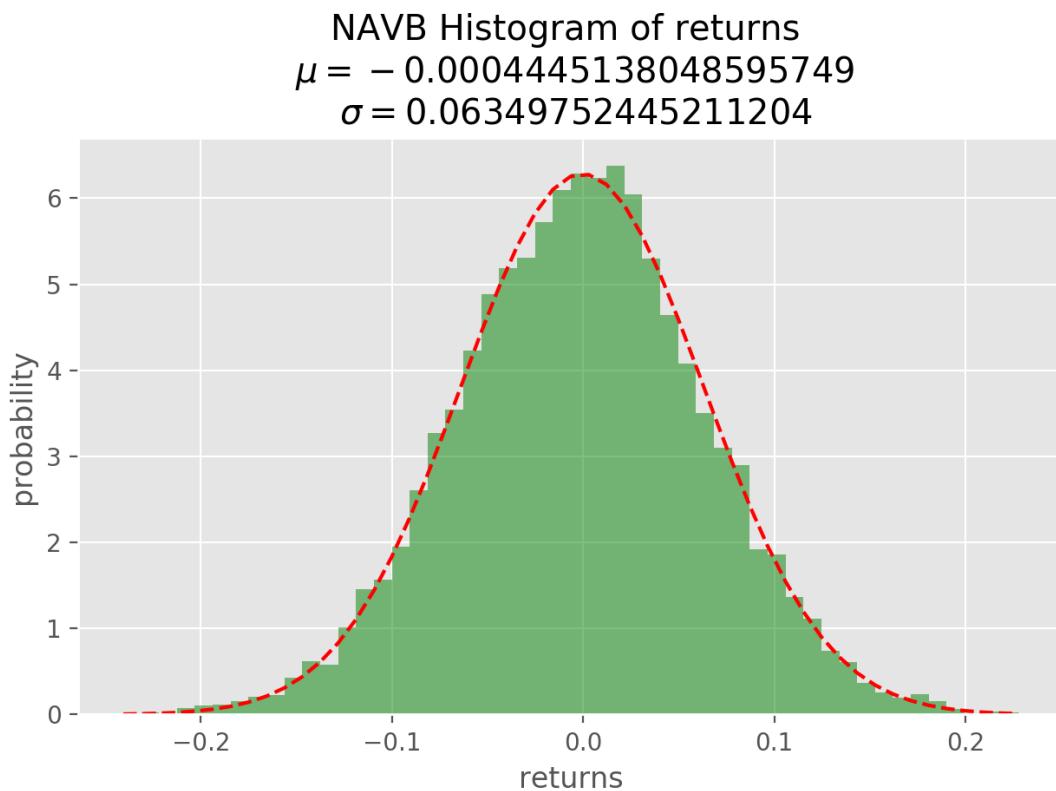


FIGURE 4.31: NAVB AR histogram of returns

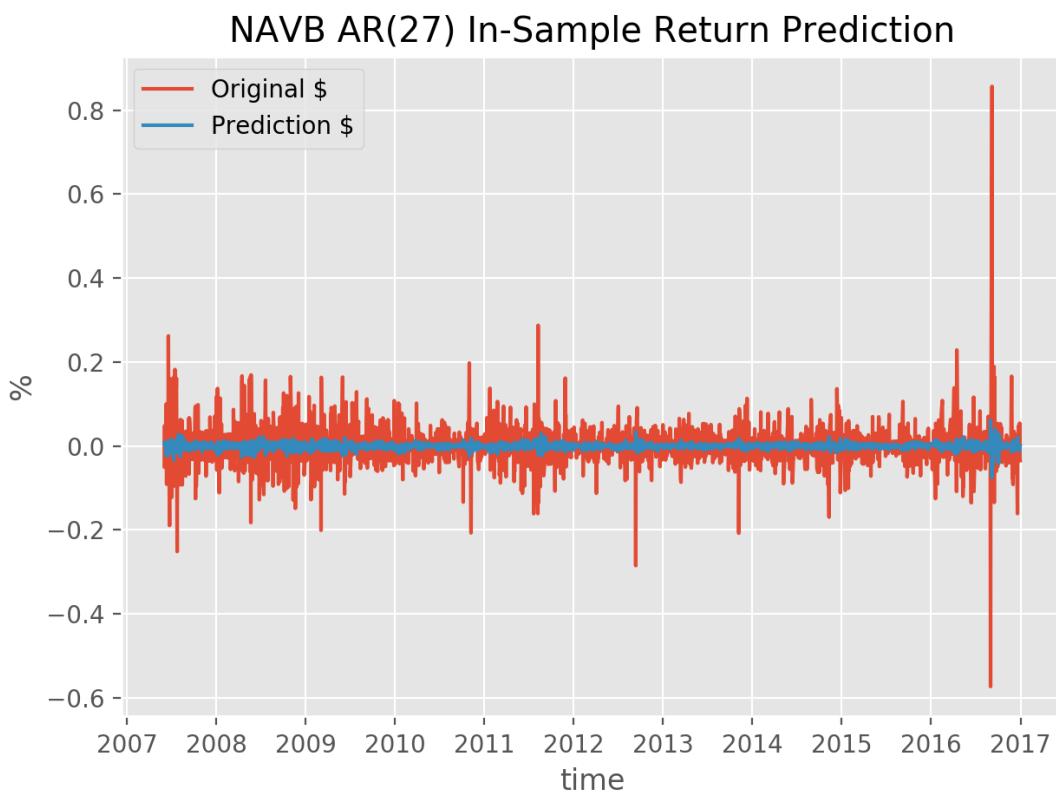


FIGURE 4.32: NAVB AR in-sample returns prediction

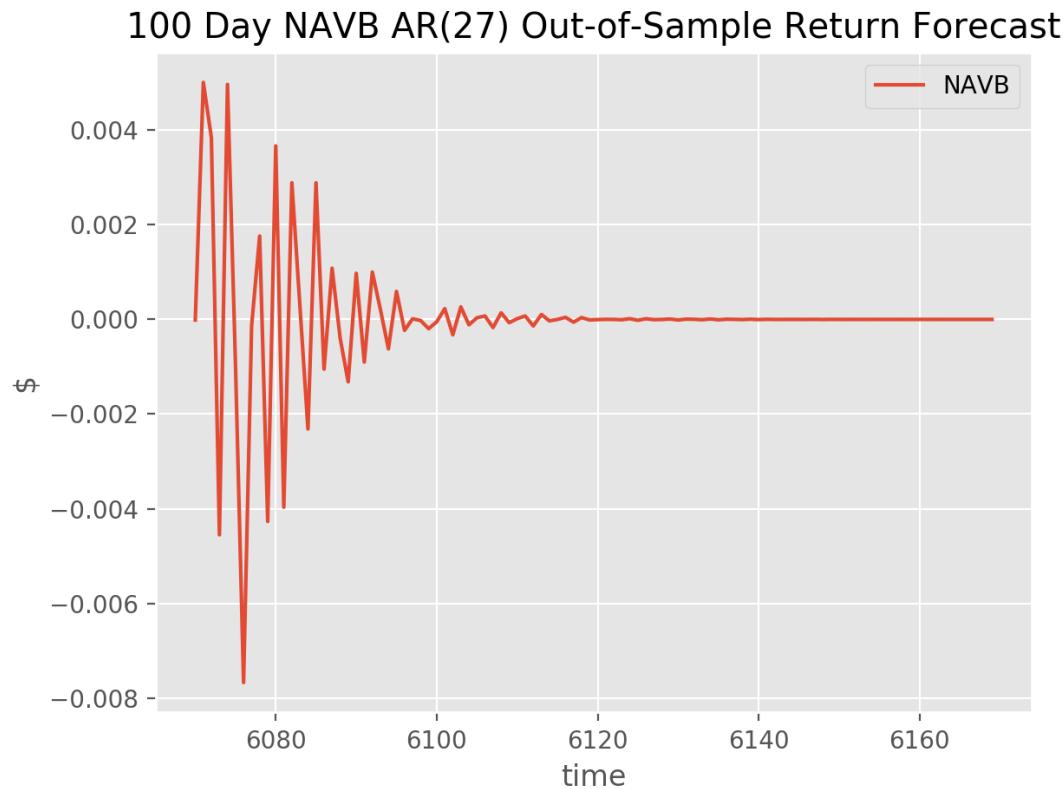


FIGURE 4.33: 100 day NAVB AR in-sample returns forecast

HRG scored sharpe ratios of 0.627 for the original returns, and -2.603 for the predicted returns in the in-sample test.

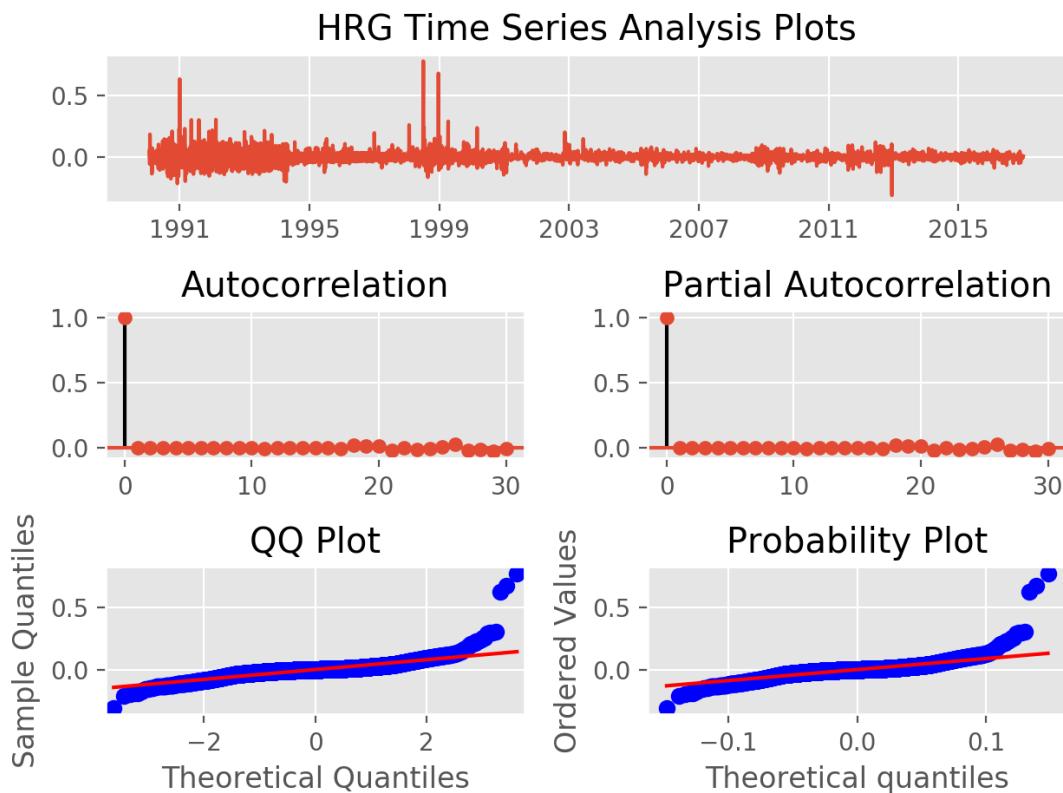


FIGURE 4.34: HRG AR time series analysis

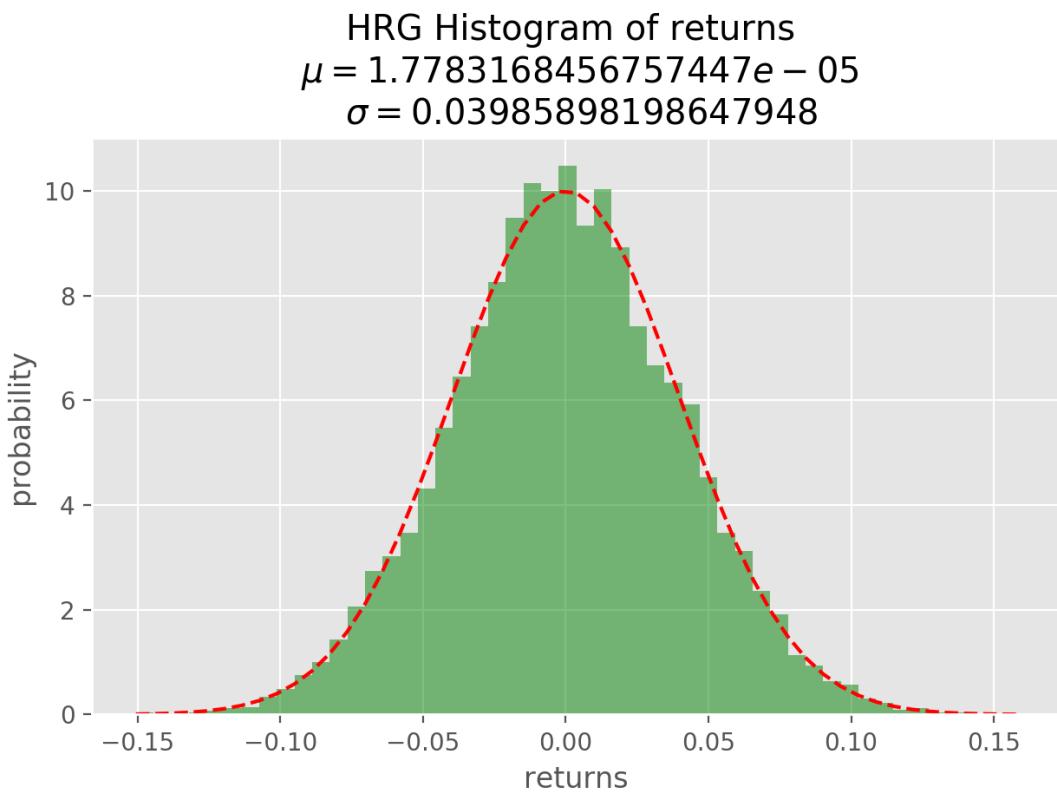


FIGURE 4.35: HRG AR histogram of returns

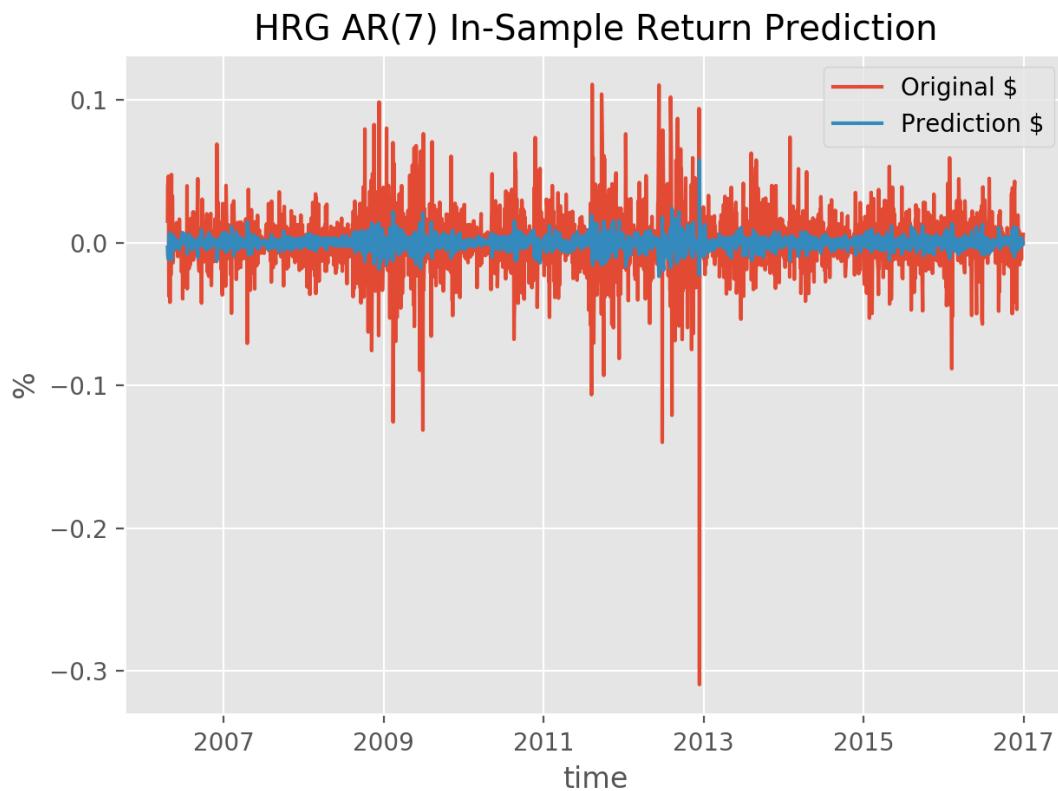


FIGURE 4.36: HL AR in-sample returns prediction

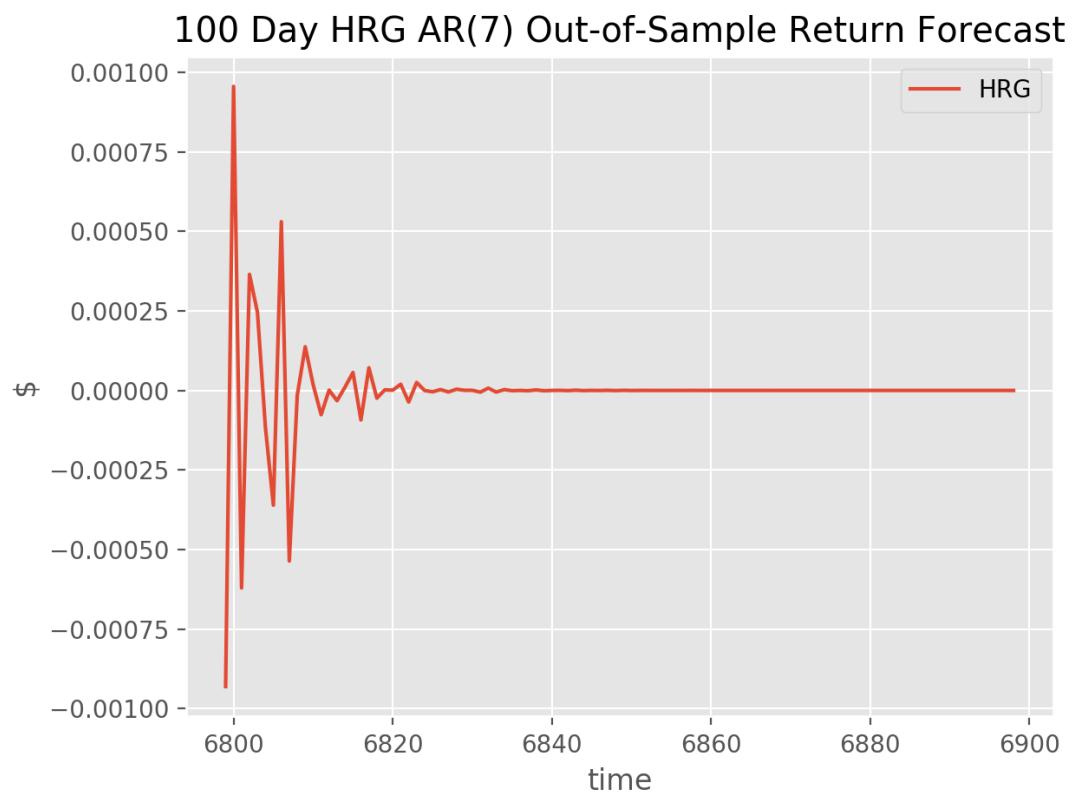


FIGURE 4.37: 100 day HRG AR in-sample returns forecast

HL scored sharpe ratios of 0.695 for the original returns, and -1.880 for the predicted returns in the in-sample test.

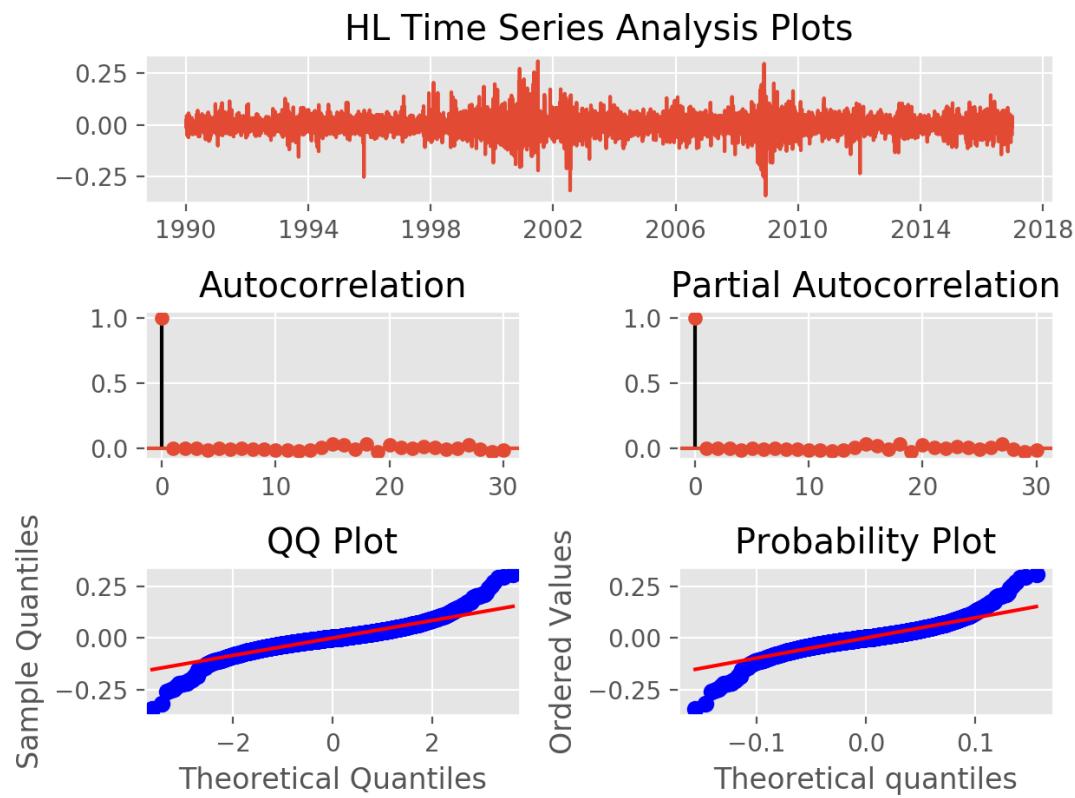


FIGURE 4.38: HL AR time series analysis

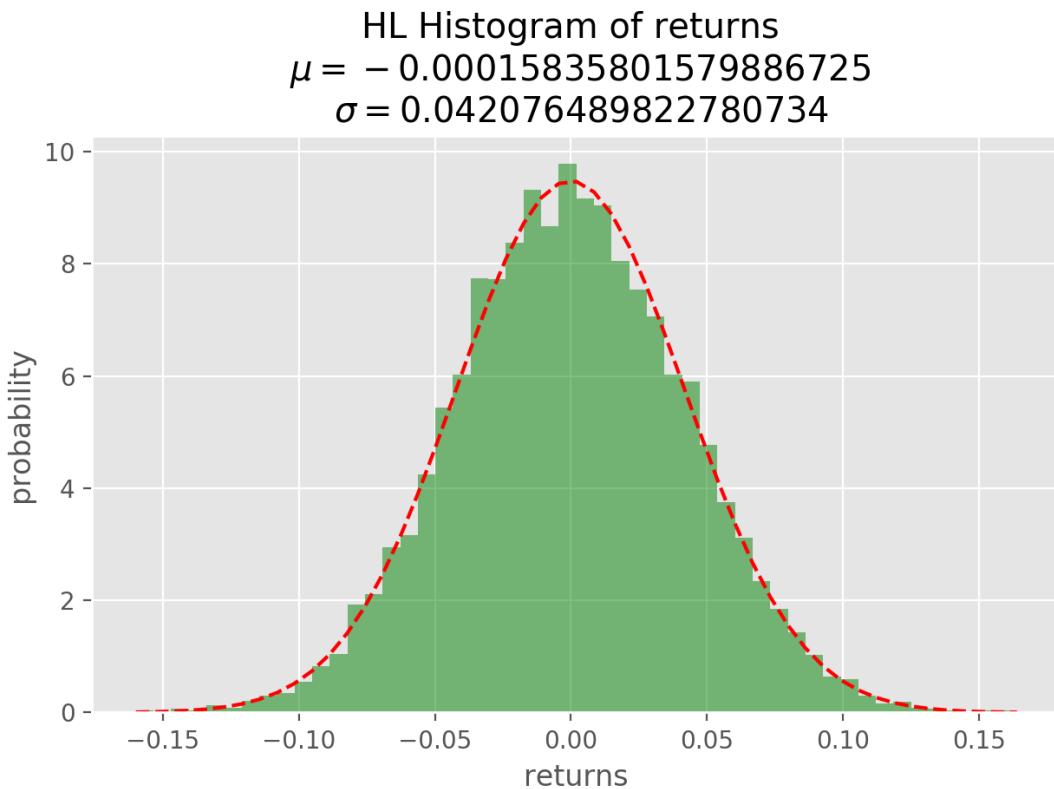


FIGURE 4.39: HL AR histogram of returns

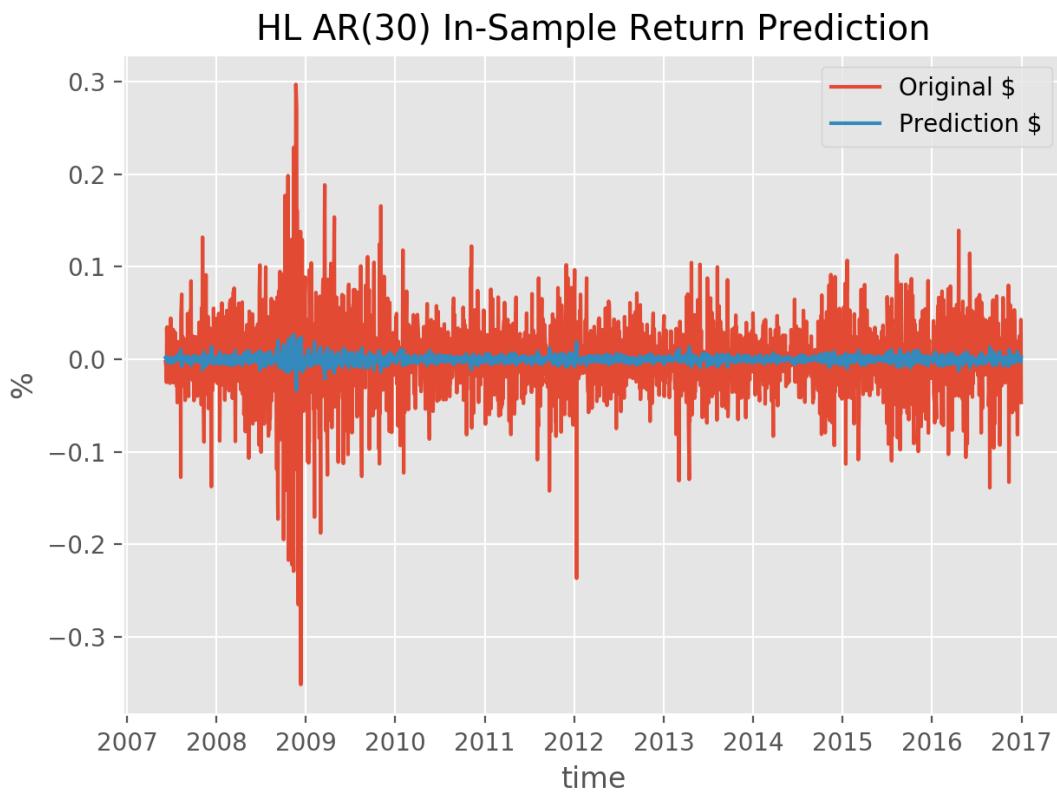


FIGURE 4.40: HL AR in-sample returns prediction

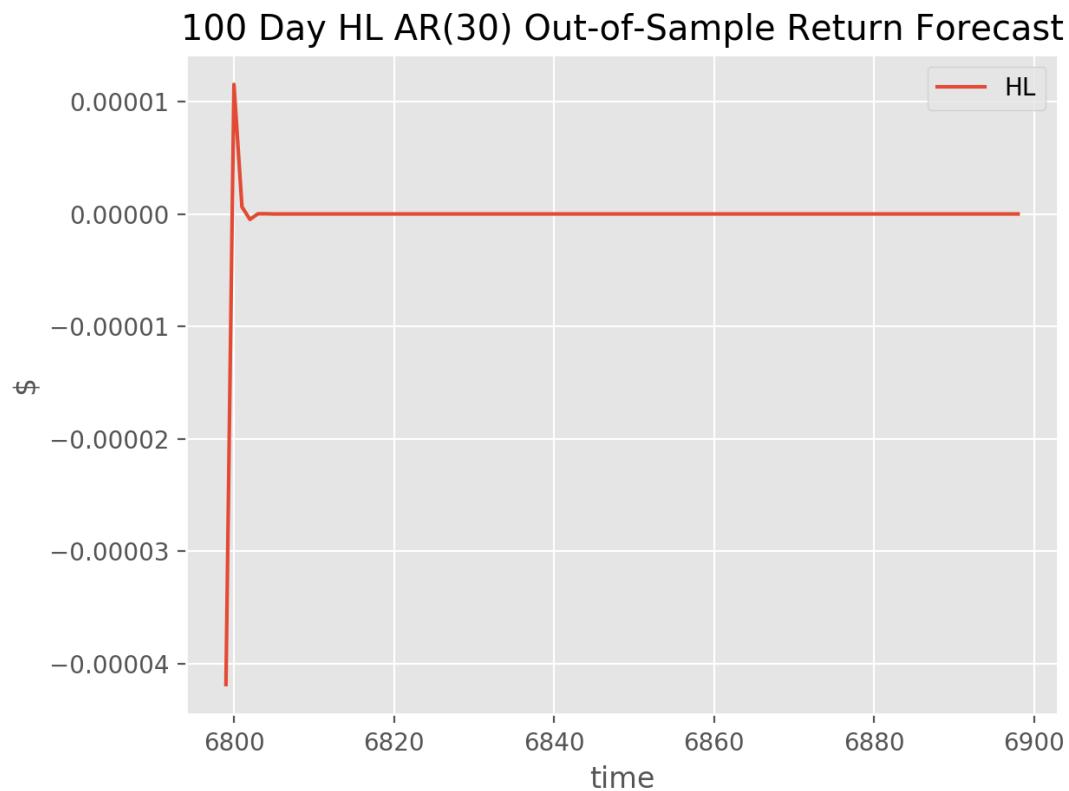


FIGURE 4.41: 100 day HL AR in-sample returns forecast

4.1.4 Moving Average (MA)

MSFT scored sharpe ratios of 0.358 for the original returns, and -5.610 for the predicted returns in the in-sample test.

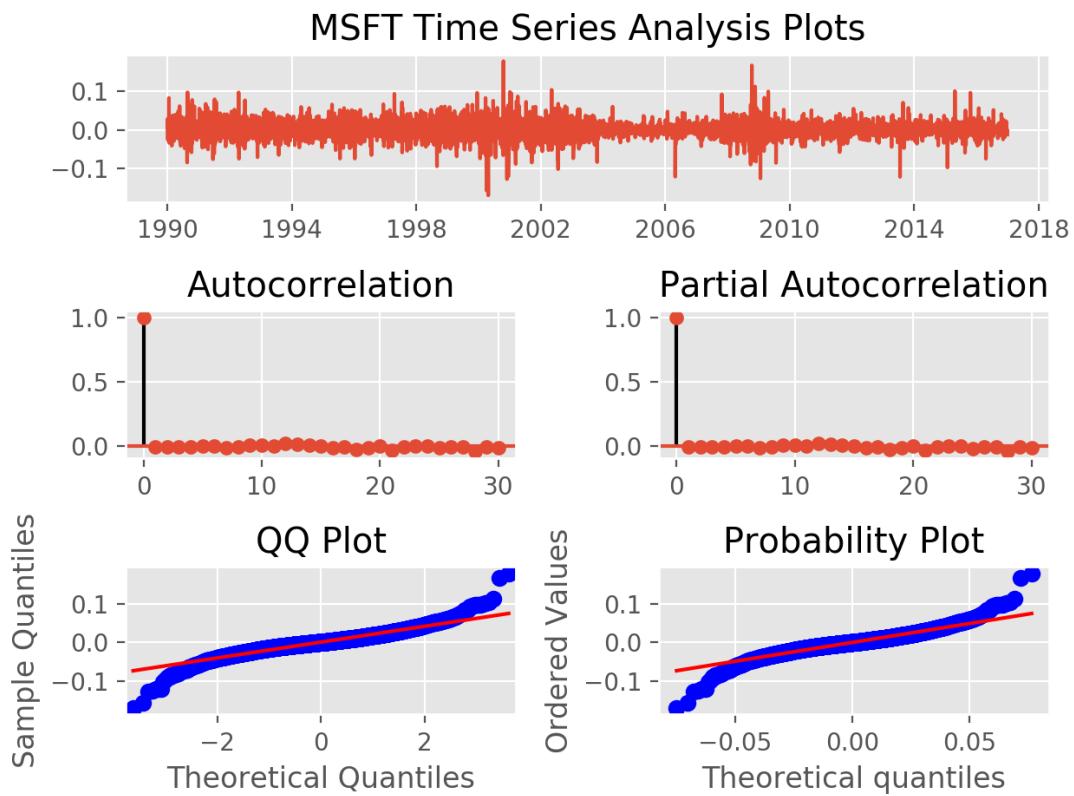


FIGURE 4.42: MSFT MA time series analysis

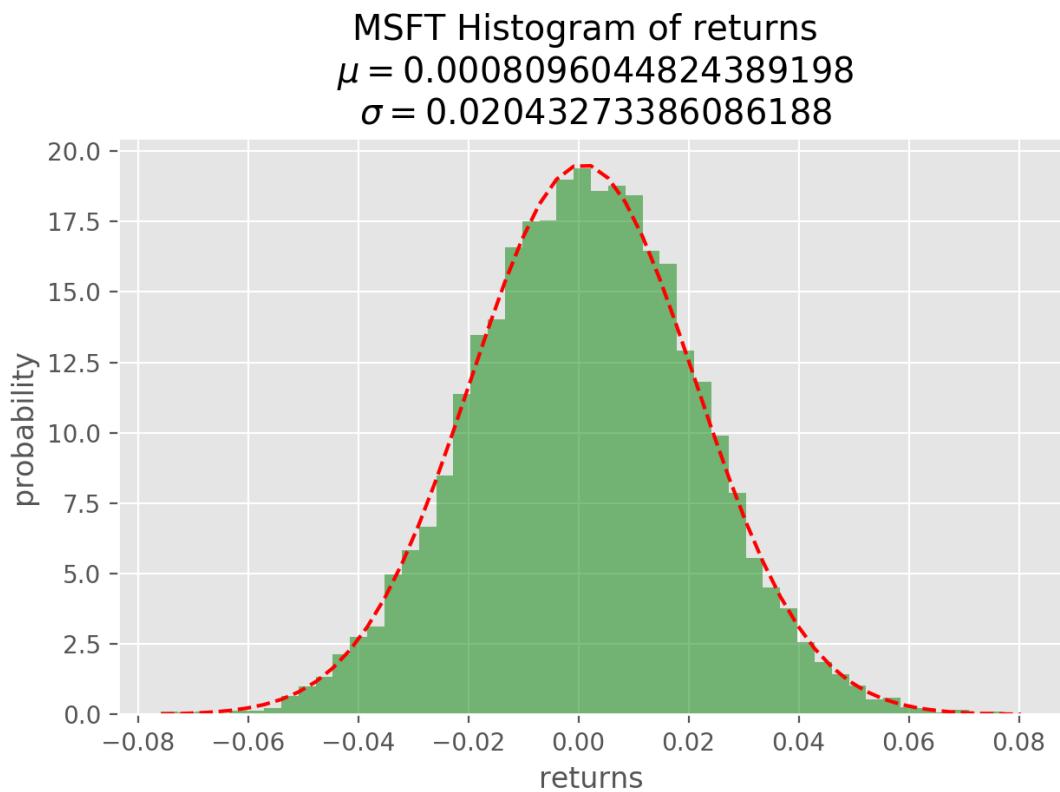


FIGURE 4.43: MSFT MA histogram of returns

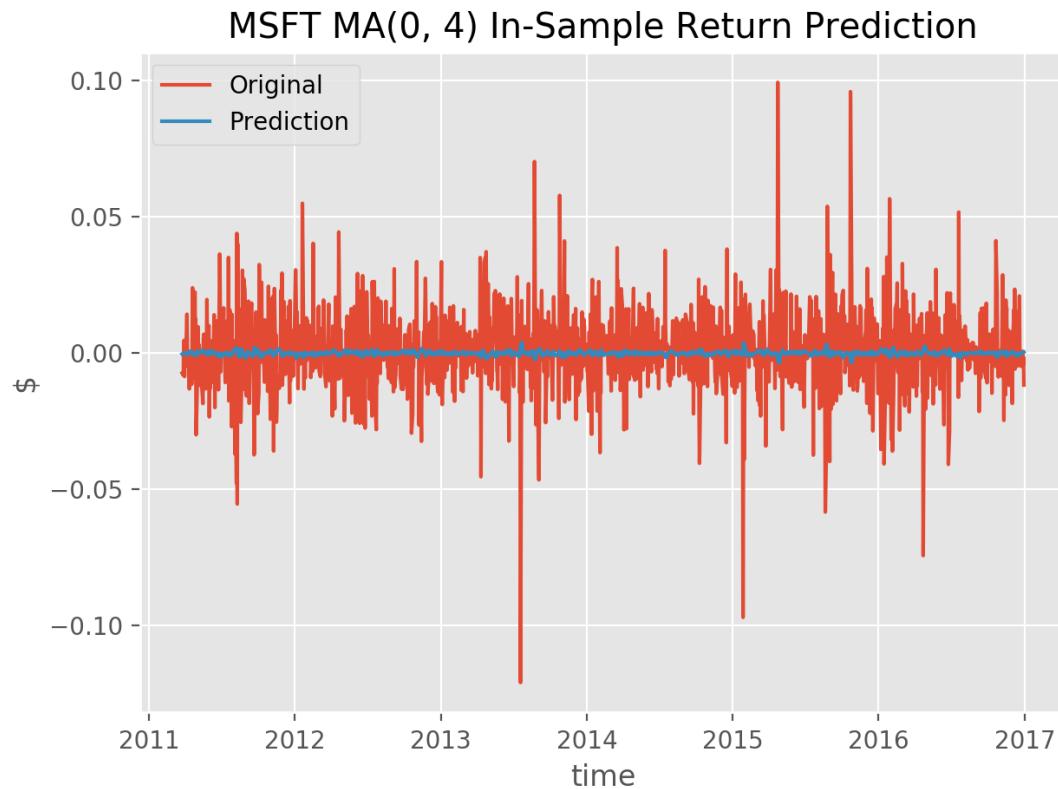


FIGURE 4.44: MSFT MA in-sample returns prediction

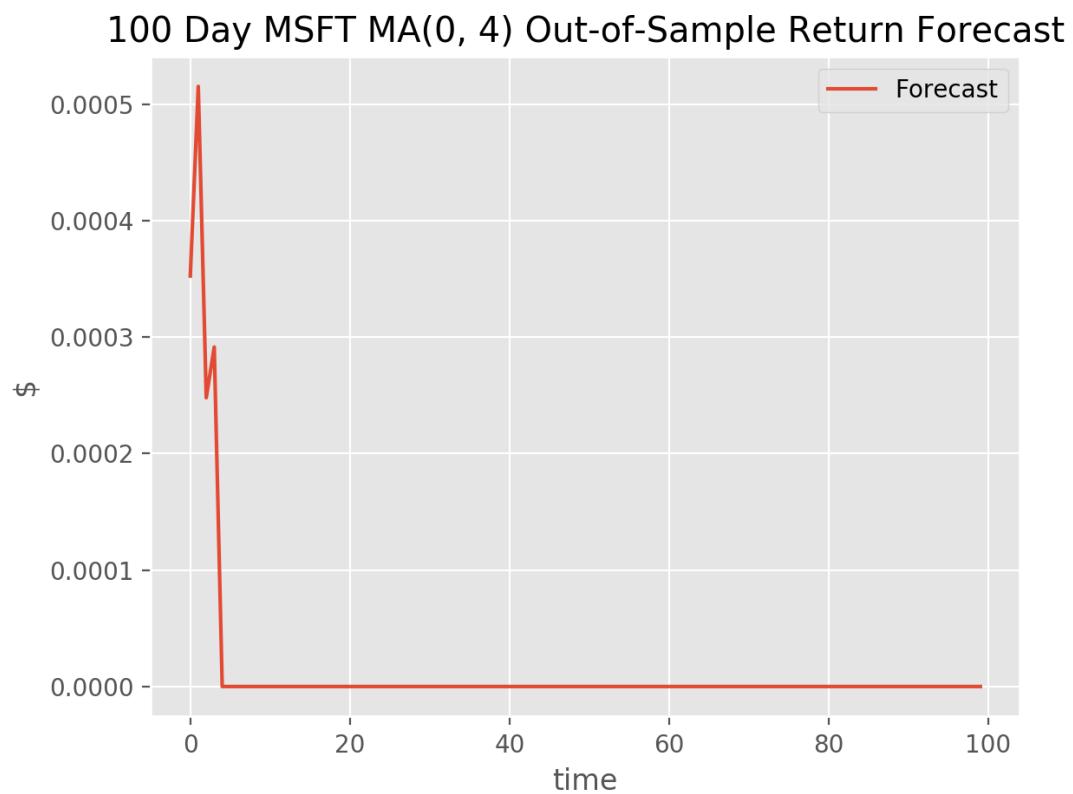


FIGURE 4.45: 100 day MSFT MA in-sample returns forecast

CDE scored sharpe ratios of -0.199 for the original returns, and -0.760 for the predicted returns in the in-sample test.

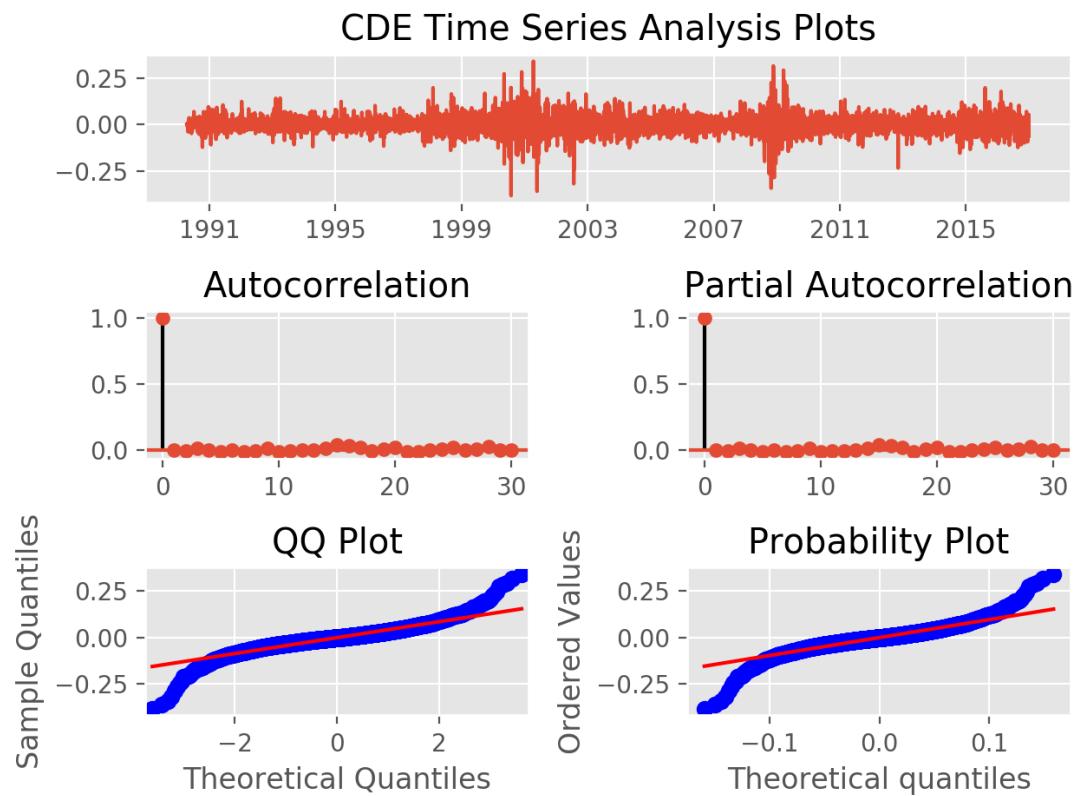


FIGURE 4.46: CDE MA time series analysis

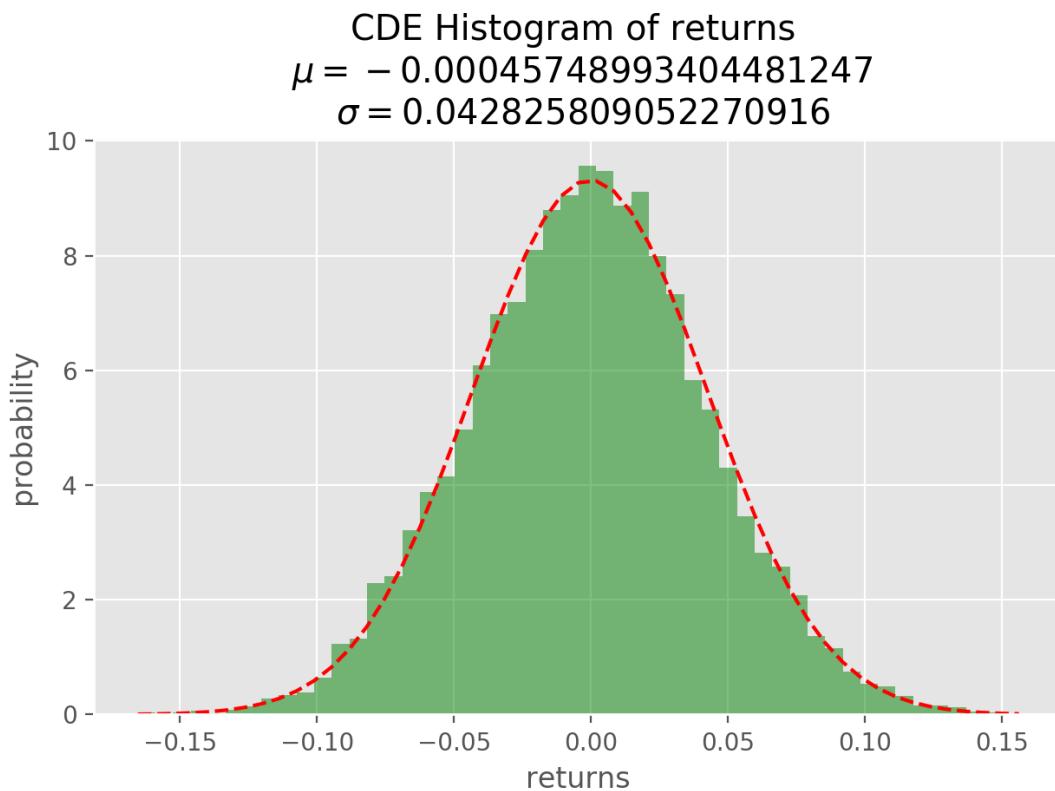


FIGURE 4.47: CDE MA histogram of returns

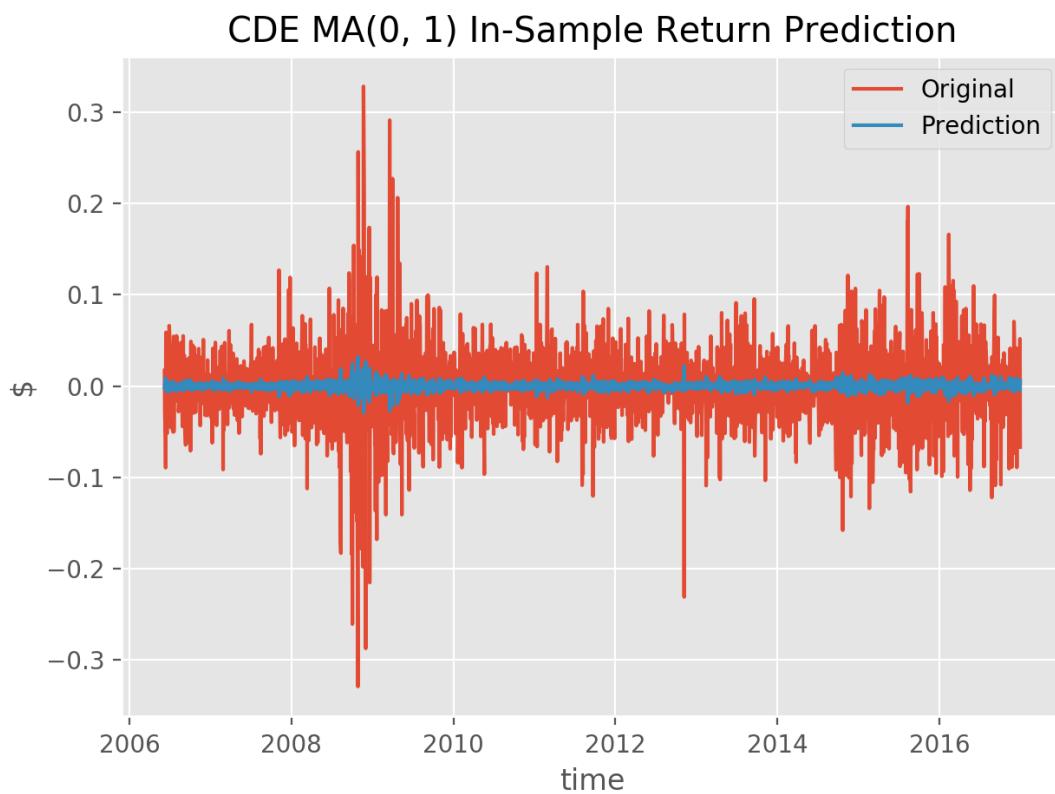


FIGURE 4.48: CDE MA in-sample returns prediction

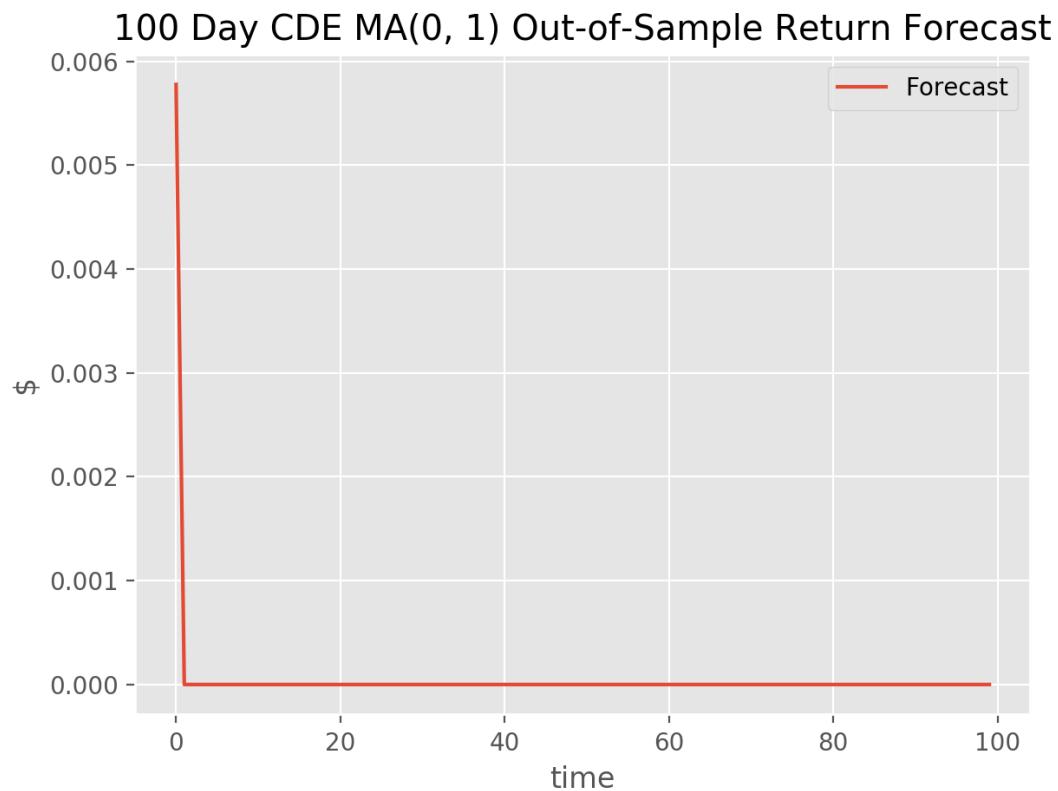


FIGURE 4.49: 100 day CDE MA in-sample returns forecast

NAV B scored sharpe ratios of -0.067 for the original returns, and 0.590 for the predicted returns in the in-sample test.

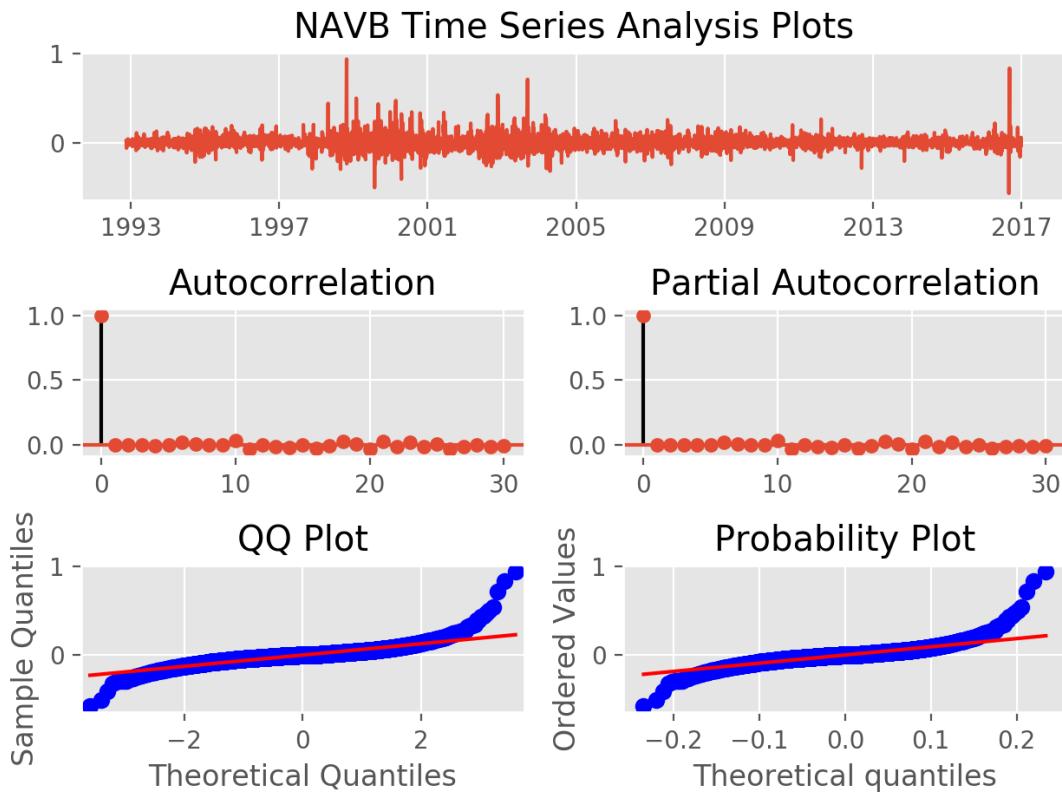


FIGURE 4.50: NAVB MA time series analysis

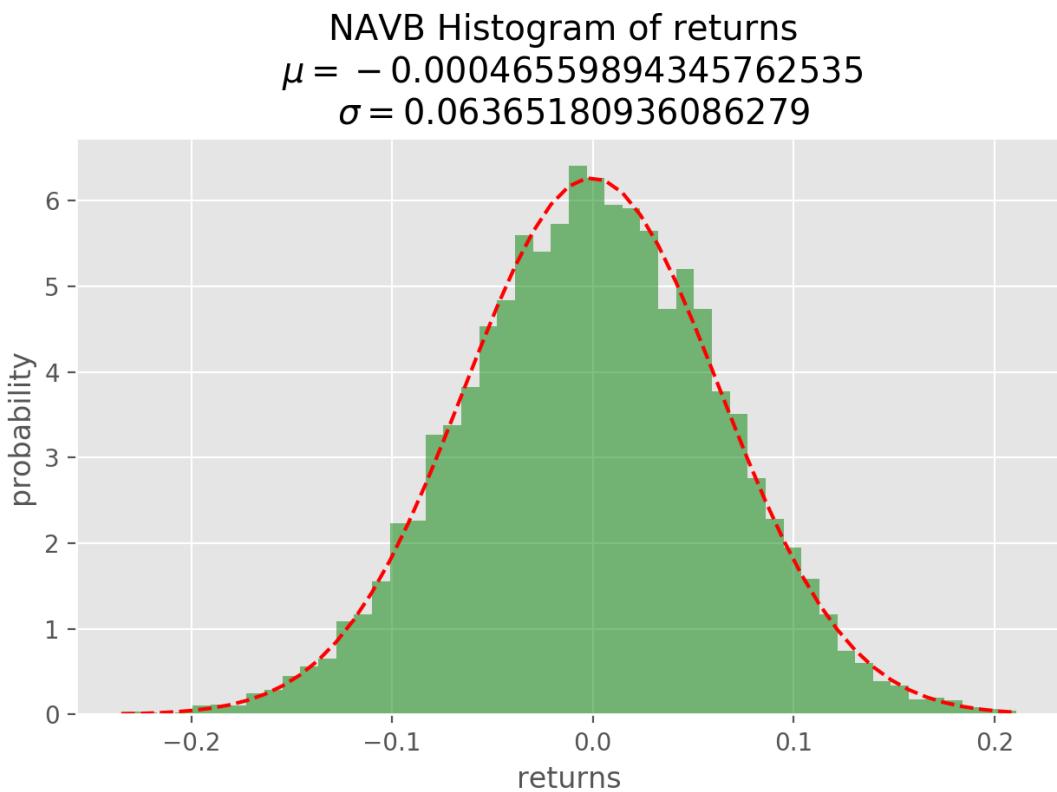


FIGURE 4.51: NAVB MA histogram of returns

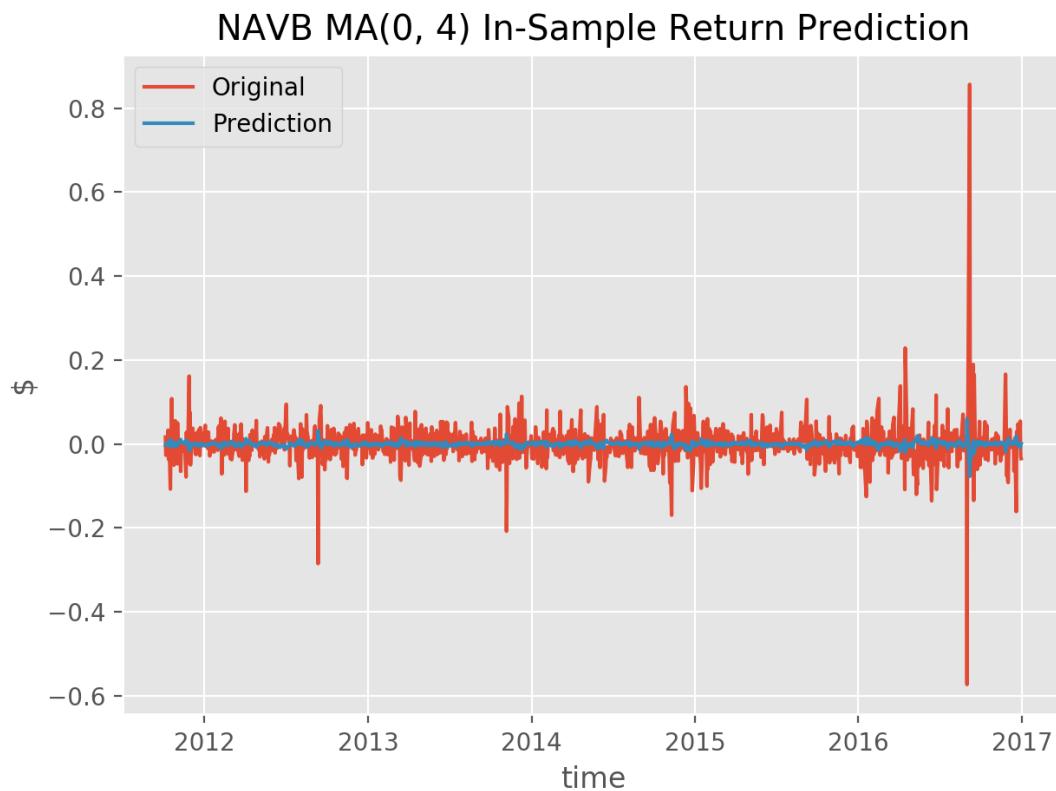


FIGURE 4.52: NAVB MA in-sample returns prediction

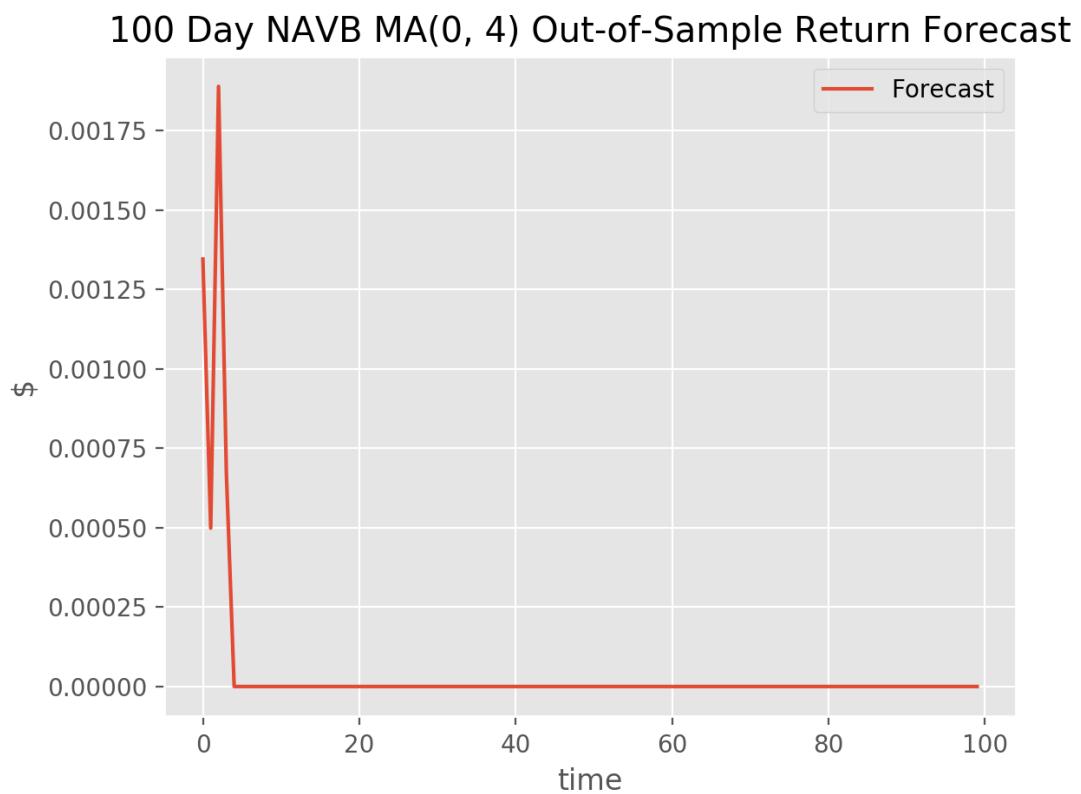


FIGURE 4.53: 100 day NAVB MA in-sample returns forecast

HRG scored sharpe ratios of 0.084 for the original returns, and -1.020 for the predicted returns in the in-sample test.

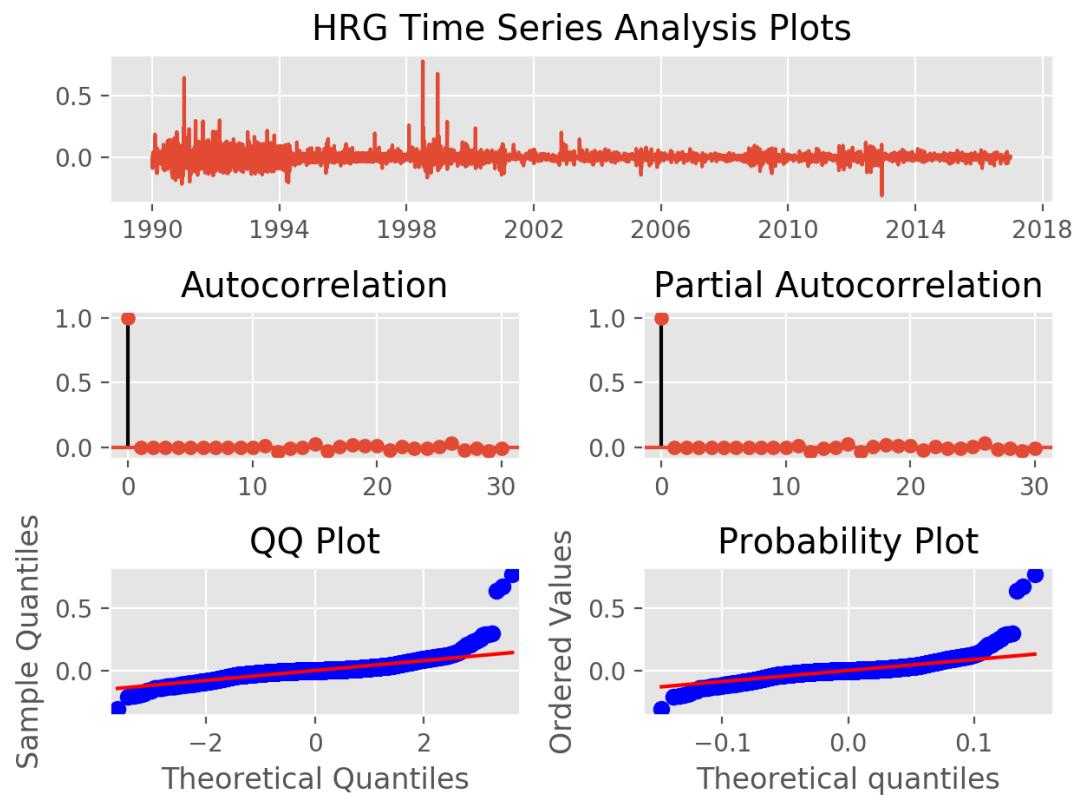


FIGURE 4.54: HRG MA time series analysis

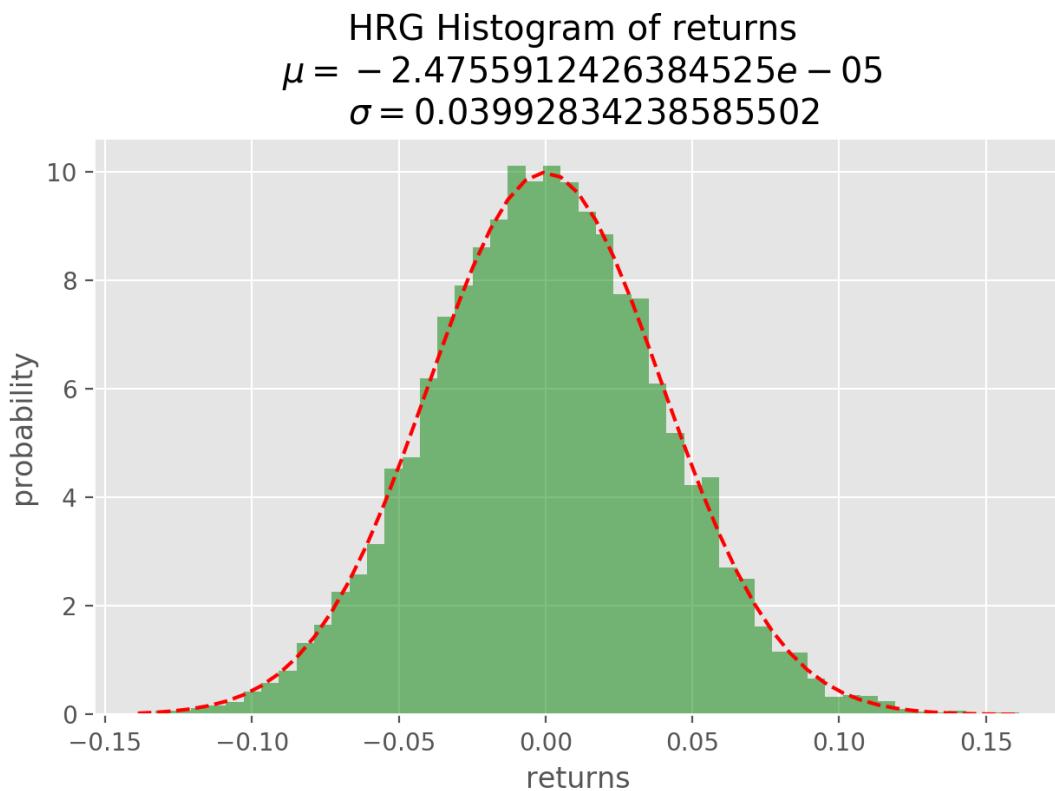


FIGURE 4.55: HRG MA histogram of returns

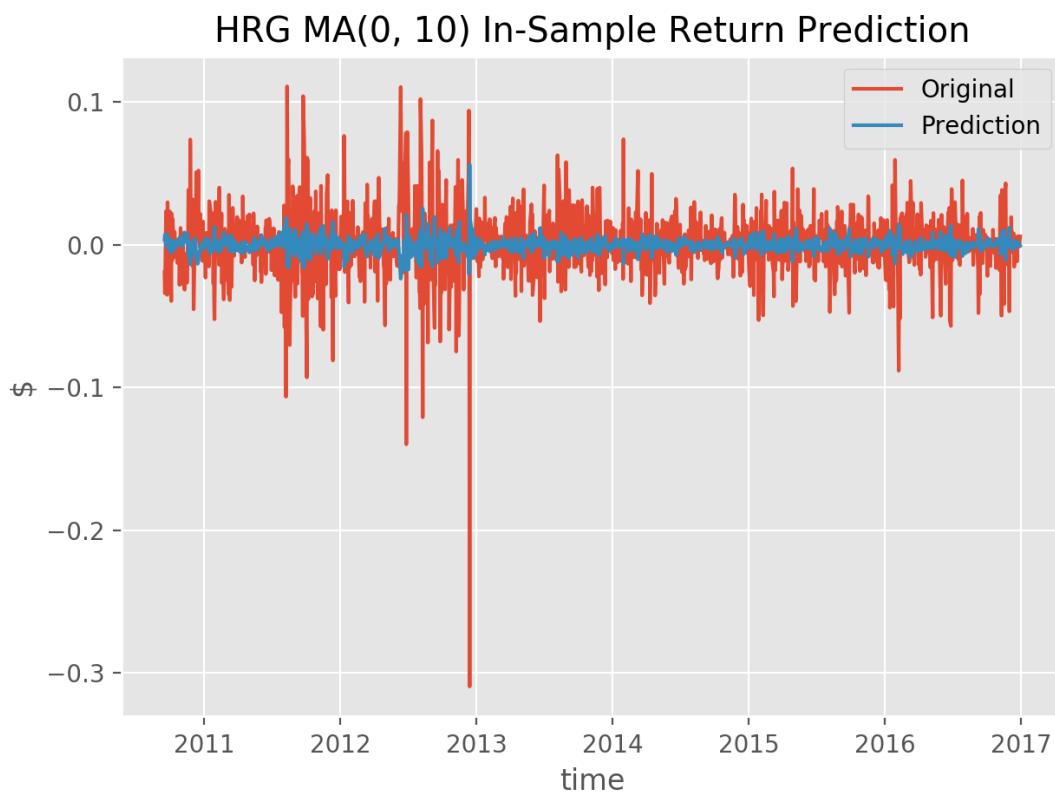


FIGURE 4.56: HRG MA in-sample returns prediction

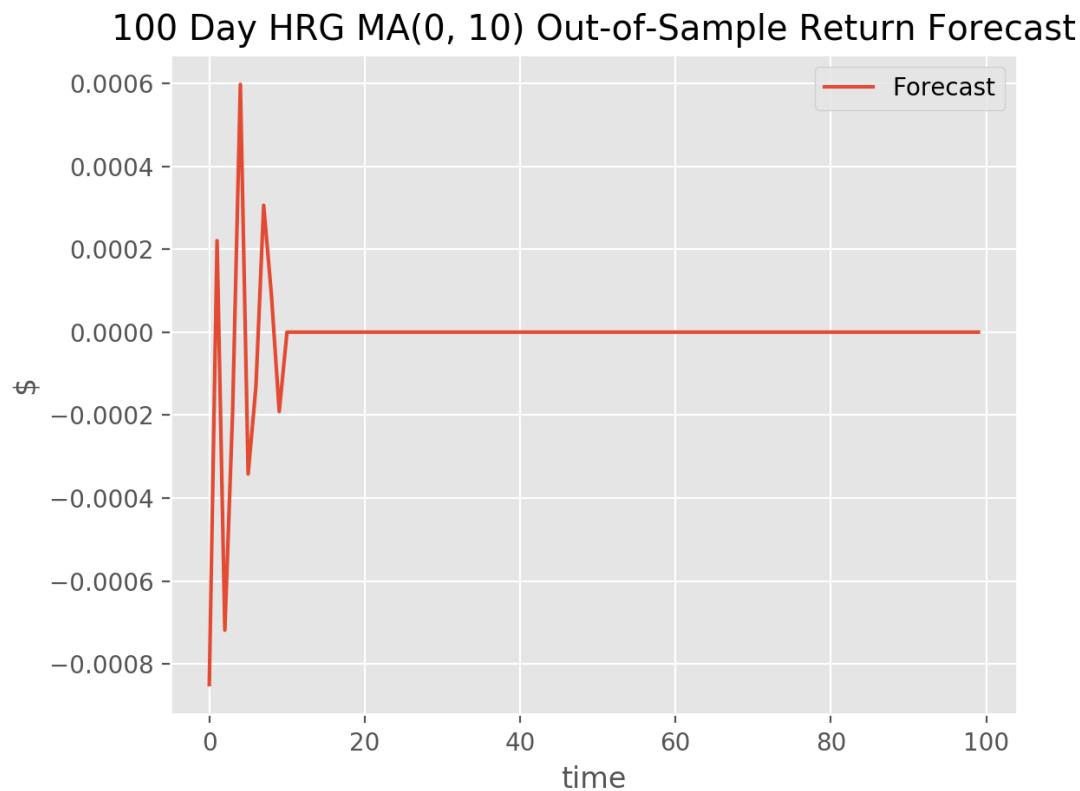


FIGURE 4.57: 100 day HRG MA in-sample returns forecast

HL scored sharpe ratios of -0.128 for the original returns, and -0.977 for the predicted returns in the in-sample test.

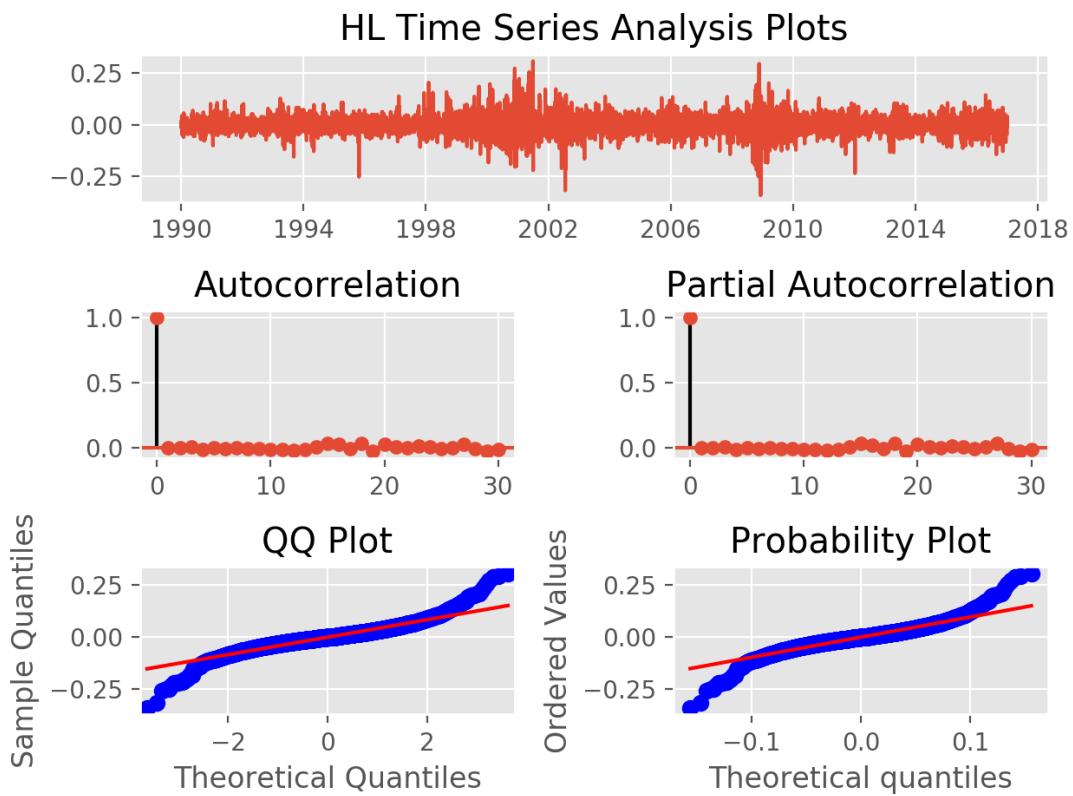


FIGURE 4.58: HL MA time series analysis

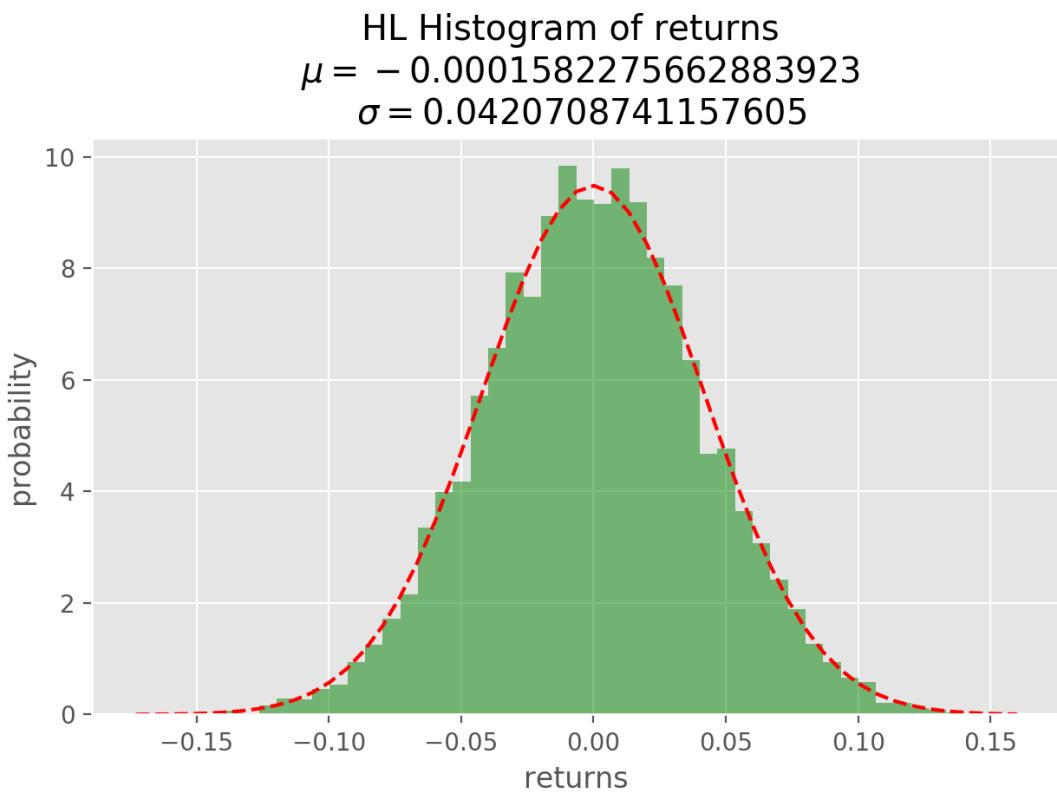


FIGURE 4.59: HL MA histogram of returns

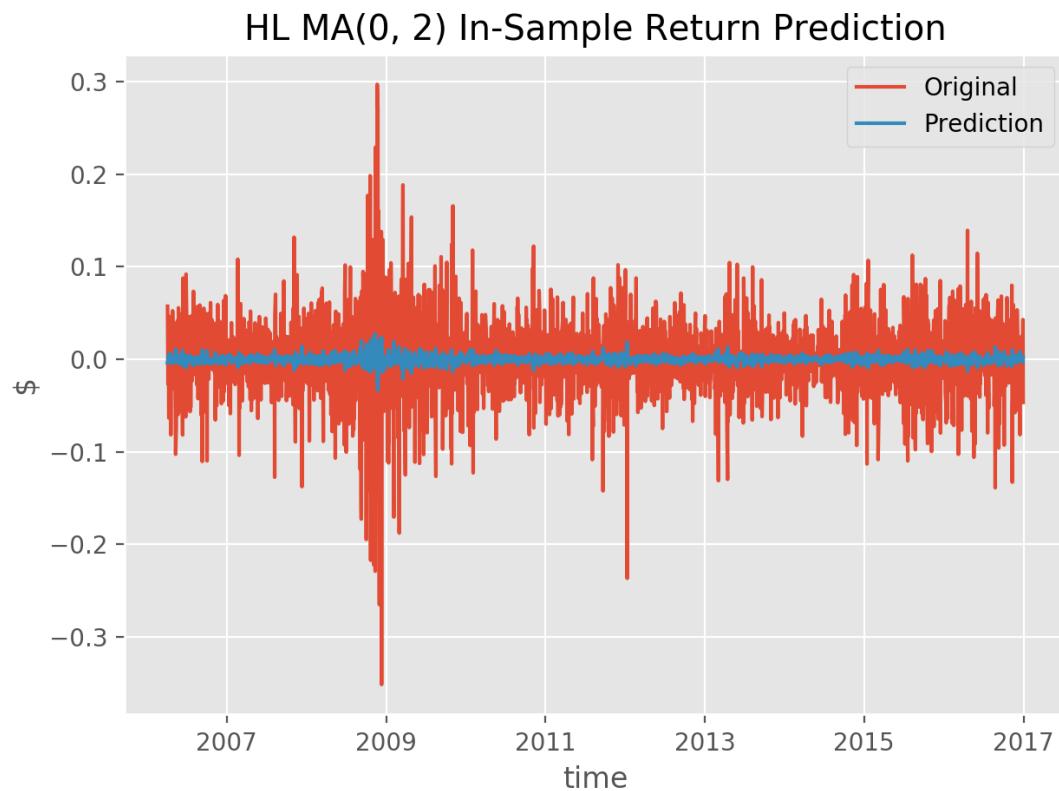


FIGURE 4.60: HL MA in-sample returns prediction

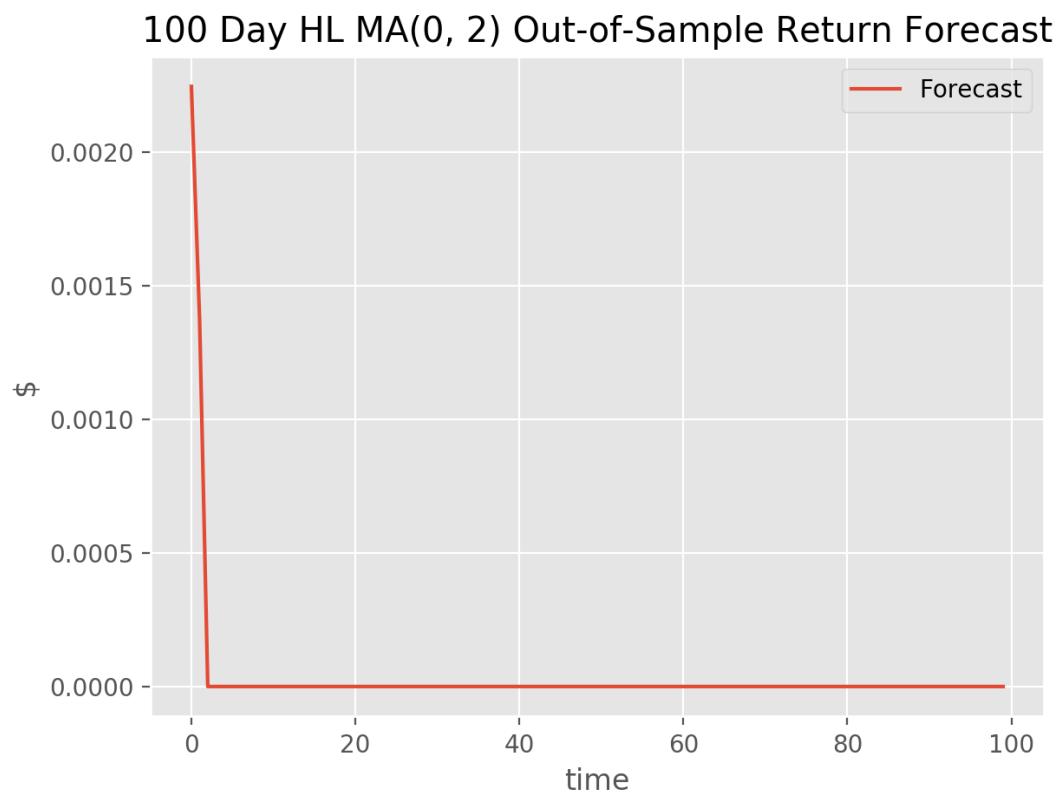


FIGURE 4.61: 100 day HL MA in-sample returns forecast

4.1.5 Auto Regressive Moving Average (ARMA)

HL scored sharpe ratios of 0.537 for the original returns, and -6.485 for the predicted returns in the in-sample test.

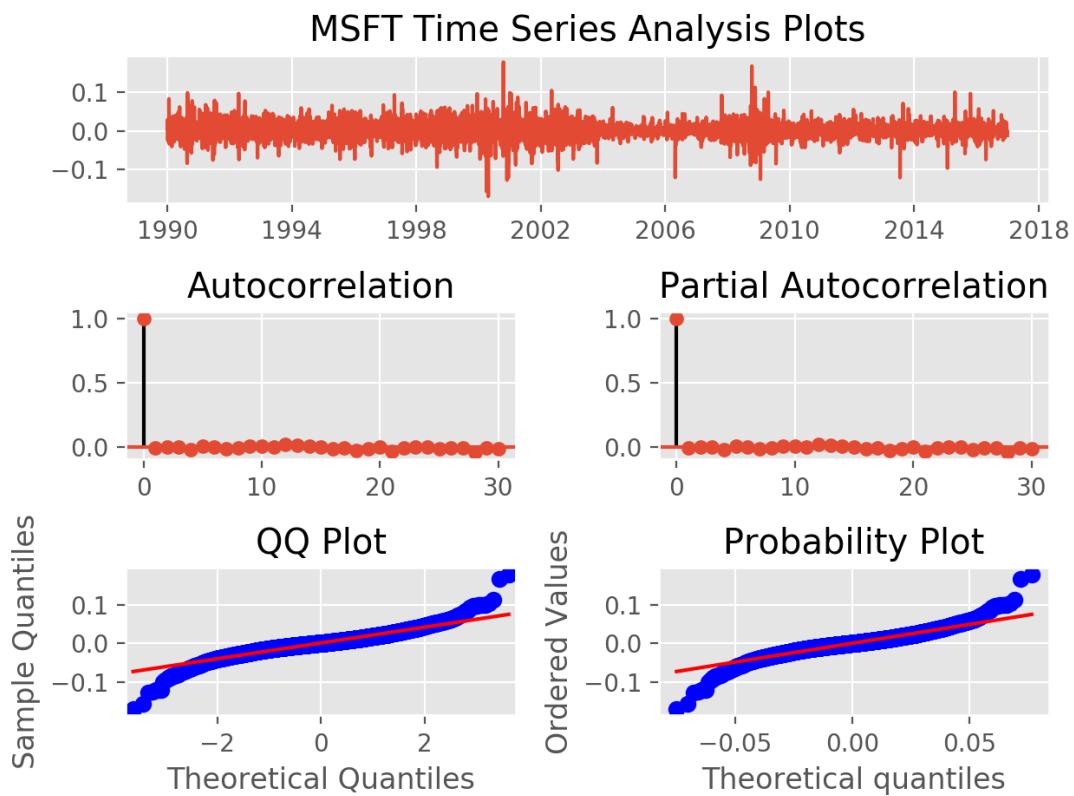


FIGURE 4.62: MSFT ARMA time series analysis

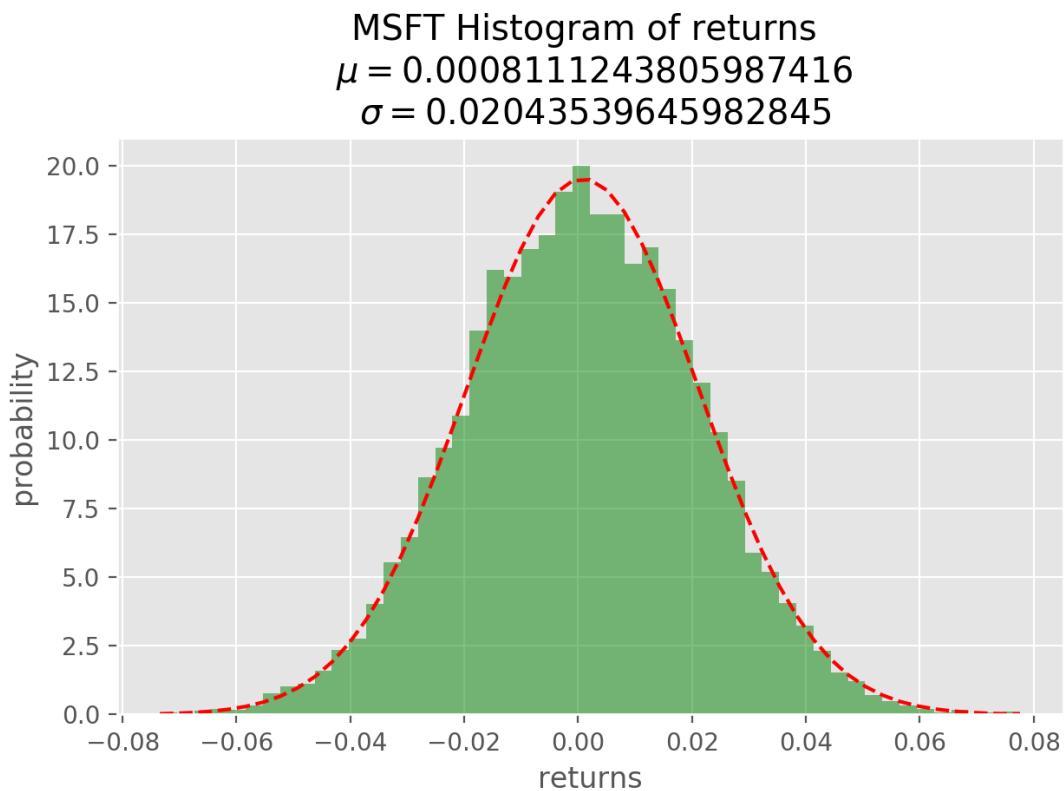


FIGURE 4.63: MSFT ARMA histogram of returns

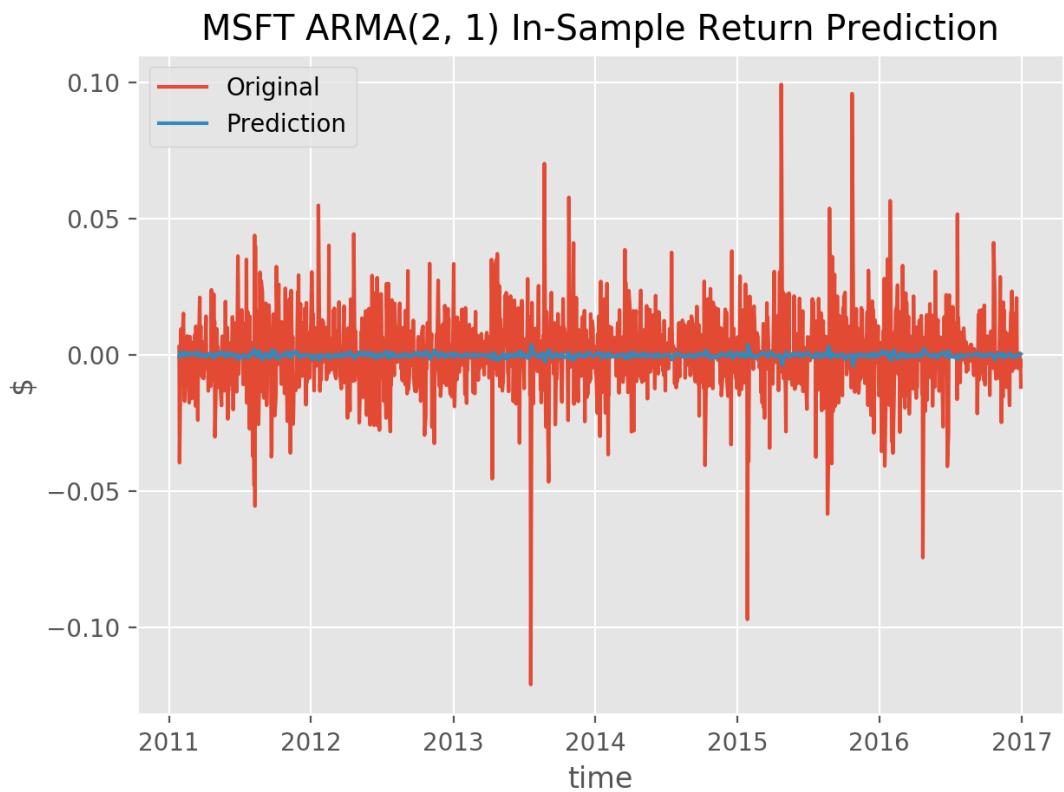


FIGURE 4.64: MSFT ARMA in-sample returns prediction

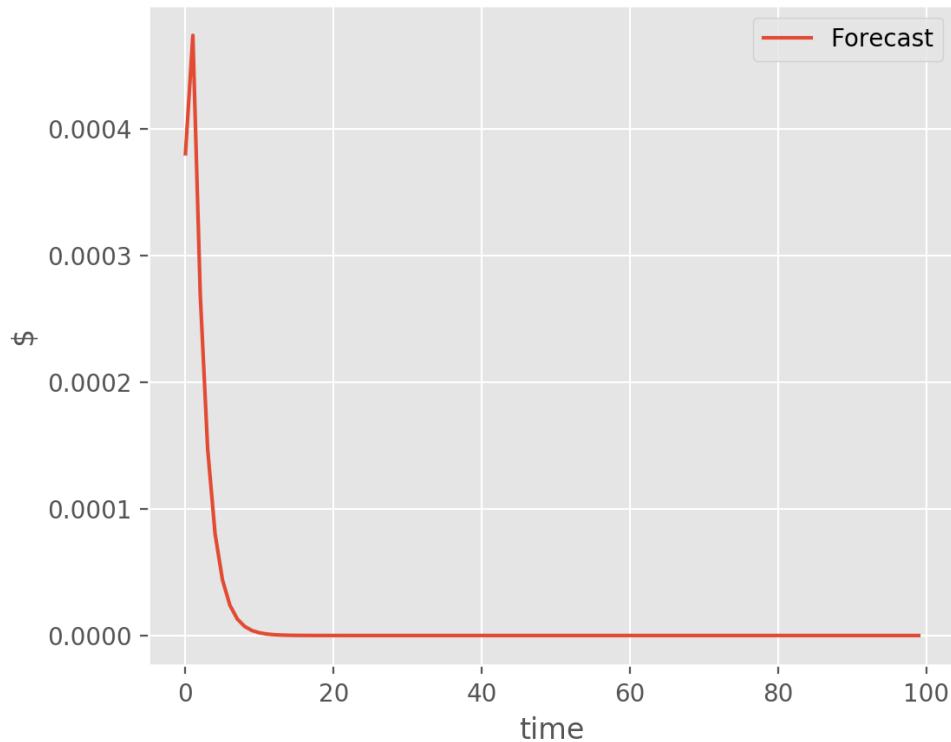
100 Day MSFT ARMA(2, 1) Out-of-Sample Return Forecast.png

FIGURE 4.65: 100 day MSFT ARMA in-sample returns forecast

MSFT scored sharpe ratios of -0.128 for the original returns, and -0.977 for the predicted returns in the in-sample test.

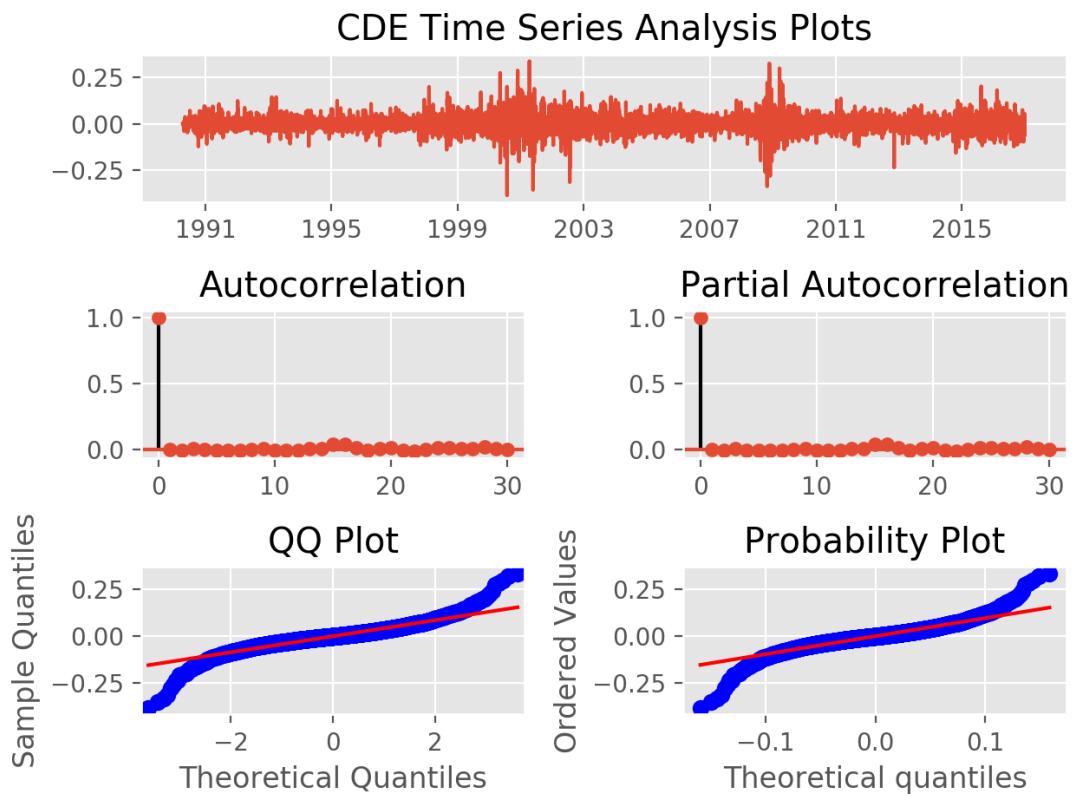


FIGURE 4.66: CDE ARMA time series analysis

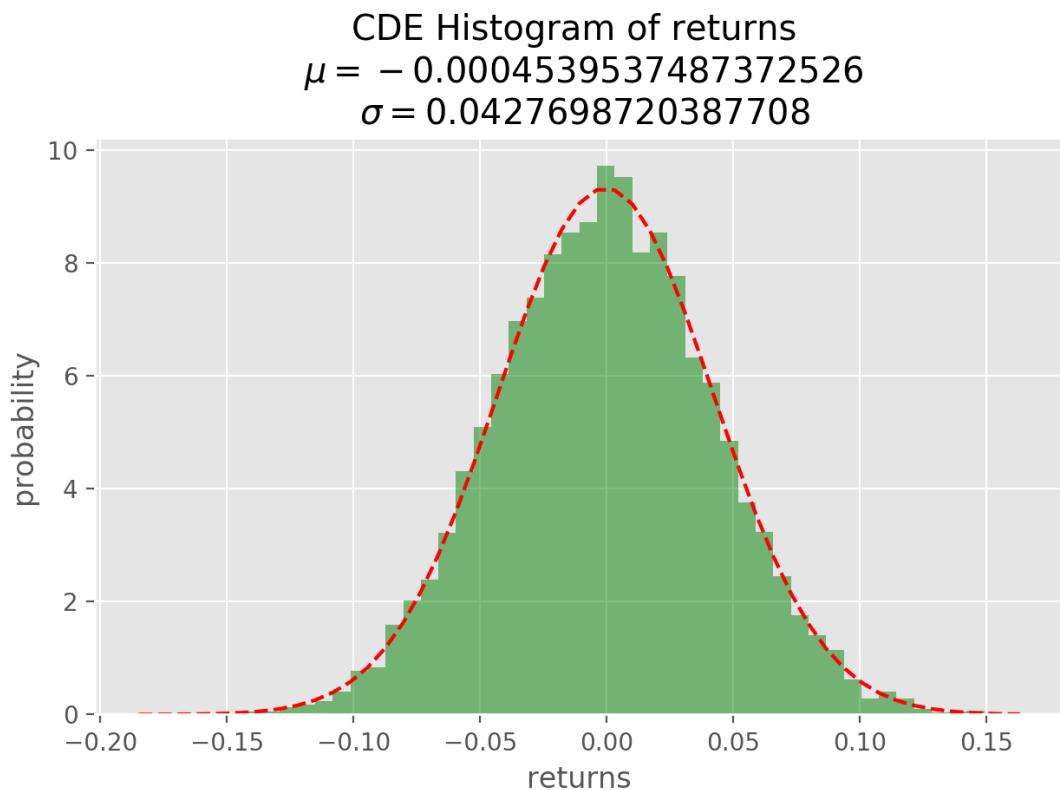


FIGURE 4.67: CDE ARMA histogram of returns

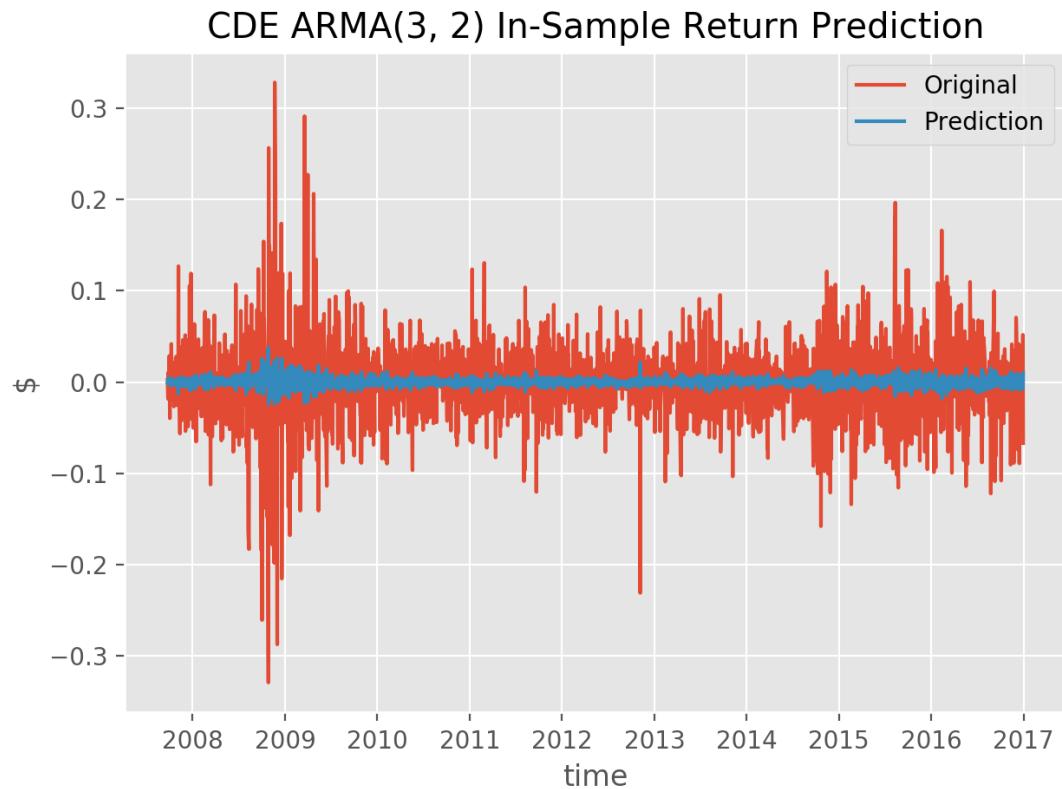


FIGURE 4.68: CDE ARMA in-sample returns prediction

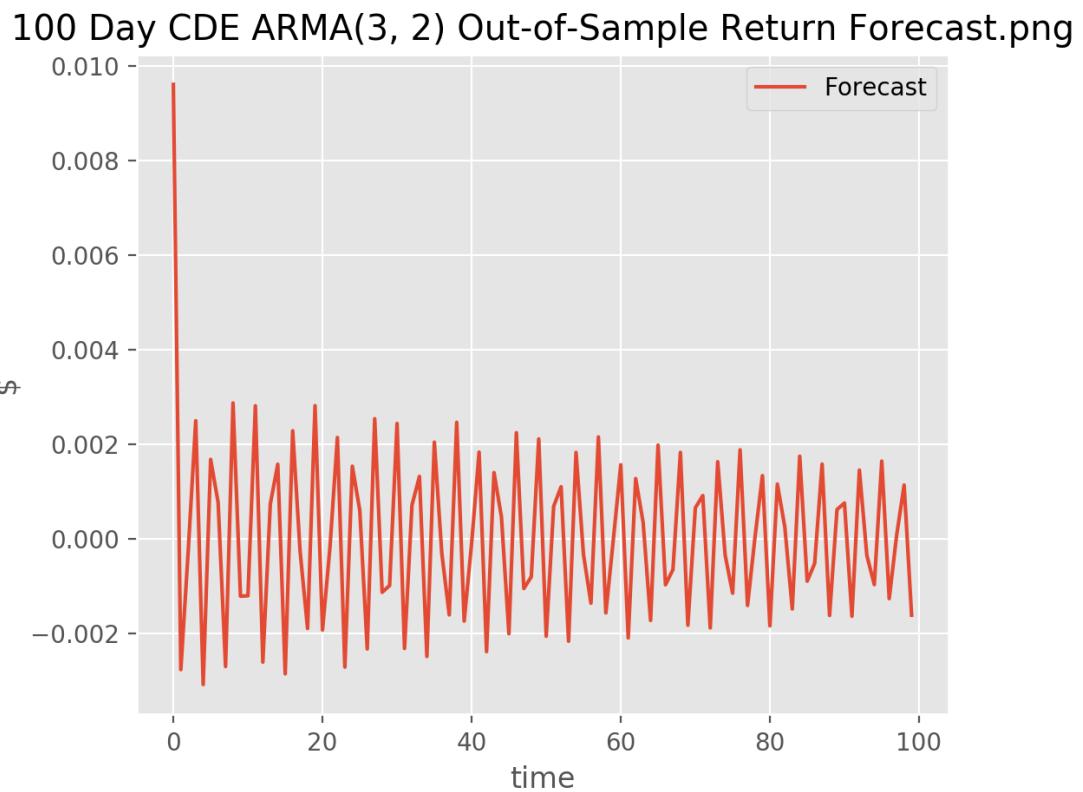


FIGURE 4.69: 100 day CDE ARMA in-sample returns forecast

CDE scored sharpe ratios of -0.245 for the original returns, and -0.614 for the predicted returns in the in-sample test.

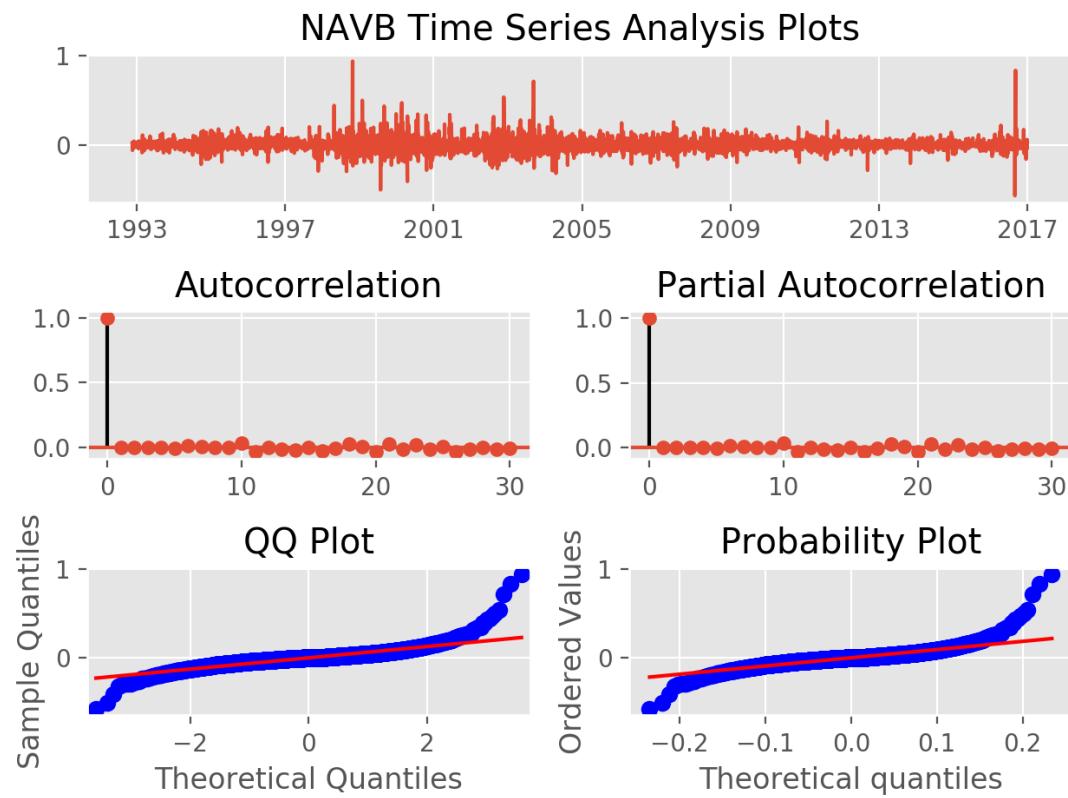


FIGURE 4.70: NAVB ARMA time series analysis

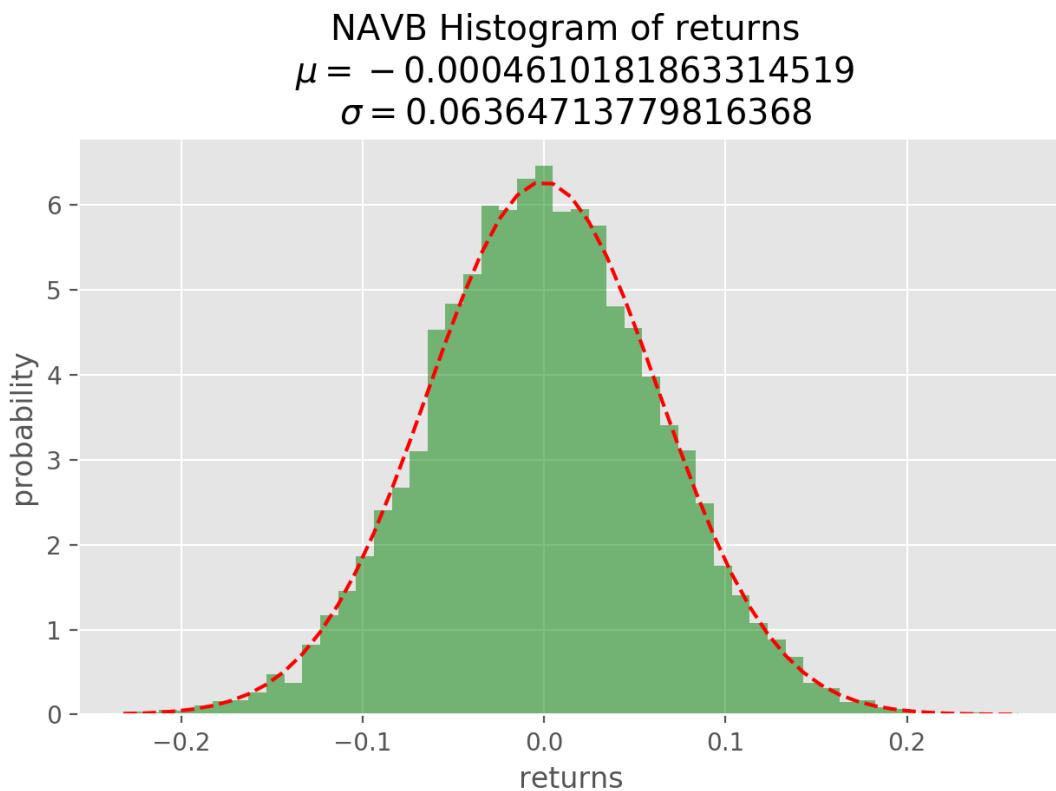


FIGURE 4.71: NAVB ARMA histogram of returns

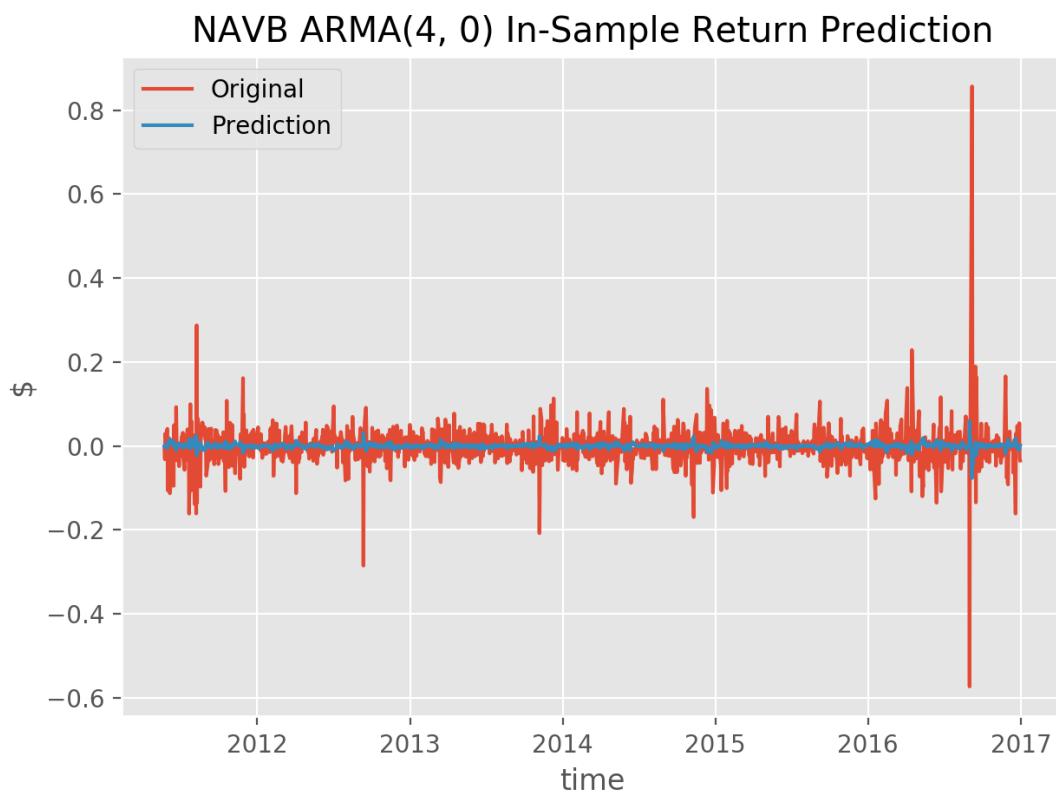


FIGURE 4.72: NAVB ARMA in-sample returns prediction

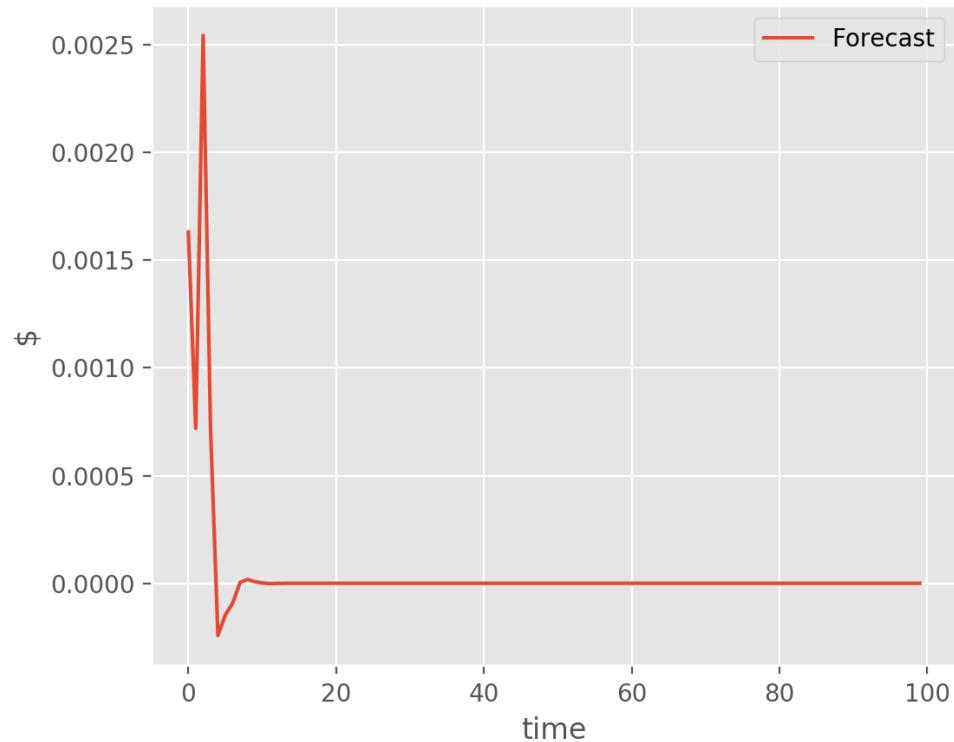
100 Day NAVB ARMA(4, 0) Out-of-Sample Return Forecast.png

FIGURE 4.73: 100 day NAVB ARMA in-sample returns forecast

NAVB scored sharpe ratios of -0.032 for the original returns, and -0.641 for the predicted returns in the in-sample test.

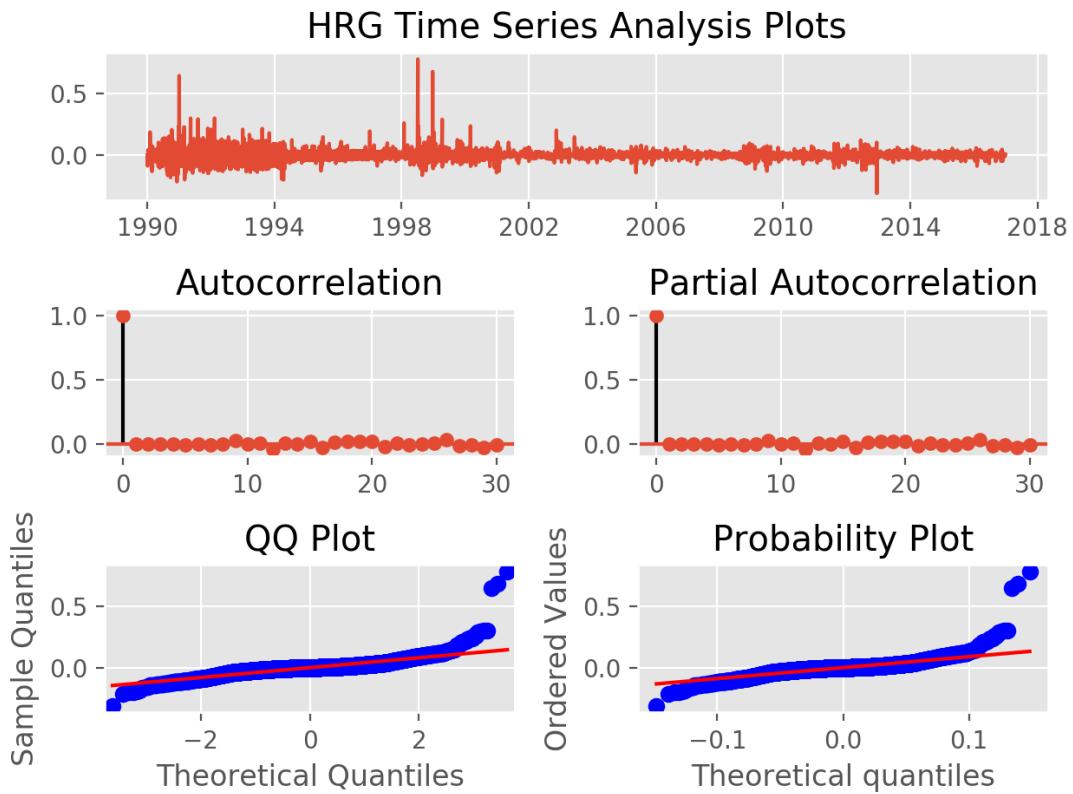


FIGURE 4.74: HRG ARMA time series analysis

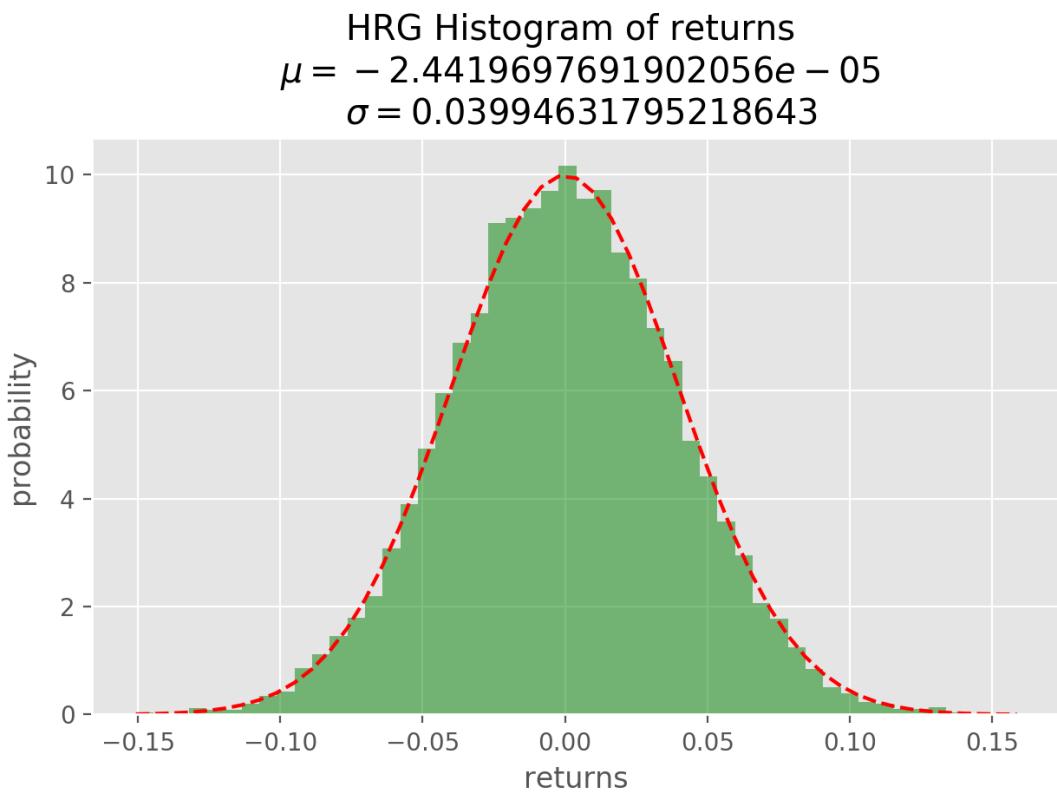


FIGURE 4.75: HRG ARMA histogram of returns

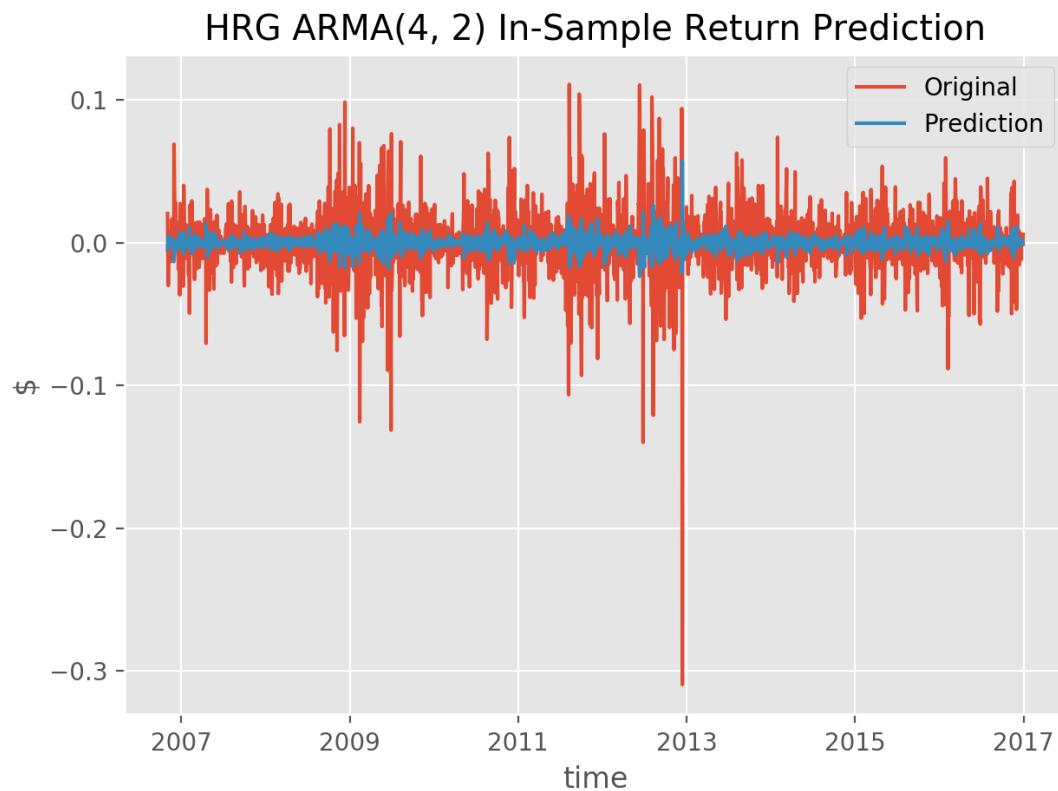


FIGURE 4.76: HRG ARMA in-sample returns prediction

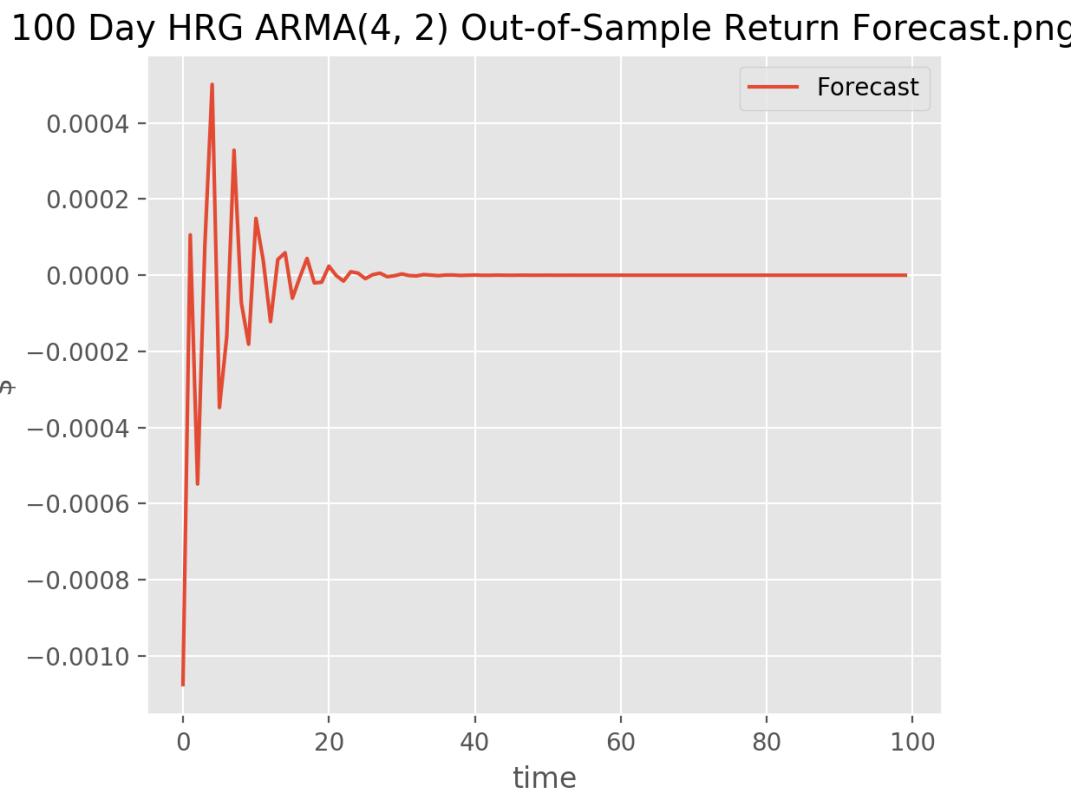


FIGURE 4.77: 100 day HRG ARMA in-sample returns forecast

HL scored sharpe ratios of 0.111 for the original returns, and -1.023 for the predicted returns in the in-sample test.

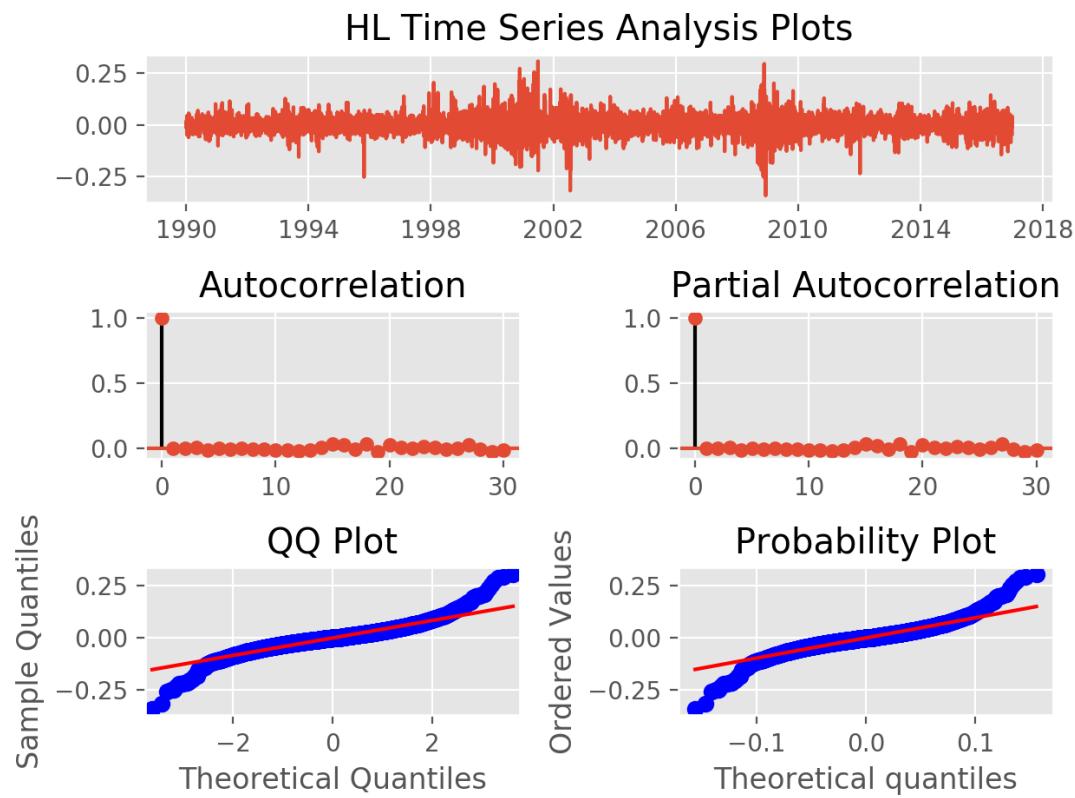


FIGURE 4.78: HL ARMA time series analysis

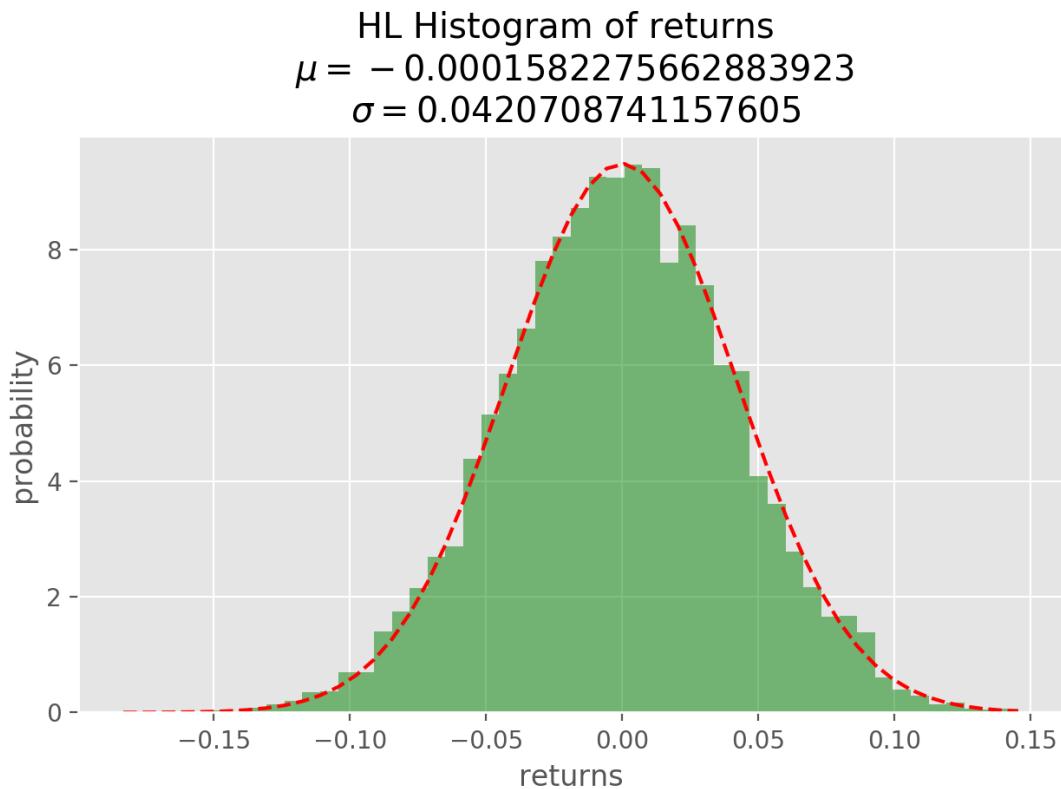


FIGURE 4.79: HL ARMA histogram of returns

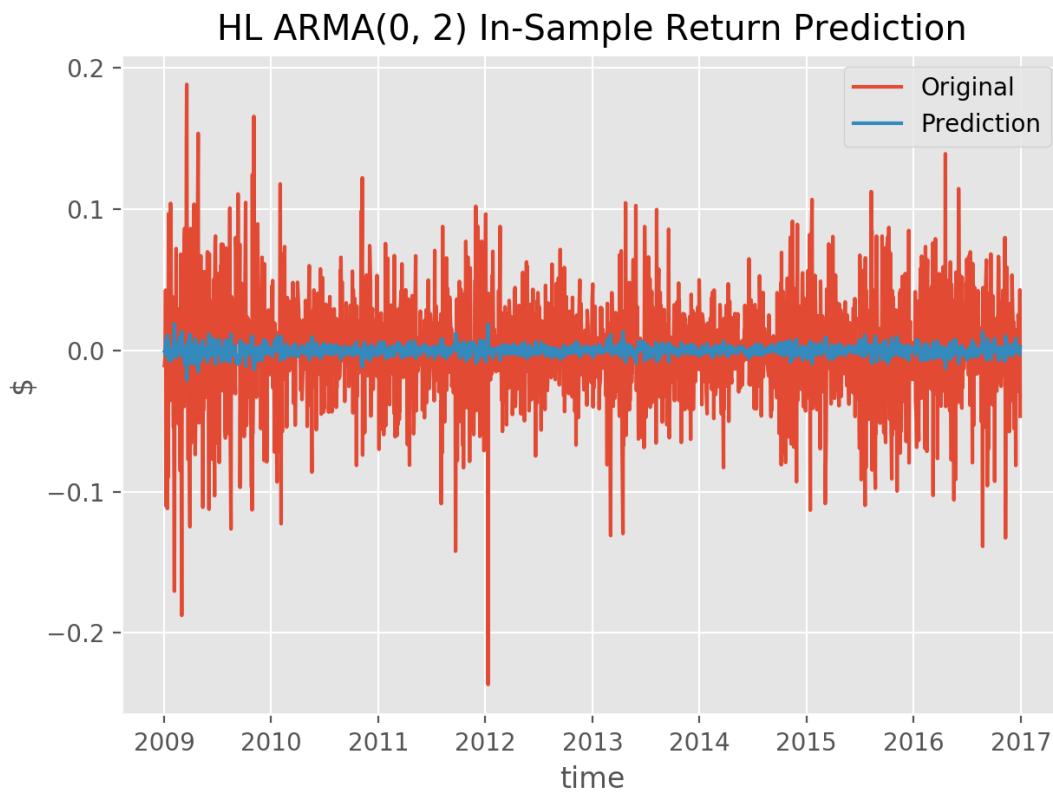


FIGURE 4.80: HL ARMA in-sample returns prediction

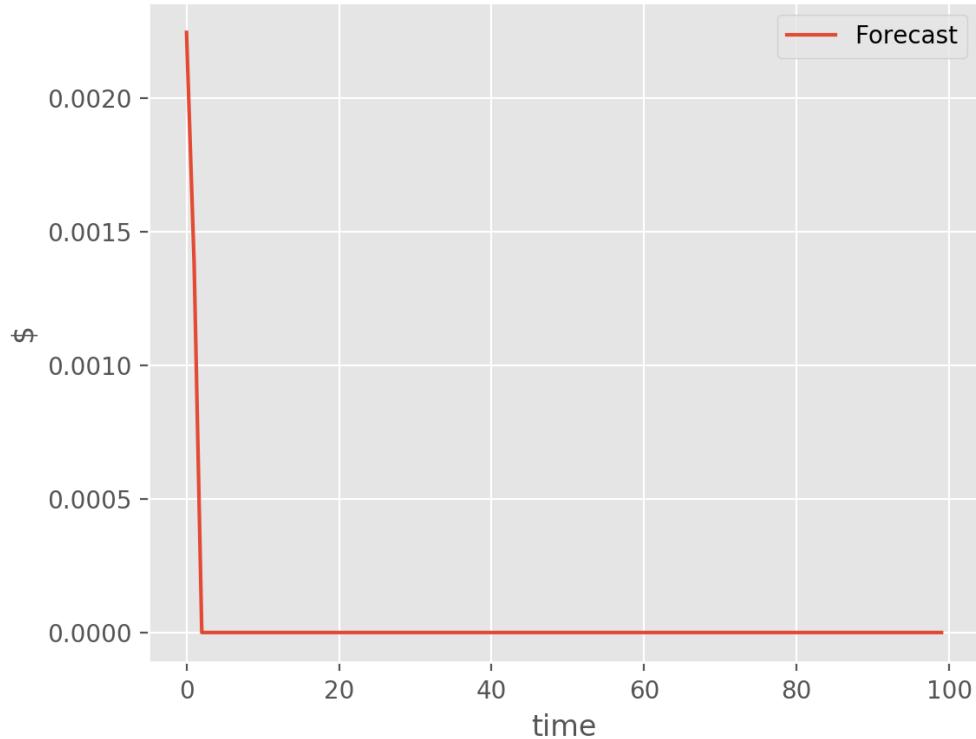
100 Day HL ARMA(0, 2) Out-of-Sample Return Forecast.png

FIGURE 4.81: 100 day HL ARMA in-sample returns forecast

4.1.6 Auto Regressive Integrated Moving Average (ARIMA)

MSFT scored sharpe ratios of 0.123 for the original returns, and -4.904 for the predicted returns in the in-sample test.

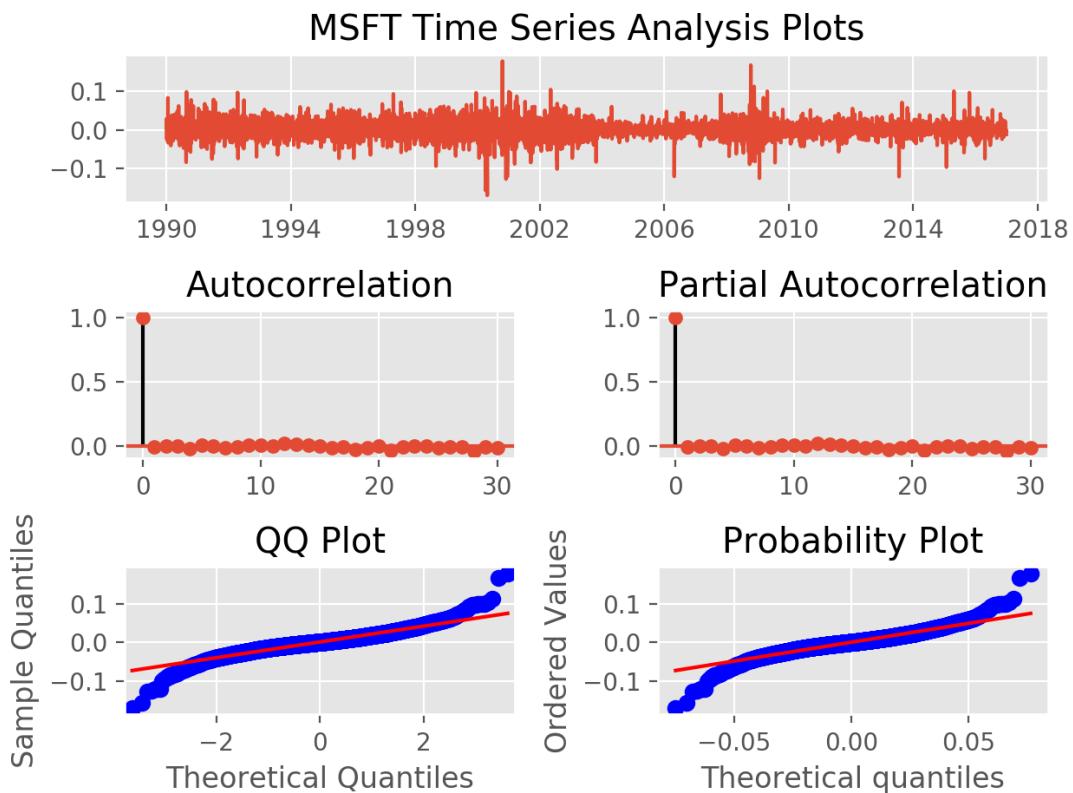


FIGURE 4.82: MSFT ARIMA time series analysis

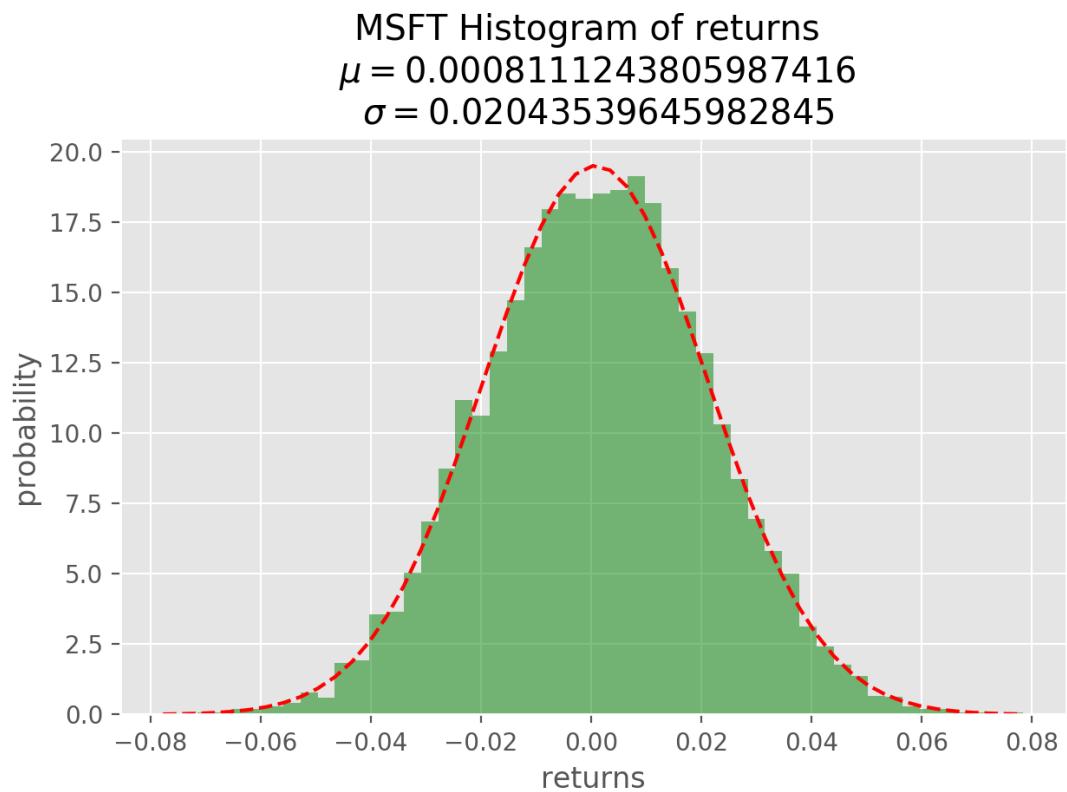


FIGURE 4.83: MSFT ARIMA histogram of returns

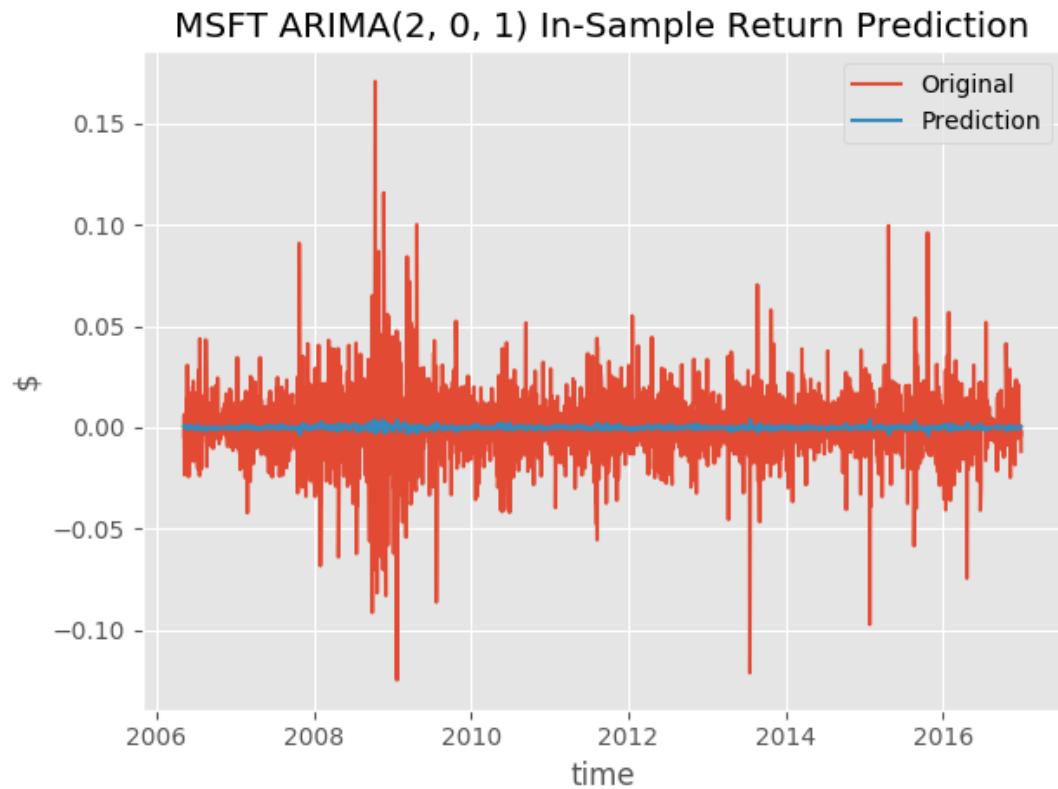


FIGURE 4.84: MSFT ARIMA in-sample returns prediction

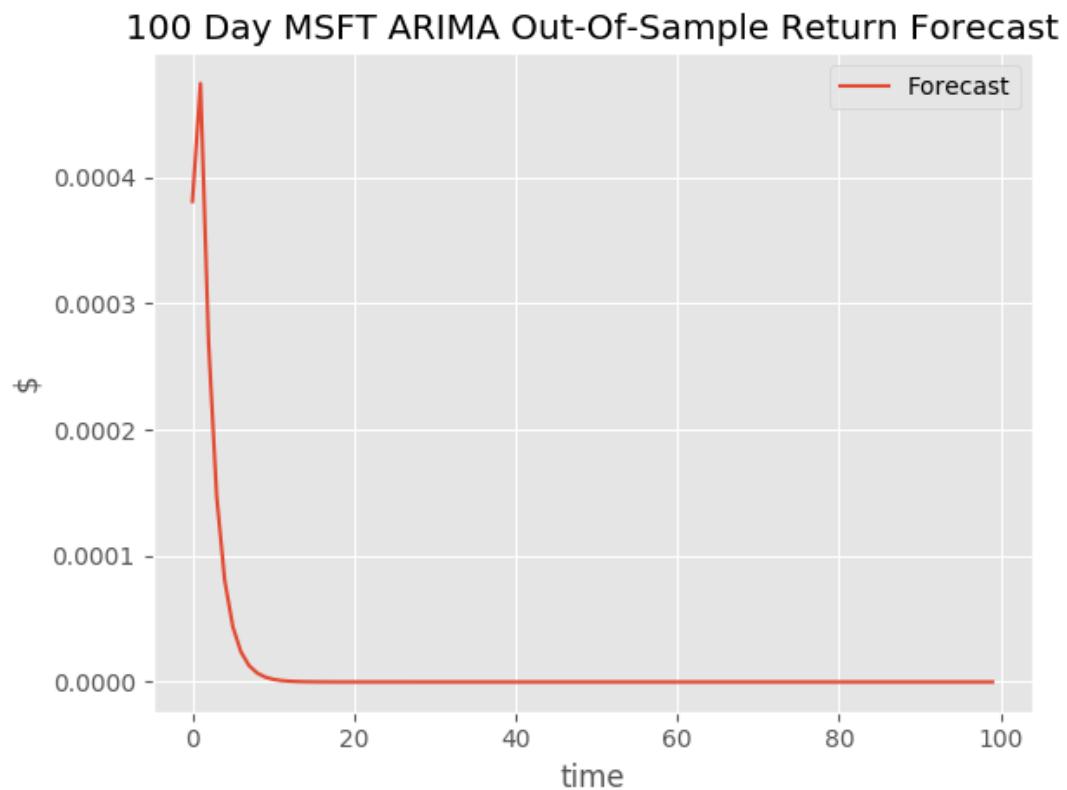


FIGURE 4.85: 100 day MSFT ARIMA in-sample returns forecast

CDE scored sharpe ratios of -0.133 for the original returns, and -0.698 for the predicted returns in the in-sample test.

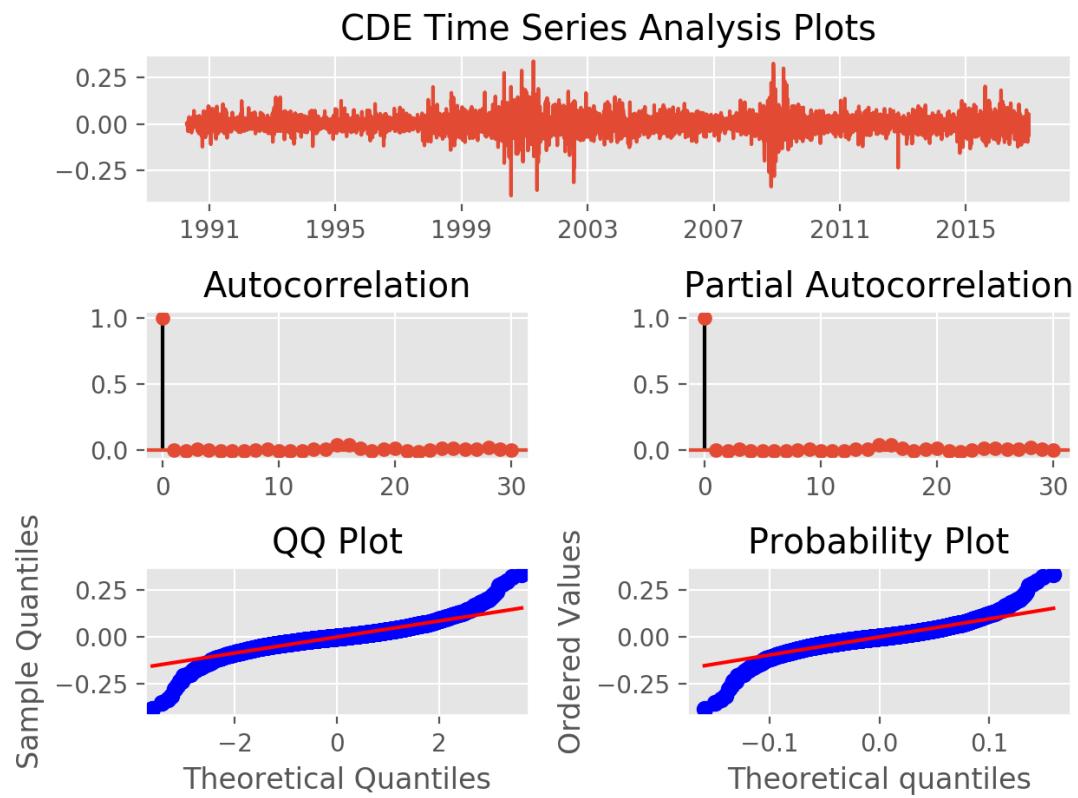


FIGURE 4.86: CDE ARIMA time series analysis

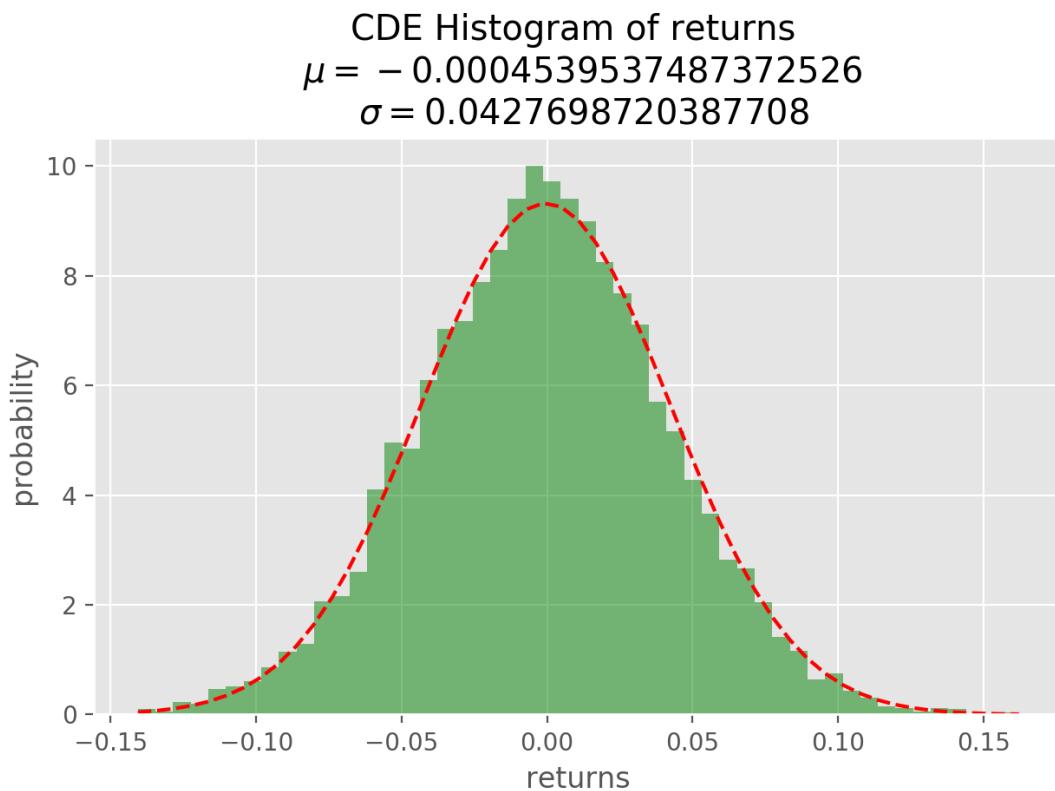


FIGURE 4.87: CDE ARIMA histogram of returns

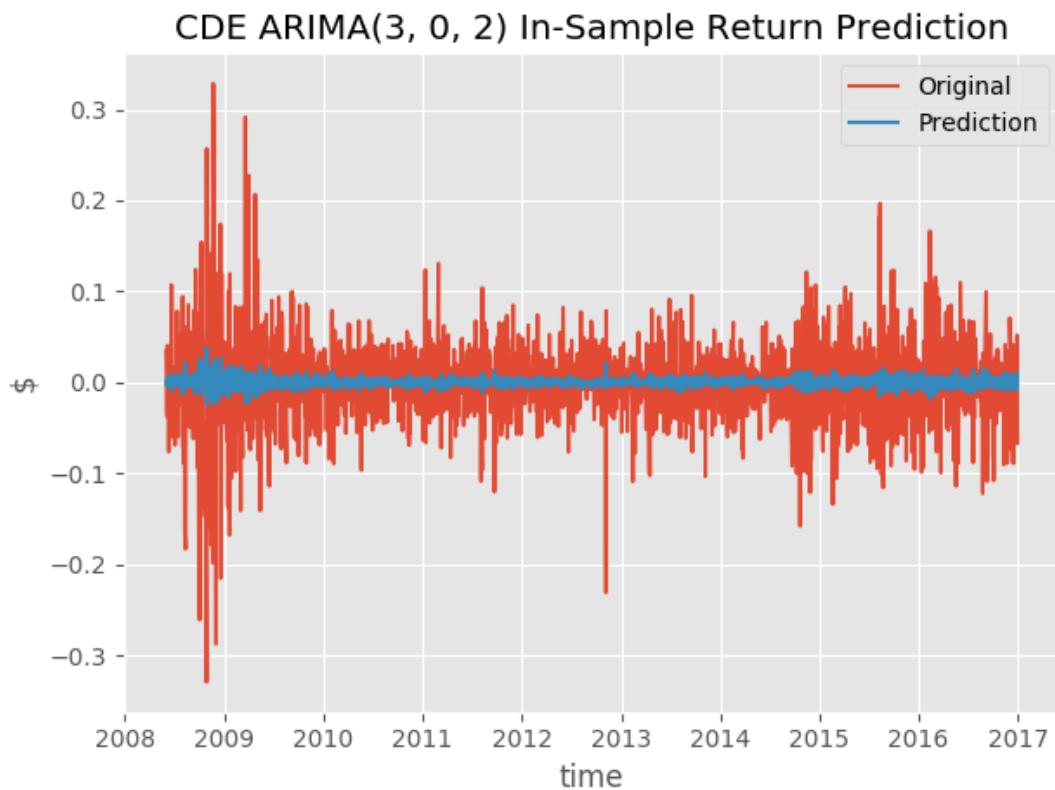


FIGURE 4.88: CDE ARIMA in-sample returns prediction

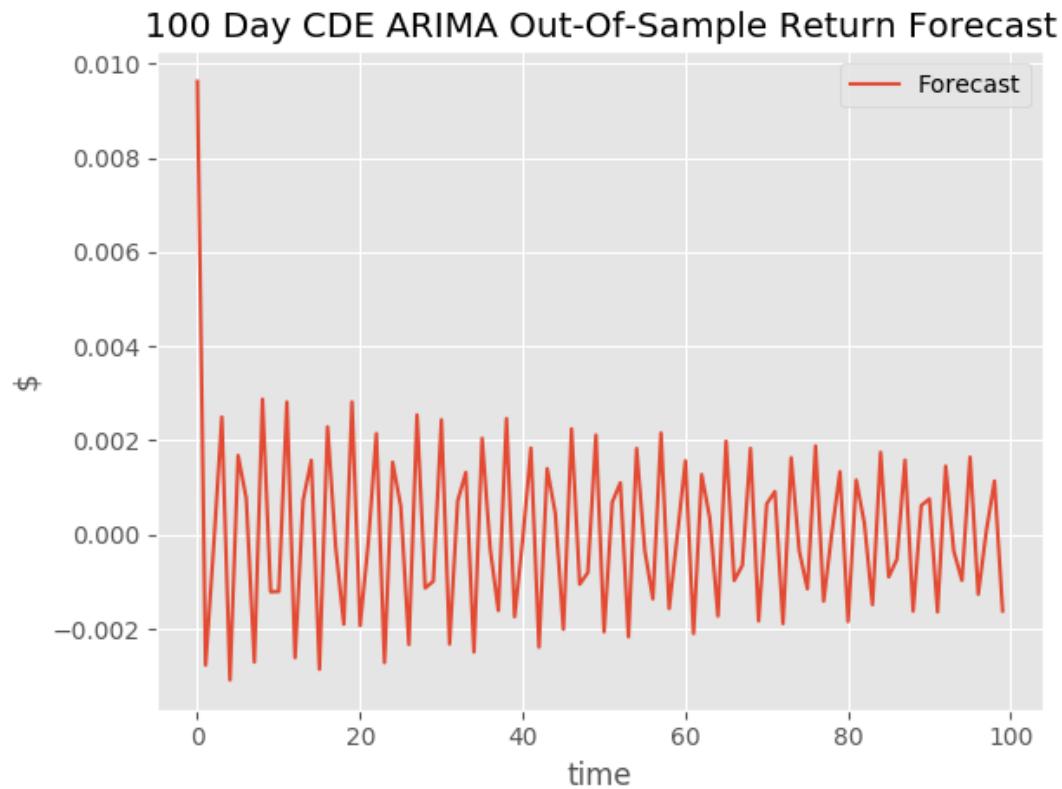


FIGURE 4.89: 100 day CDE ARIMA in-sample returns forecast

NAV B scored sharpe ratios of 0.015 for the original returns, and -0.708 for the predicted returns in the in-sample test.

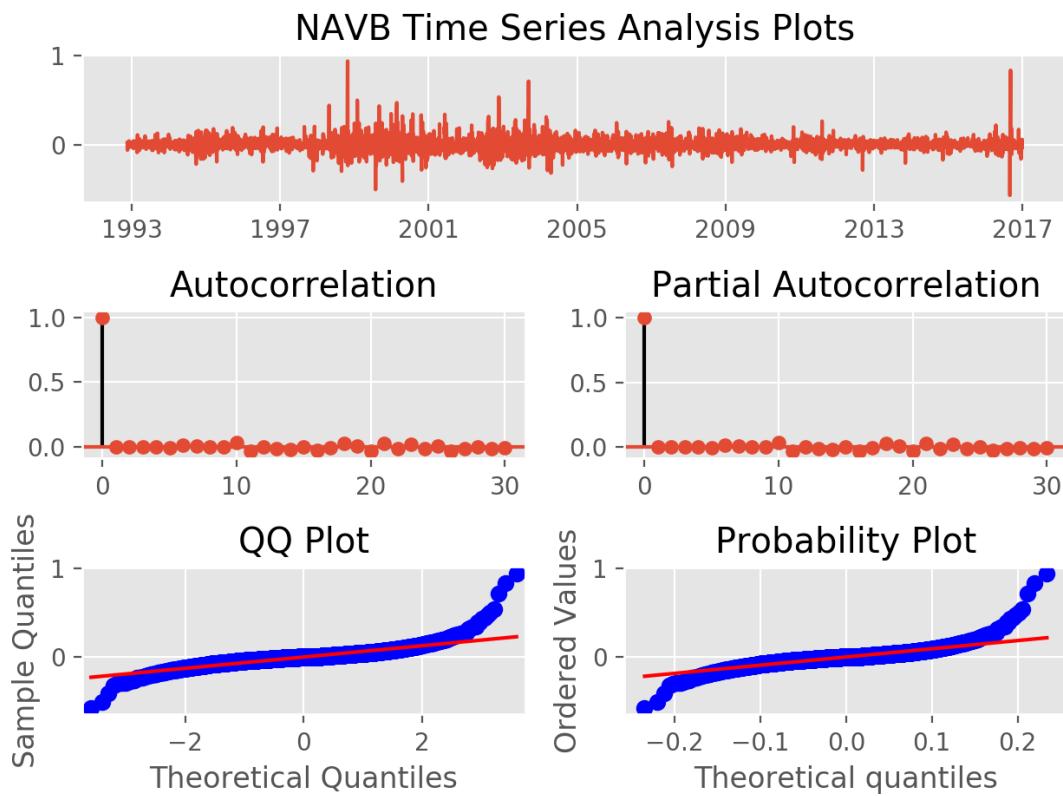


FIGURE 4.90: NAVB ARIMA time series analysis

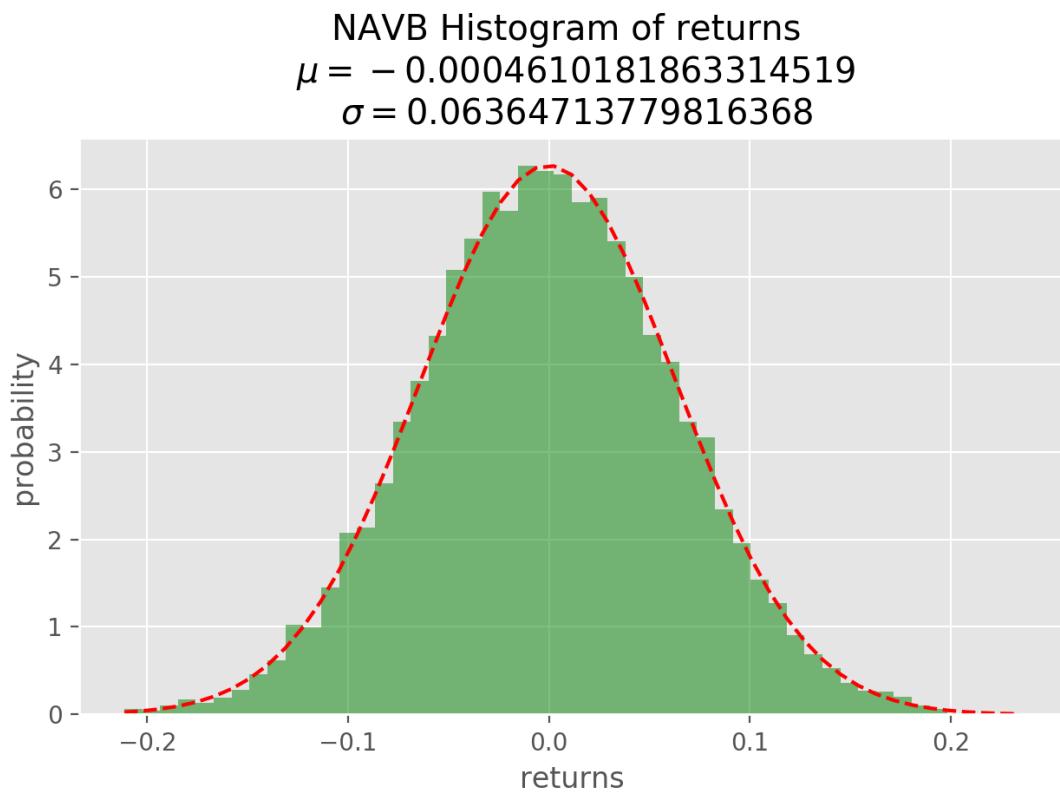


FIGURE 4.91: NAVB ARIMA histogram of returns

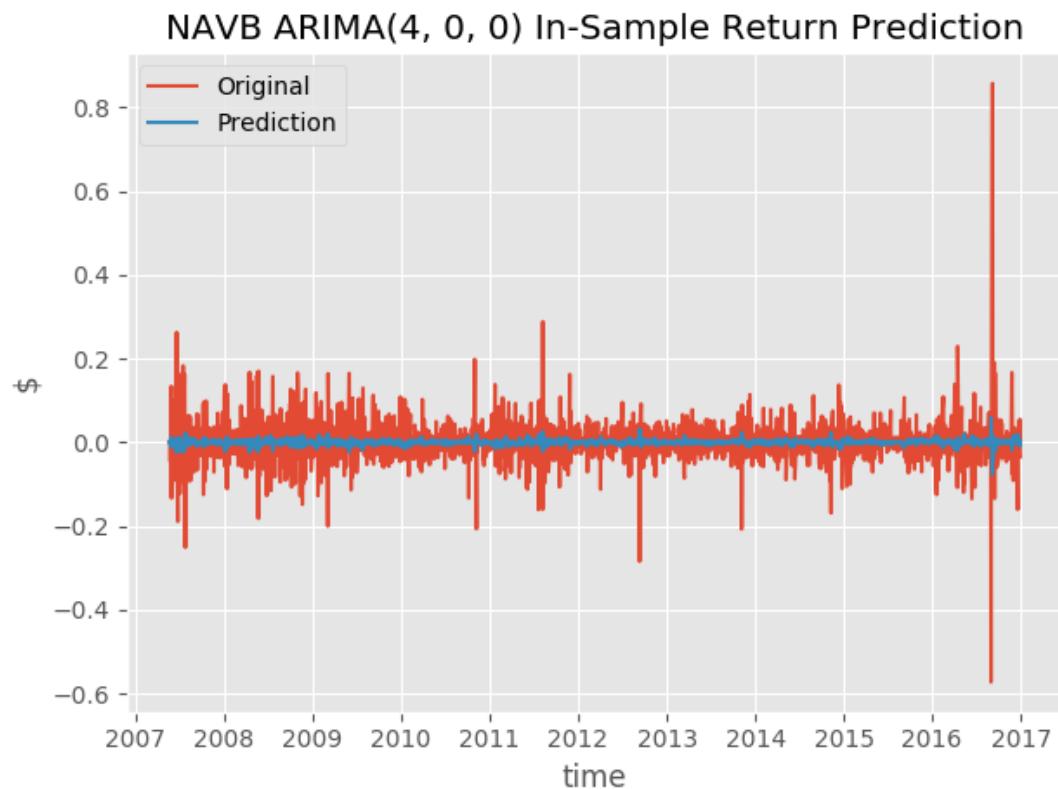


FIGURE 4.92: NAVB ARIMA in-sample returns prediction

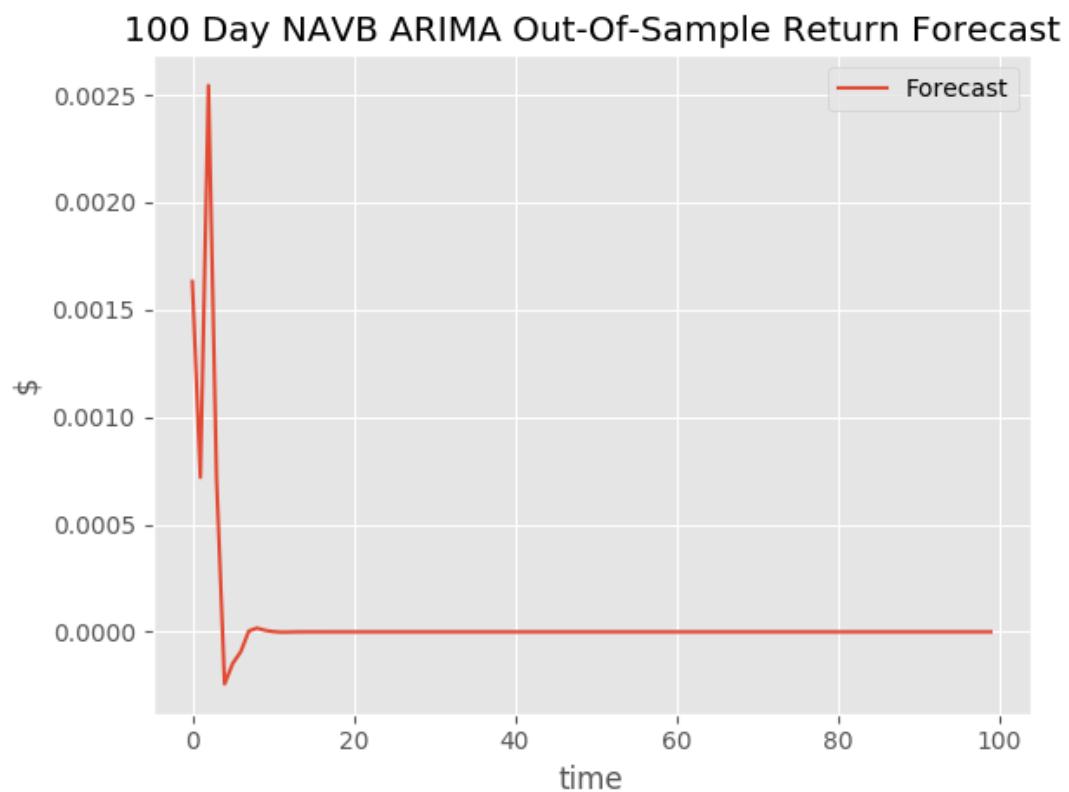


FIGURE 4.93: 100 day NAVB ARIMA in-sample returns forecast

HRG scored sharpe ratios of 0.401 for the original returns, and -1.354 for the predicted returns in the in-sample test.

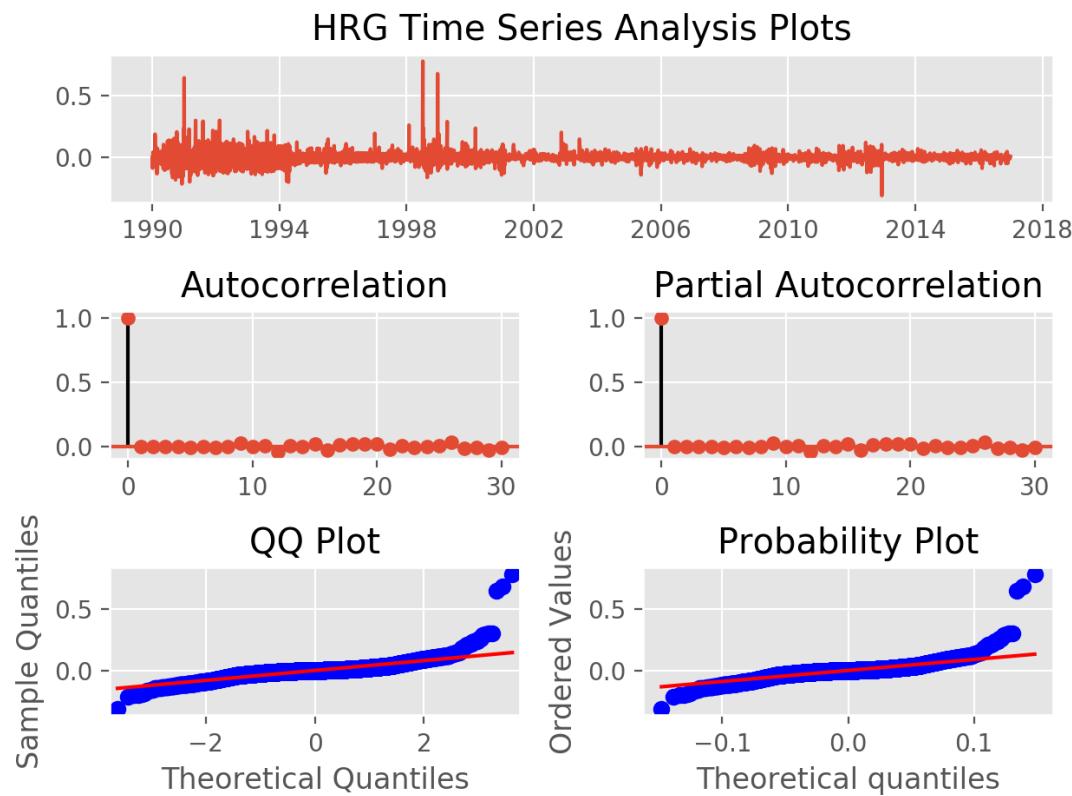


FIGURE 4.94: HRG ARIMA time series analysis

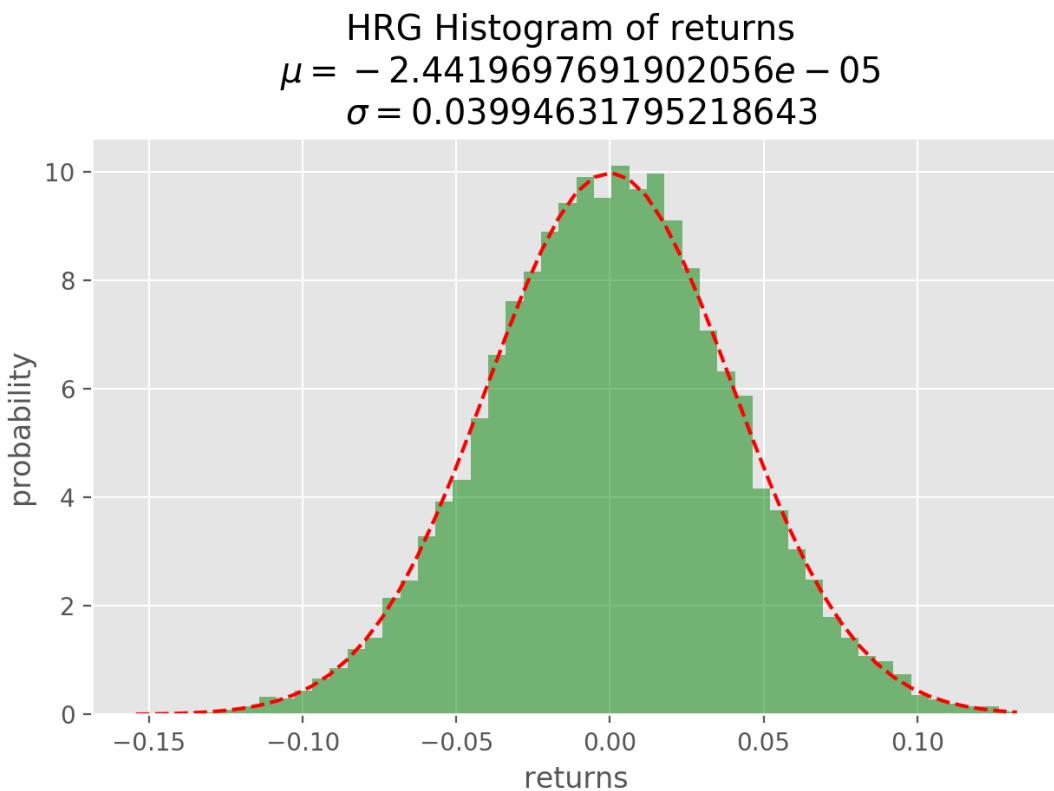


FIGURE 4.95: HRG ARIMA histogram of returns

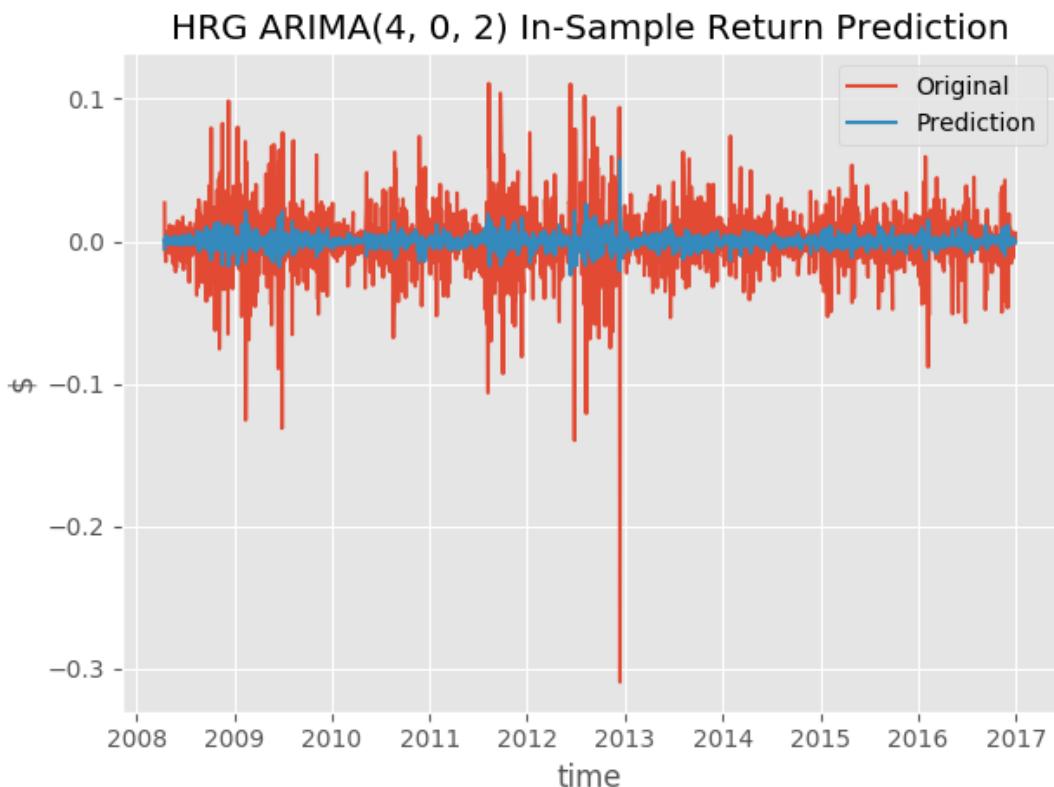


FIGURE 4.96: HRG ARIMA in-sample returns prediction

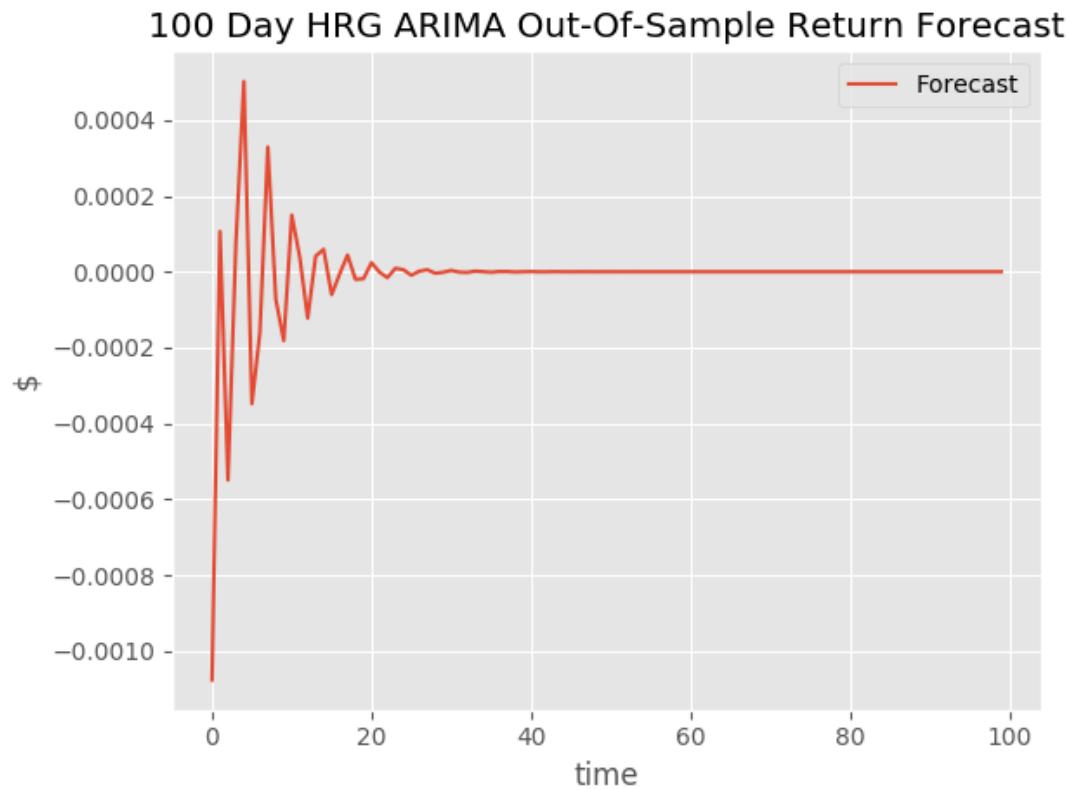


FIGURE 4.97: 100 day HRG ARIMA in-sample returns forecast

HL scored sharpe ratios of -0.127 for the original returns, and -0.974 for the predicted returns in the in-sample test.

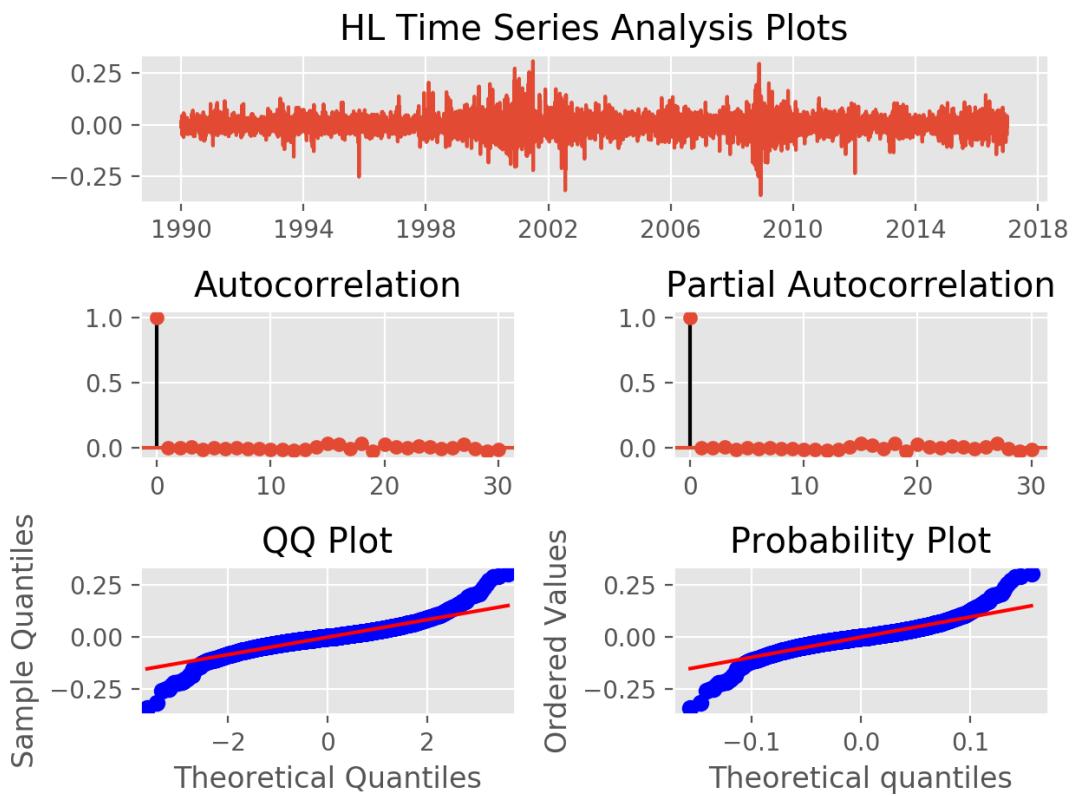


FIGURE 4.98: HL ARIMA time series analysis

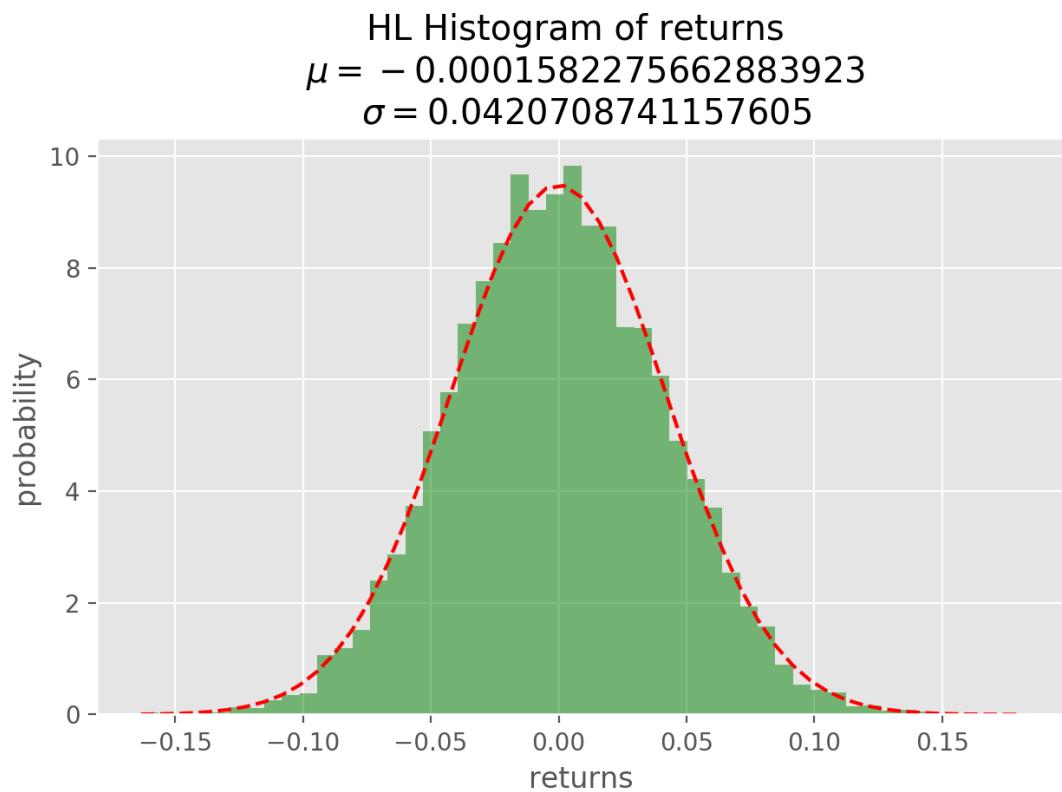


FIGURE 4.99: HL ARIMA histogram of returns

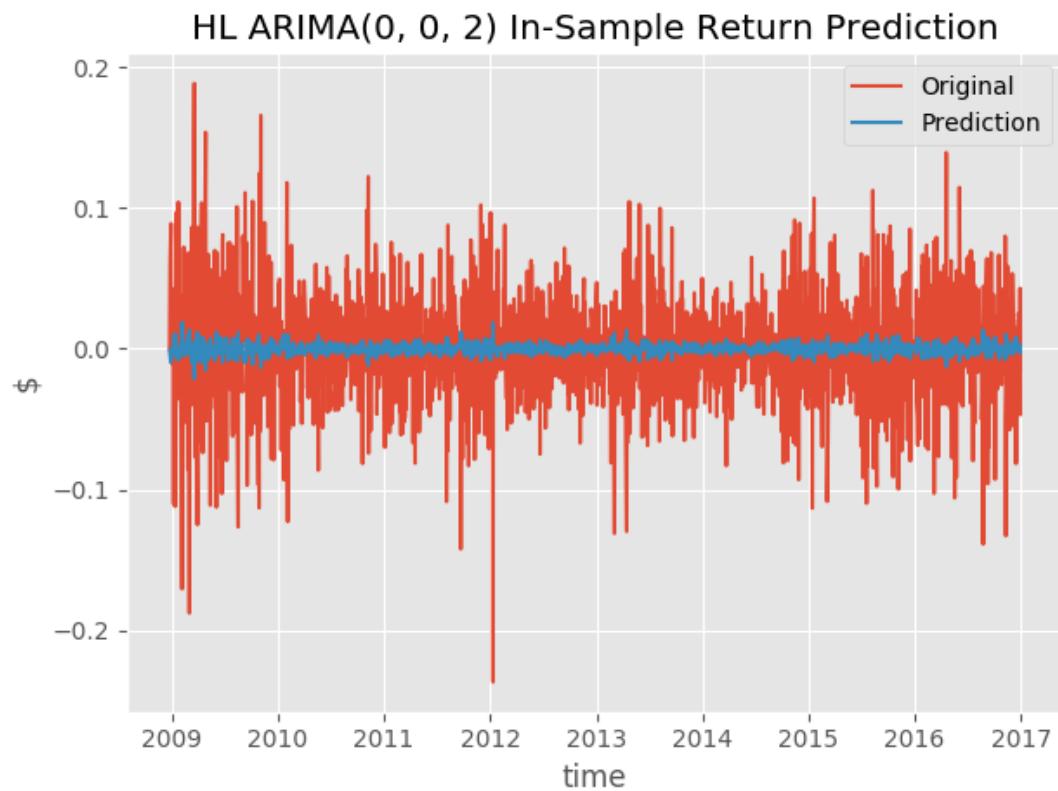


FIGURE 4.100: HL ARIMA in-sample returns prediction

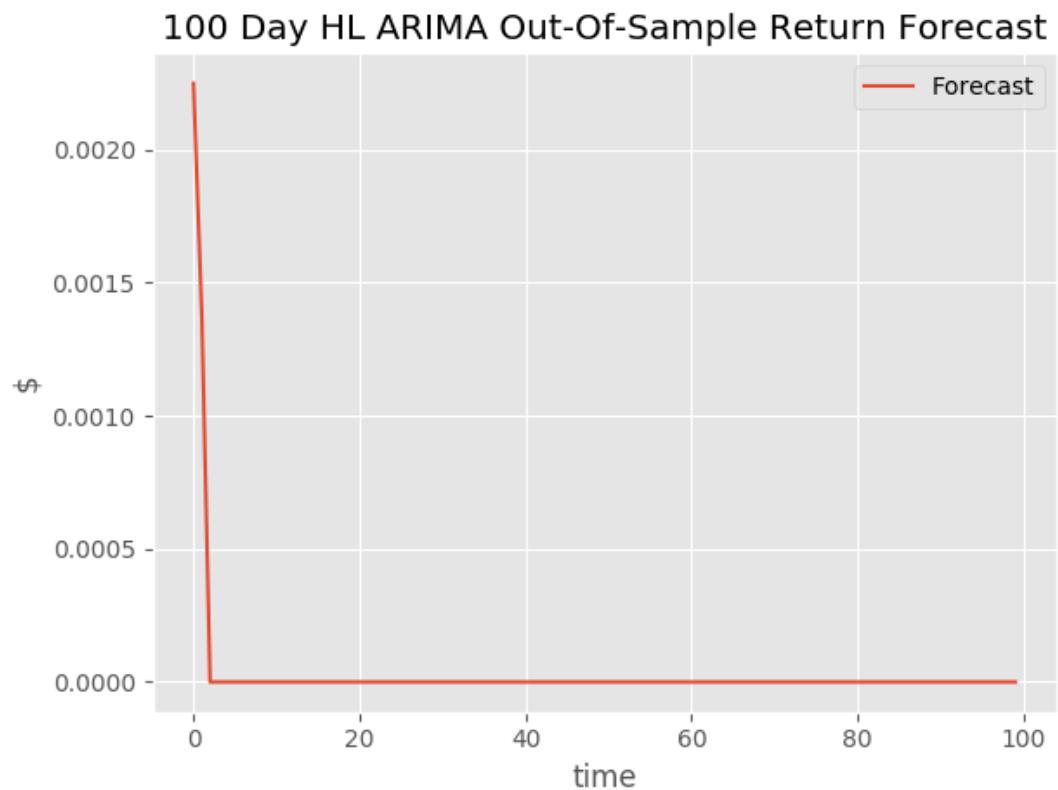


FIGURE 4.101: 100 day HL ARIMA in-sample returns forecast

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
NAV	0.76	592	165	331	126
HRG	0.75	553	210	462	135
HL	0.79	603	135	475	147

TABLE 4.2: Decision Tree results

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
NAV	0.76	588	164	332	130
HRG	0.75	542	191	481	146
HL	0.79	602	132	478	149

TABLE 4.3: Boosted Decision Tree results

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
NAV	0.76	593	169	327	125
HRG	0.75	542	191	481	146
HL	0.81	579	98	512	171

TABLE 4.4: Support Vector Machine results

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
NAV	0.76	590	164	332	128
HRG	0.75	554	210	462	134
HL	0.81	581	101	509	169

TABLE 4.5: Random Forest results

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
NAV	0.75	573	161	335	145
HRG	0.74	560	233	439	128
HL	0.81	581	101	509	169

TABLE 4.6: K-Nearest Neighbour results

4.2.1.6 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
NAV	0.76	588	164	332	130
HRG	0.75	542	191	481	146
HL	0.79	601	132	478	149

TABLE 4.7: Logistic Regression results

4.2.1.7 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
NAV	0.74	553	153	343	165
HRG	0.73	491	170	502	197
HL	0.77	590	157	453	160

TABLE 4.8: Gaussian Naive Bayes results

4.2.1.8 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
NAV	0.74	553	153	343	165
HRG	0.73	492	170	502	196
HL	0.77	590	158	452	160

TABLE 4.9: Bernoulli Naive Bayes results

4.2.1.9 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
NAV	0.76	691	183	403	153
HRG	0.74	690	269	542	164
HL	0.77	1059	270	848	164

TABLE 4.10: Neural Network results

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
NAV	0.59	1032	707	118	80
HRG	0.72	761	225	1059	518
HL	0.80	535	83	527	215

TABLE 4.11: Stochastic Gradient Descent results

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.

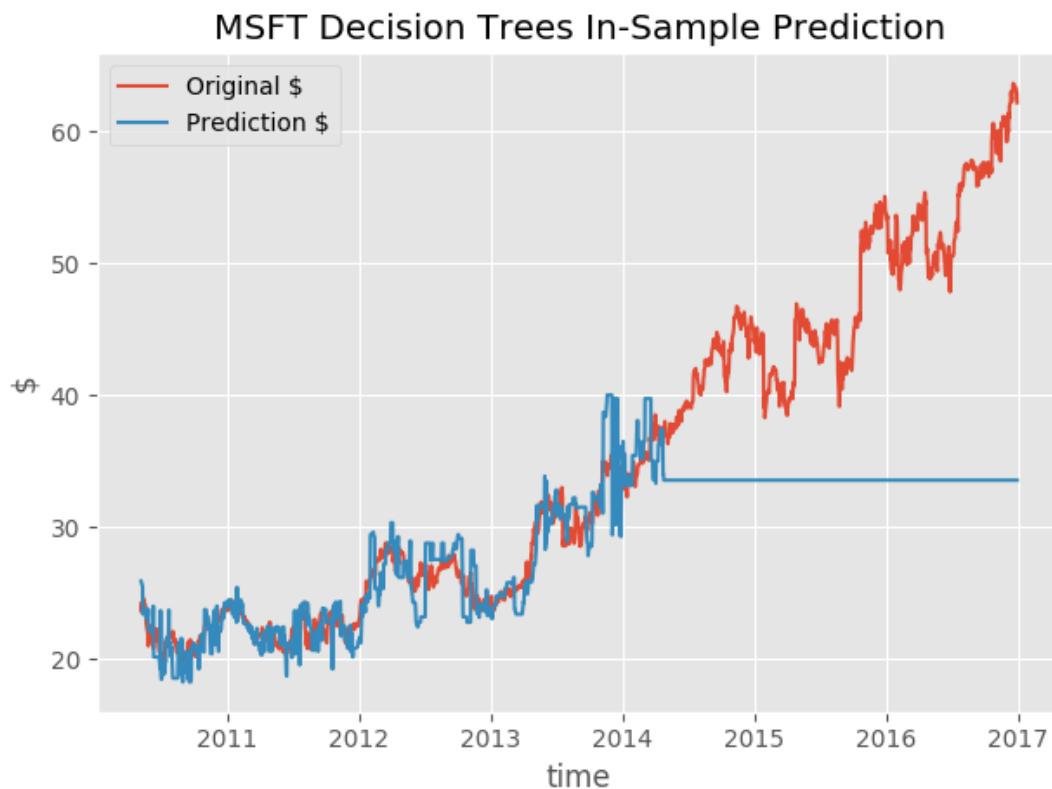


FIGURE 4.102: MSFT Decision Trees in-sample prediction

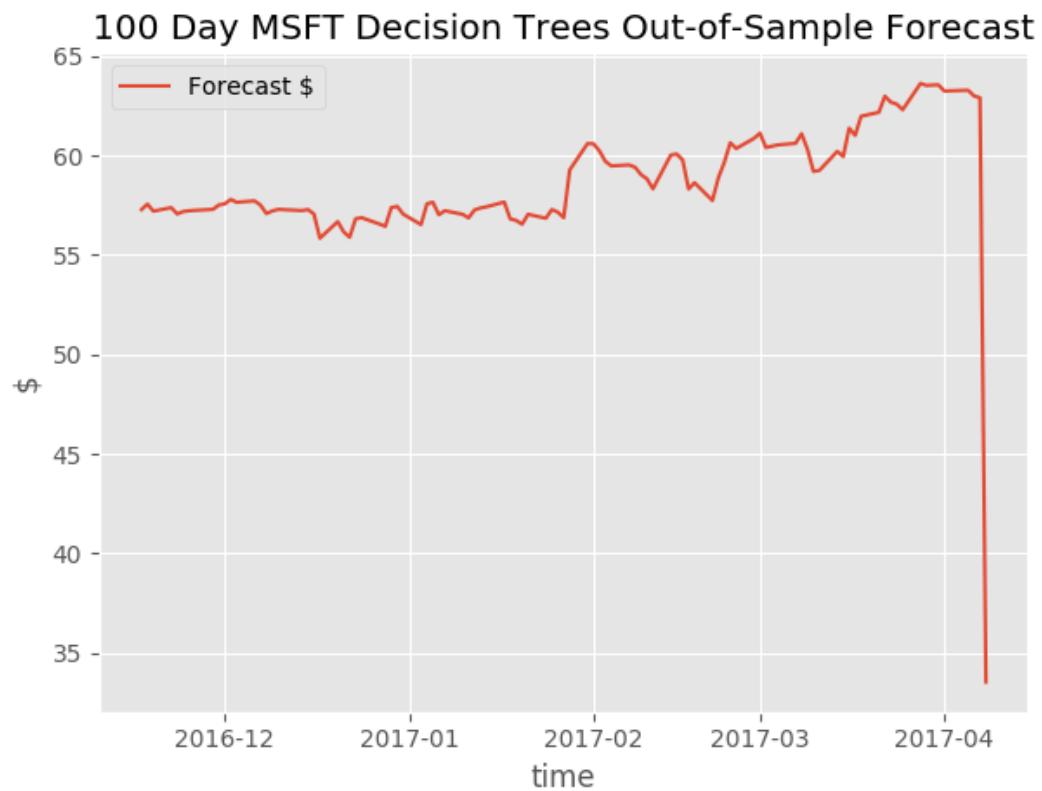


FIGURE 4.103: 100 day MSFT Decision Trees out-of-sample forecast

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

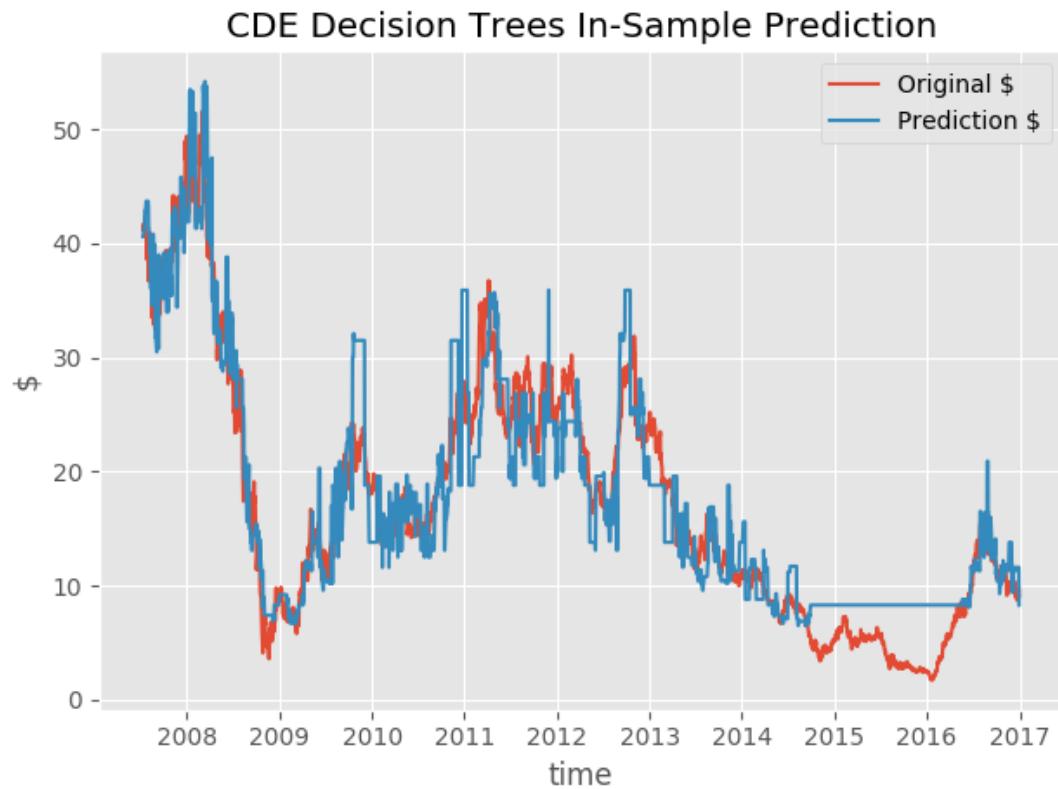


FIGURE 4.104: CDE Decision Trees in-sample prediction

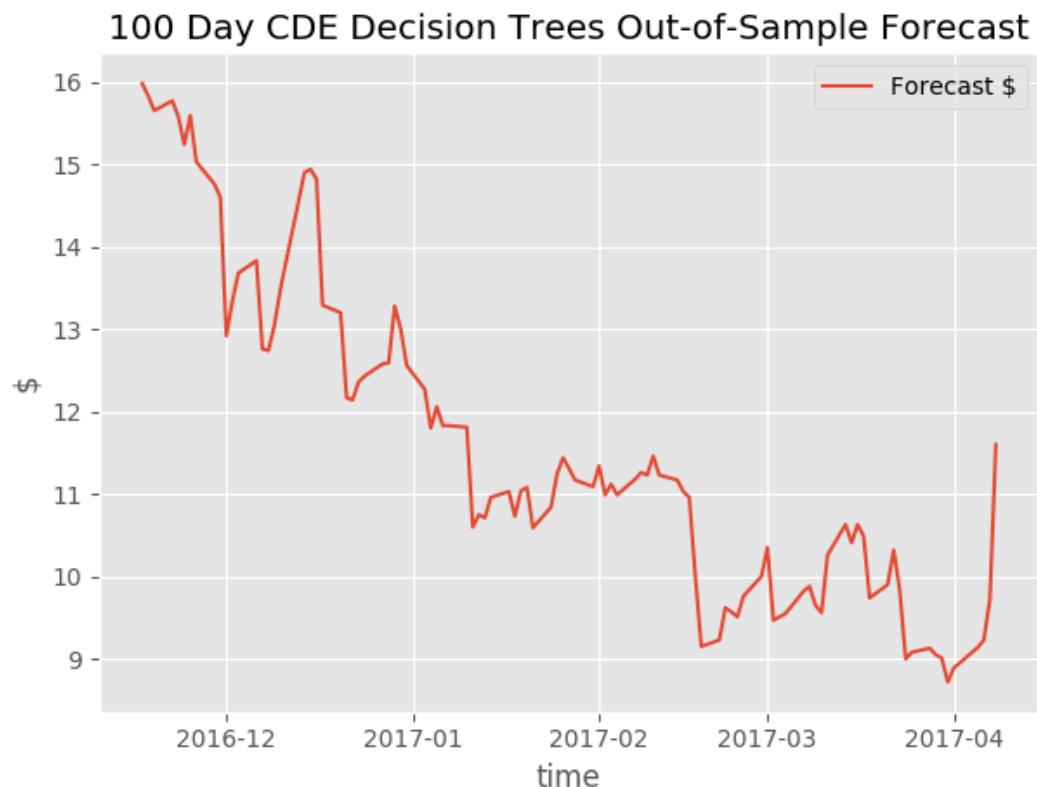


FIGURE 4.105: 100 day CDE Decision Trees out-of-sample forecast

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

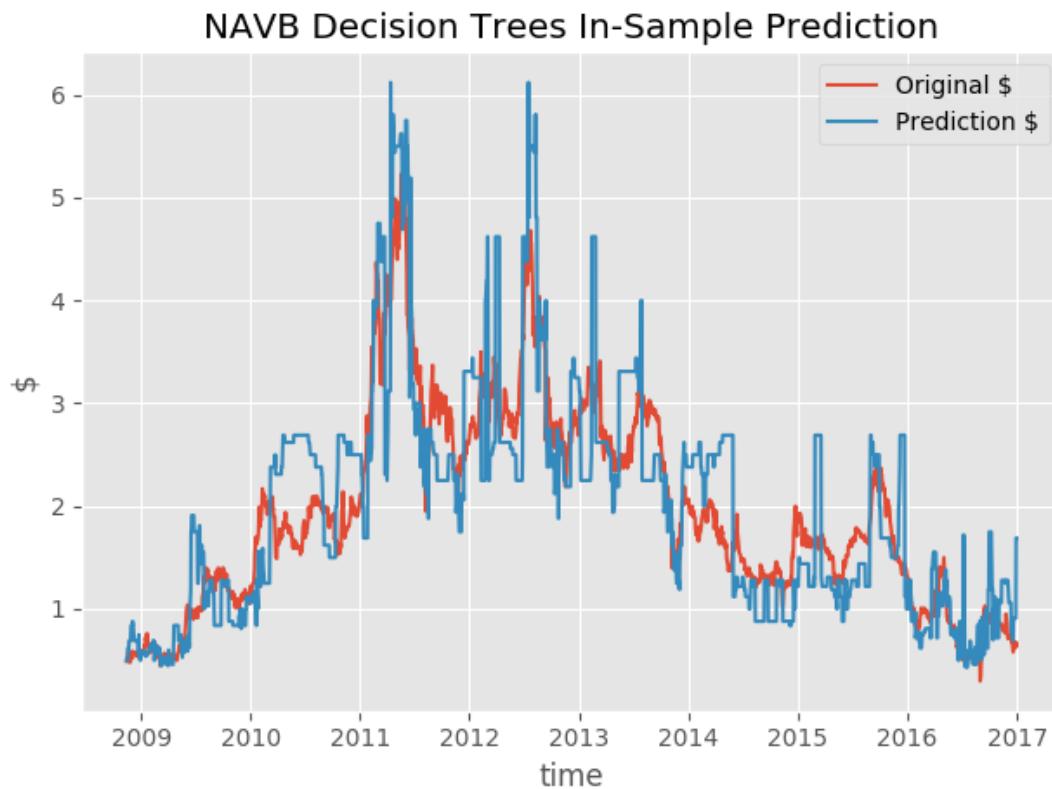


FIGURE 4.106: NAVB Decision Trees in-sample prediction

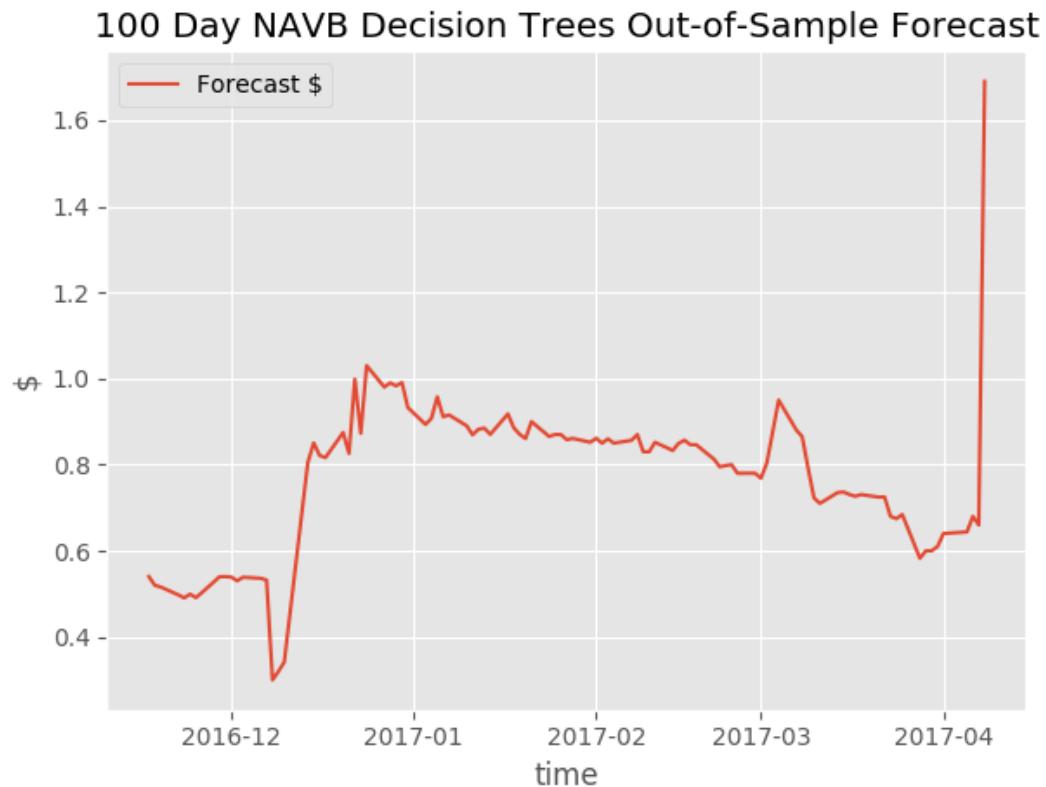


FIGURE 4.107: 100 day NAVB Decision Trees out-of-sample forecast

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

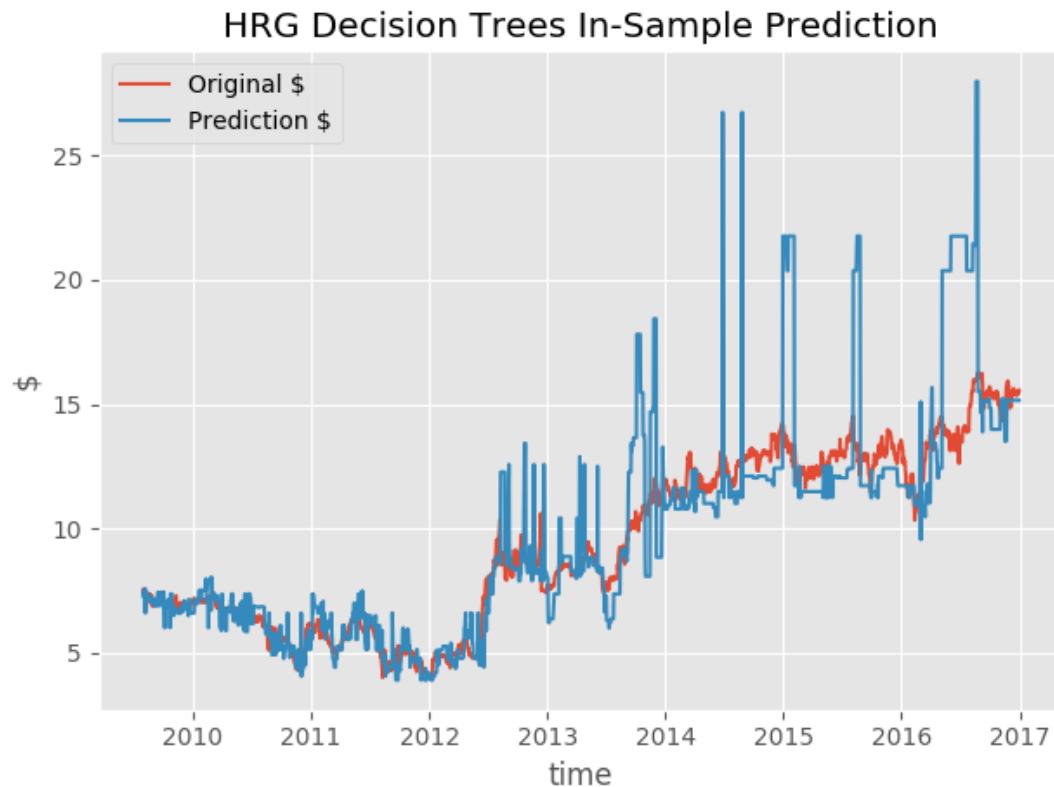


FIGURE 4.108: HRG Decision Trees in-sample prediction

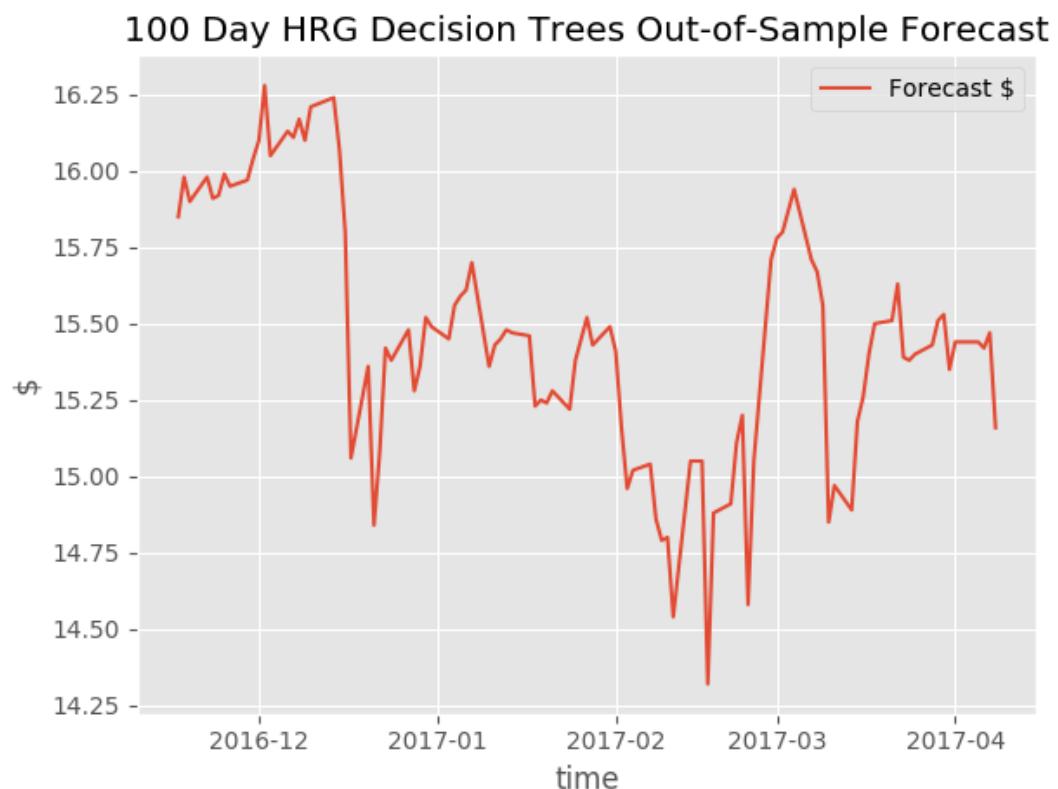


FIGURE 4.109: 100 day HRG Decision Trees out-of-sample forecast

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

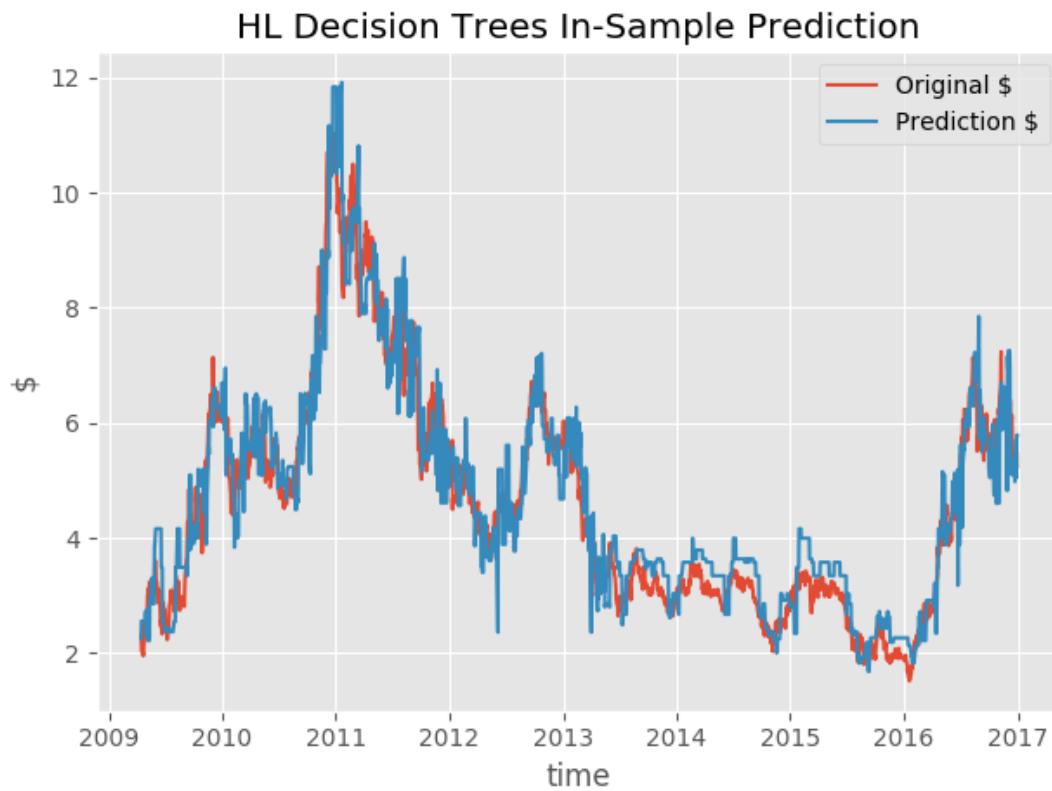


FIGURE 4.110: HL Decision Trees in-sample prediction

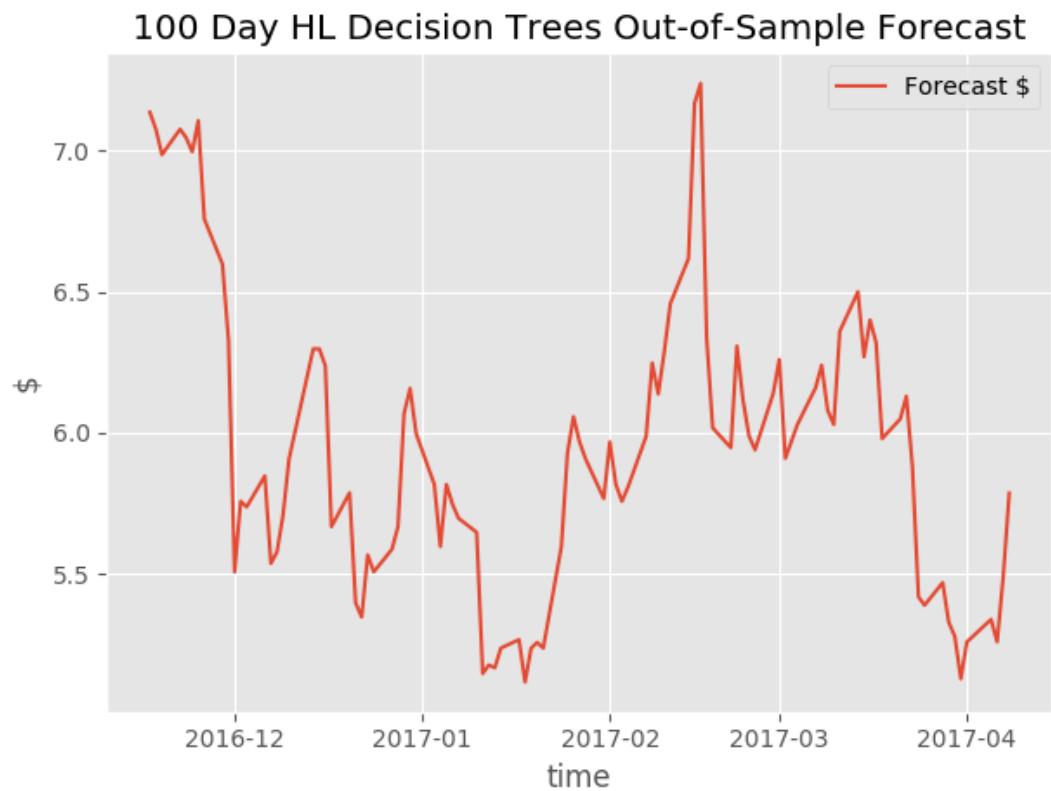


FIGURE 4.111: 100 day HL Decision Trees out-of-sample forecast

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.

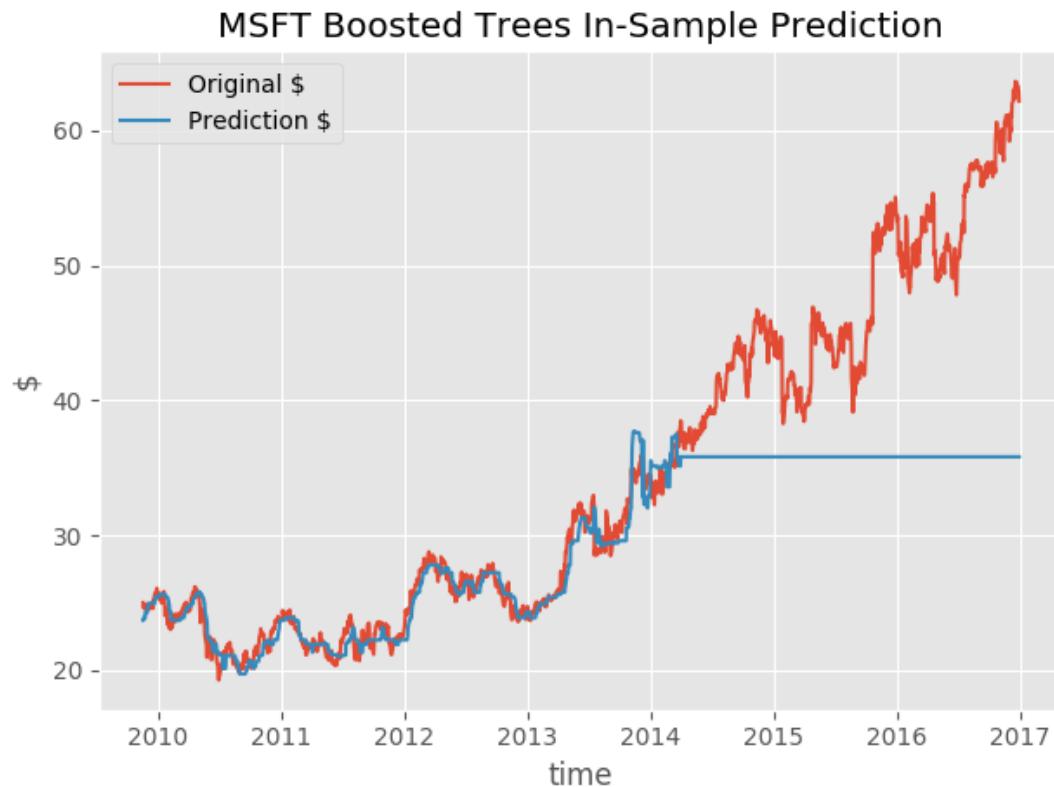


FIGURE 4.112: MSFT Boosted Decision Trees in-sample prediction

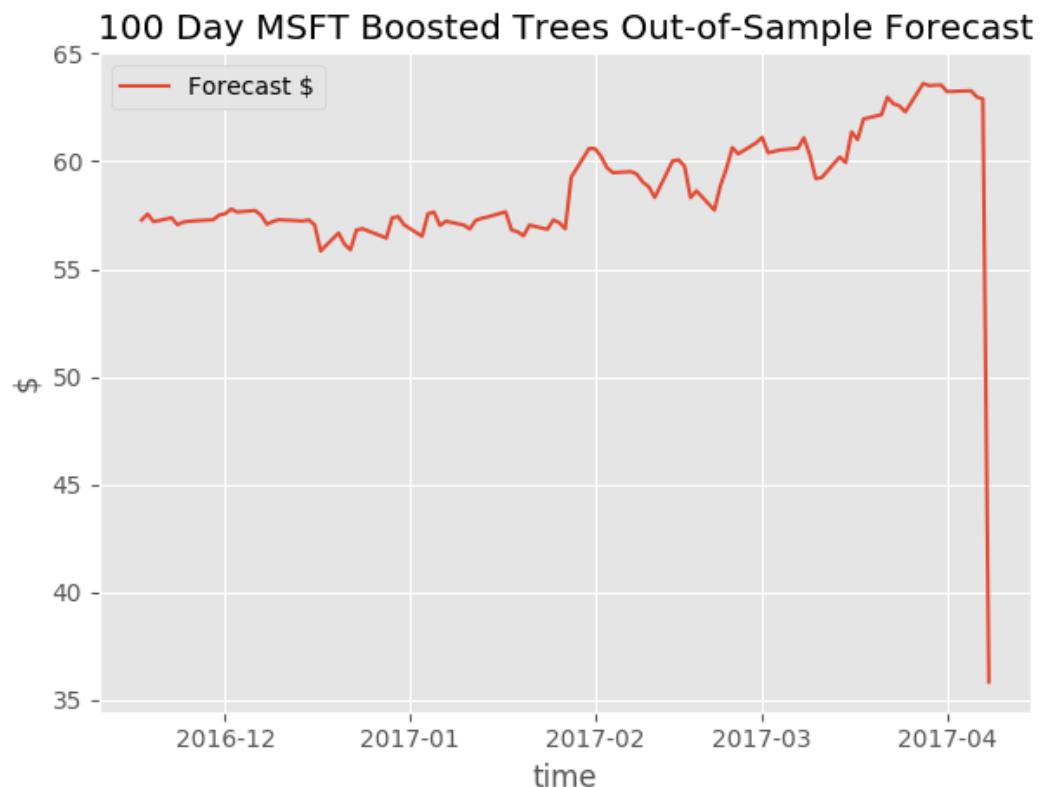


FIGURE 4.113: 100 day MSFT Boosted Decision Trees out-of-sample forecast

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

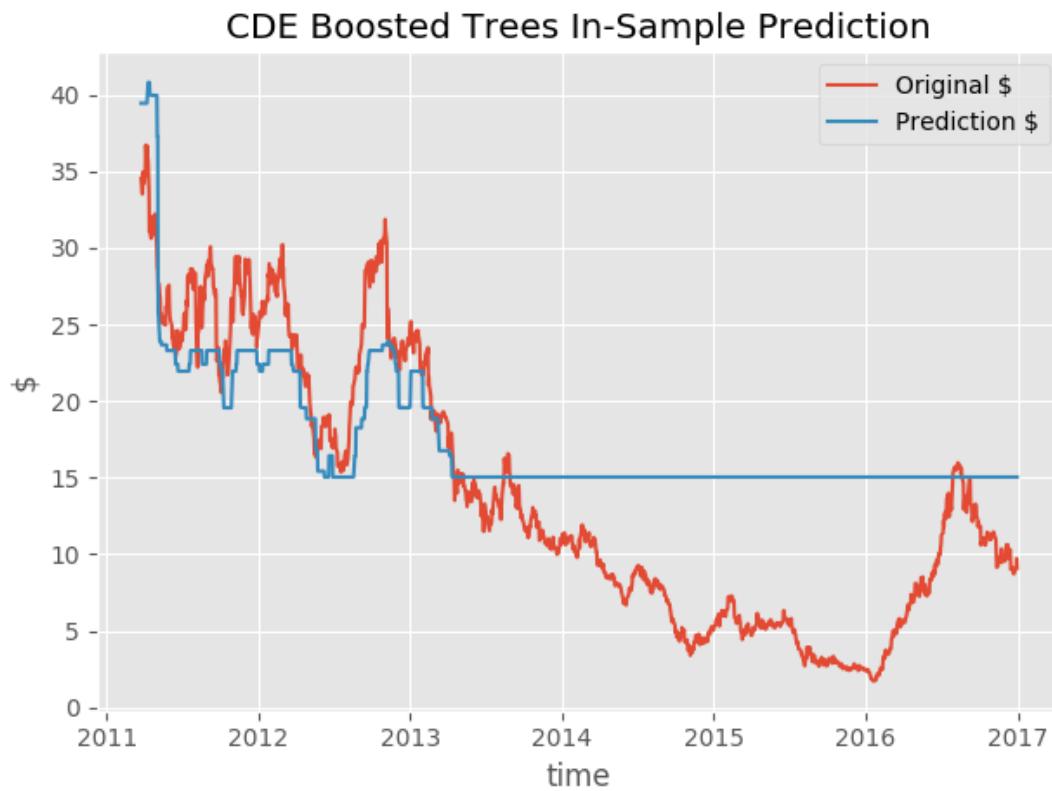


FIGURE 4.114: CDE Boosted Decision Trees in-sample prediction

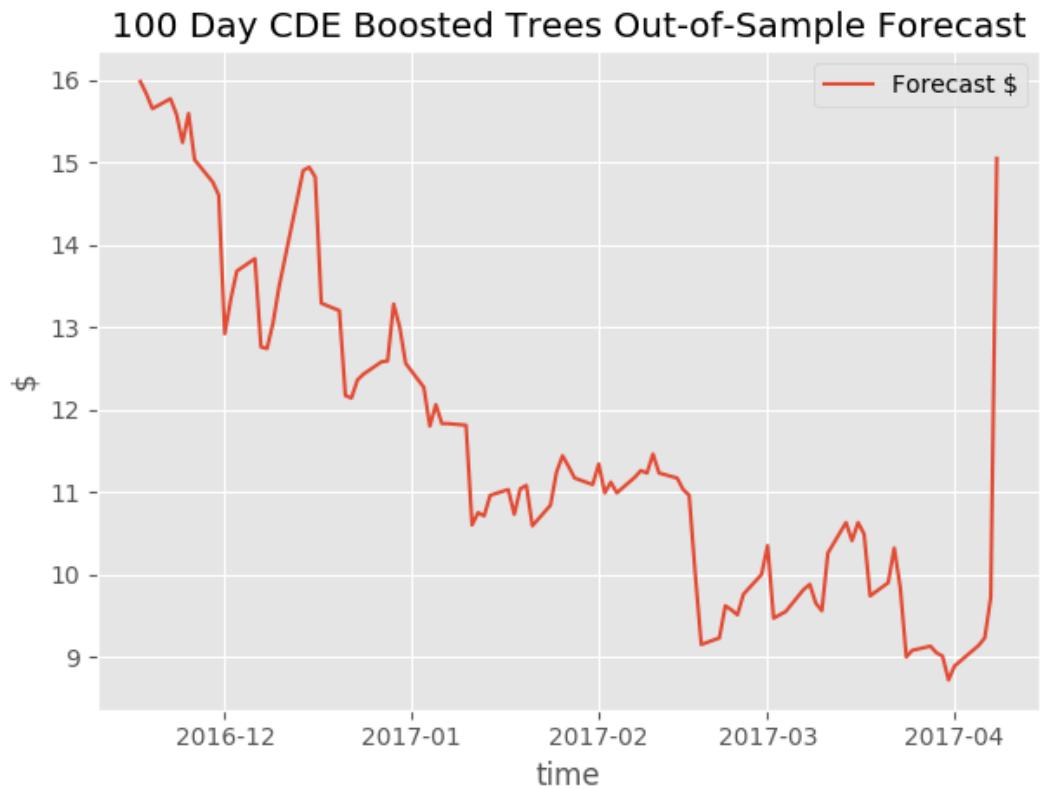


FIGURE 4.115: 100 day CDE Boosted Decision Trees out-of-sample forecast

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

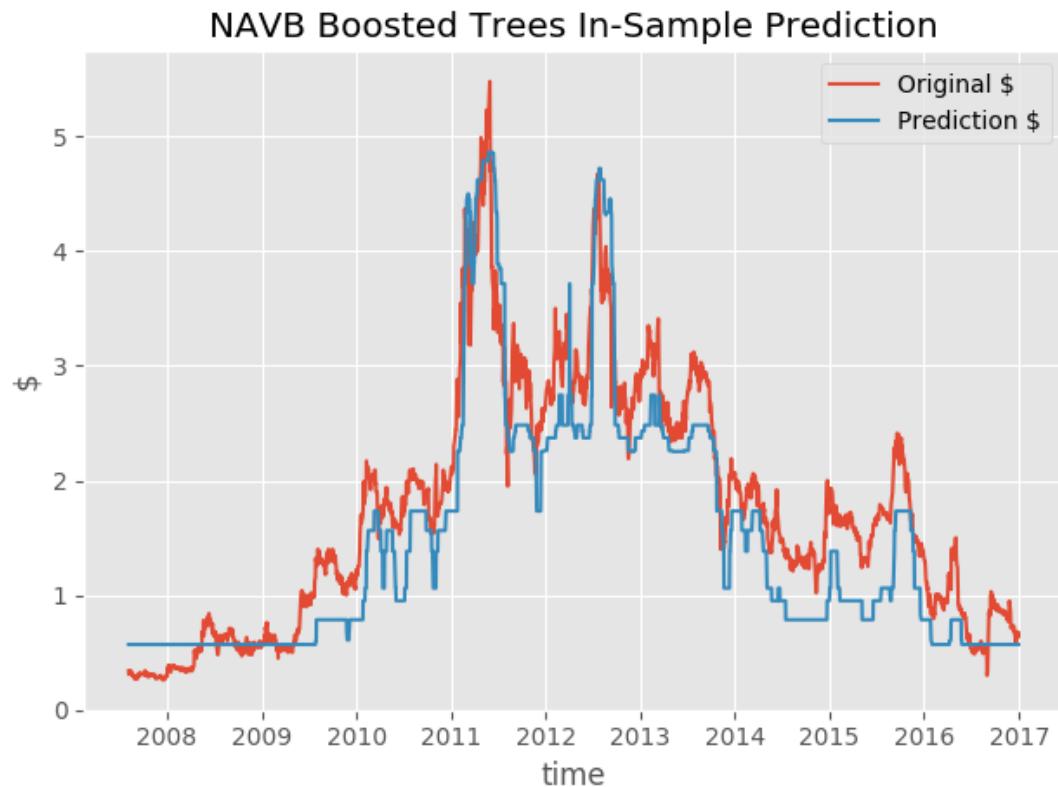


FIGURE 4.116: NAVB Boosted Decision Trees in-sample prediction

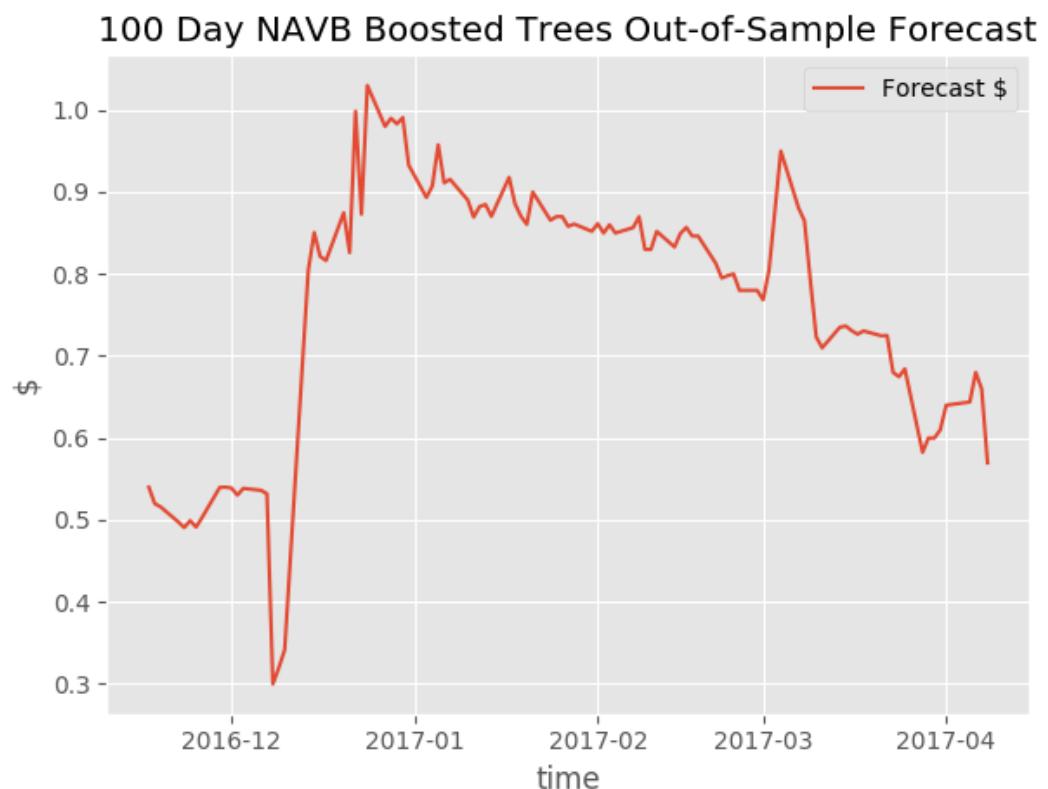


FIGURE 4.117: 100 day NAVB Boosted Decision Trees out-of-sample forecast

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

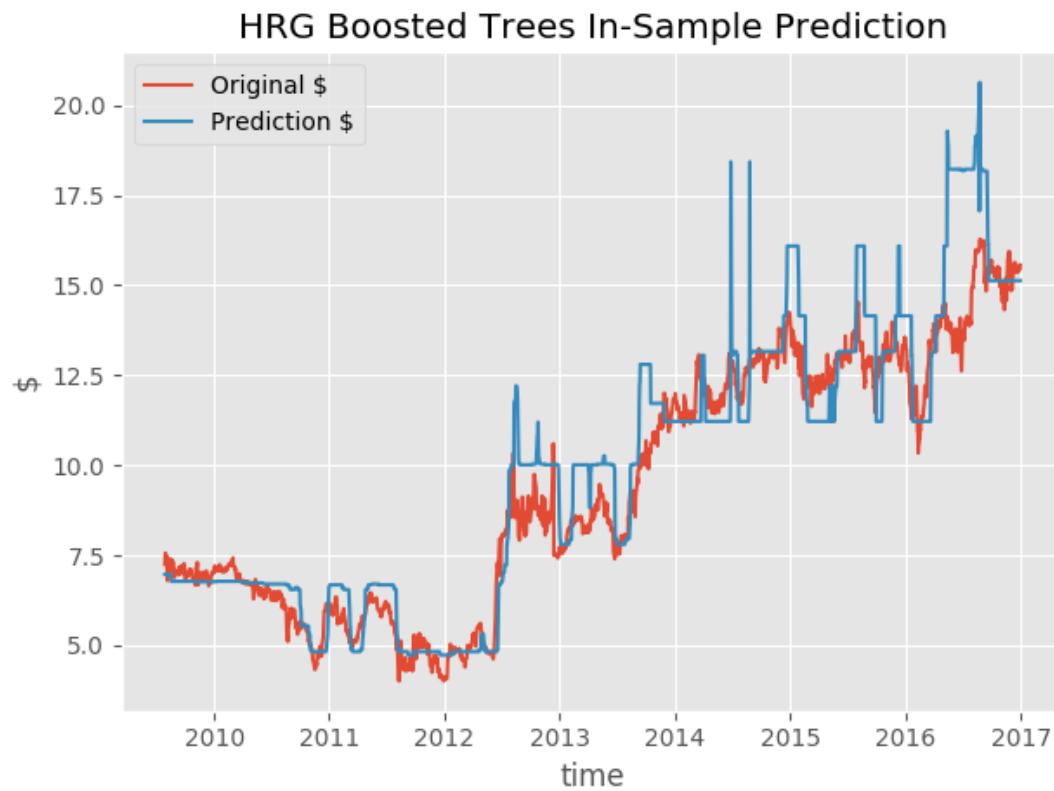


FIGURE 4.118: HRG Boosted Decision Trees in-sample prediction

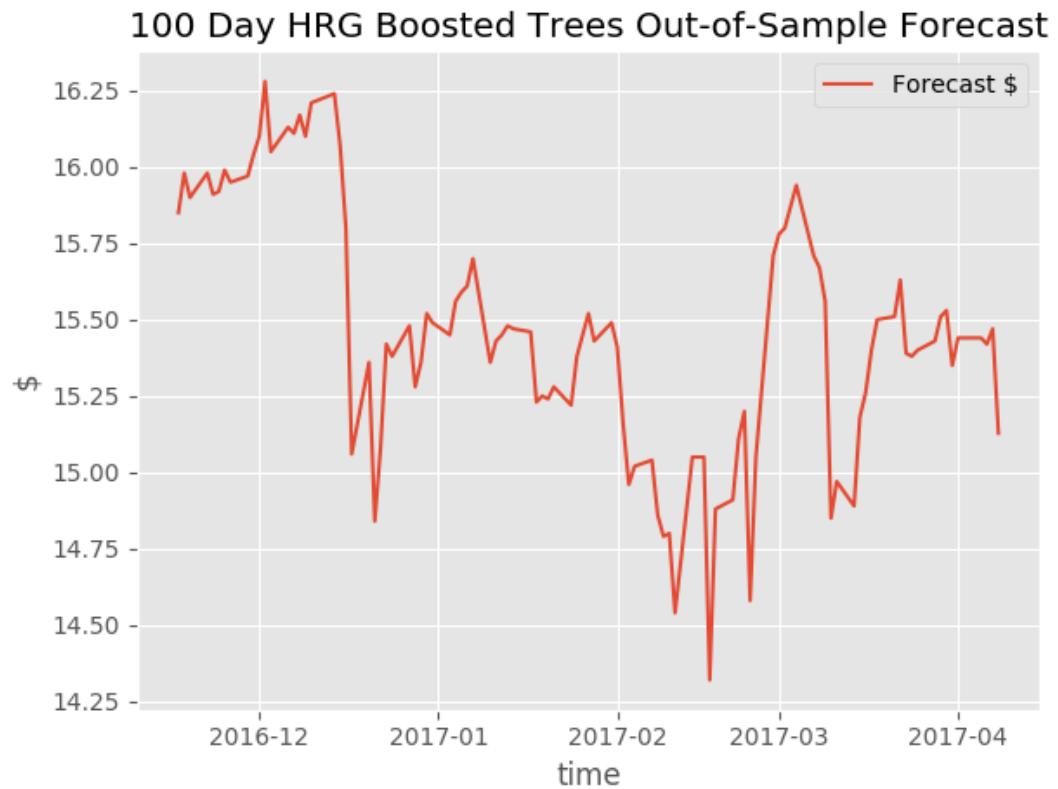


FIGURE 4.119: 100 day HRG Boosted Decision Trees out-of-sample forecast

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

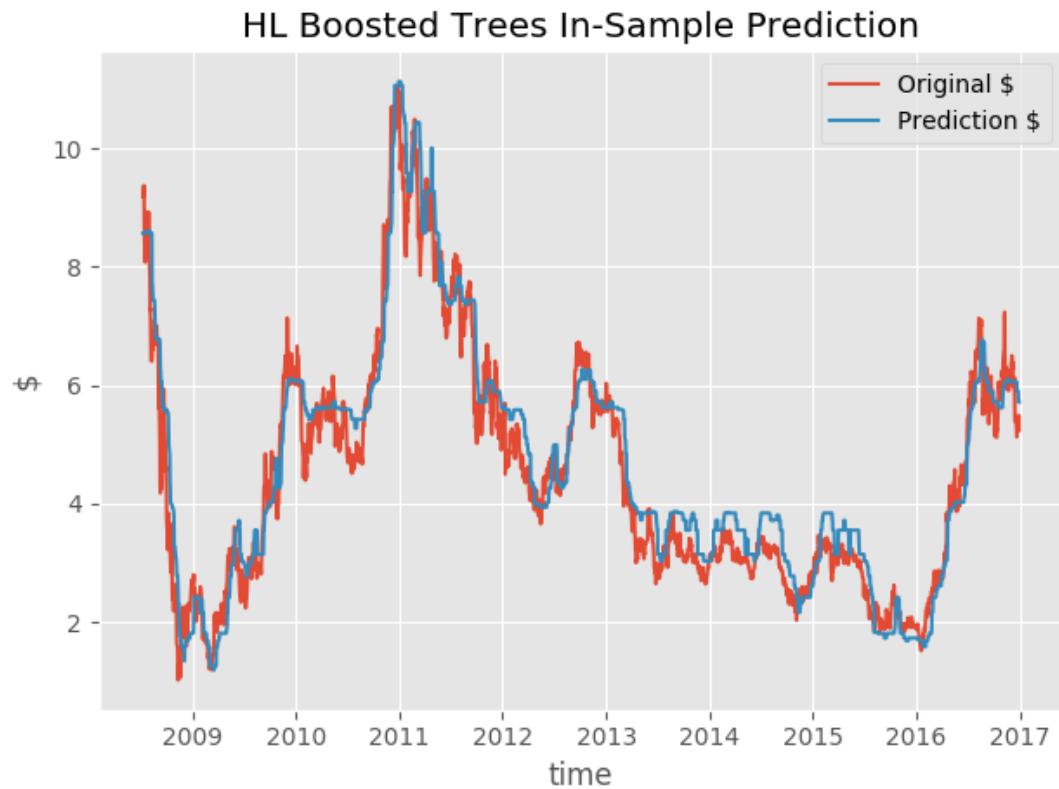


FIGURE 4.120: HL Boosted Decision Trees in-sample prediction

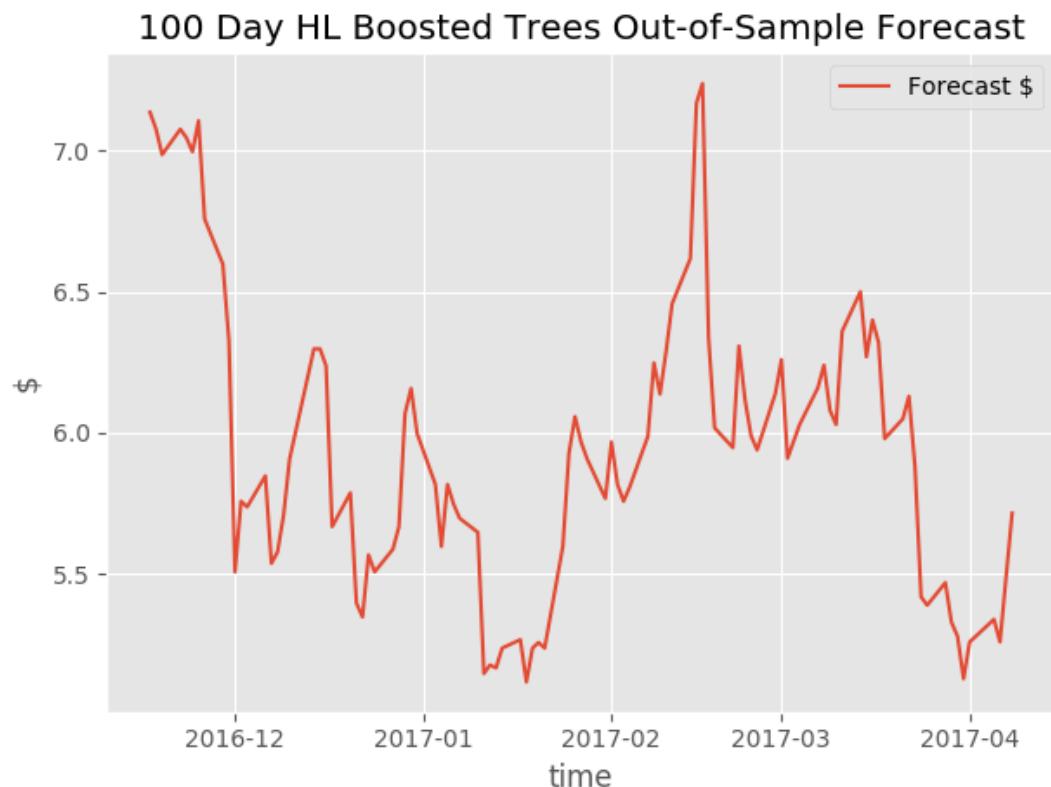


FIGURE 4.121: 100 day HL Boosted Decision Trees out-of-sample forecast

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.

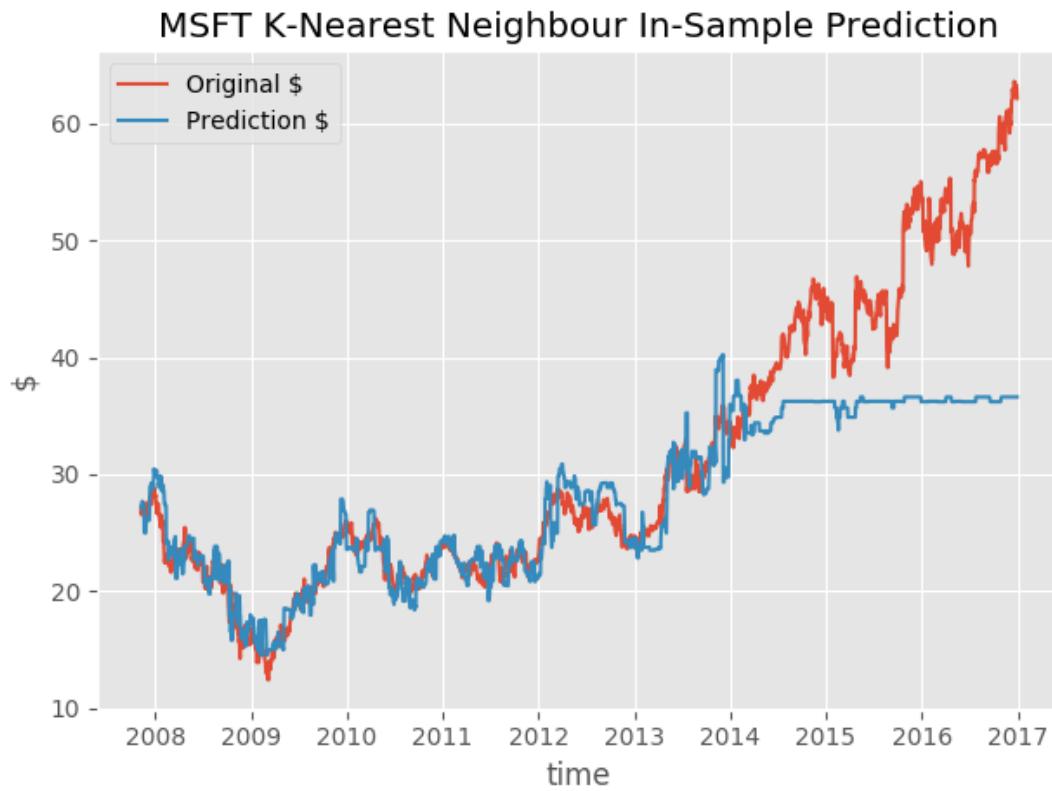


FIGURE 4.122: MSFT K-Nearest Neighbour in-sample prediction

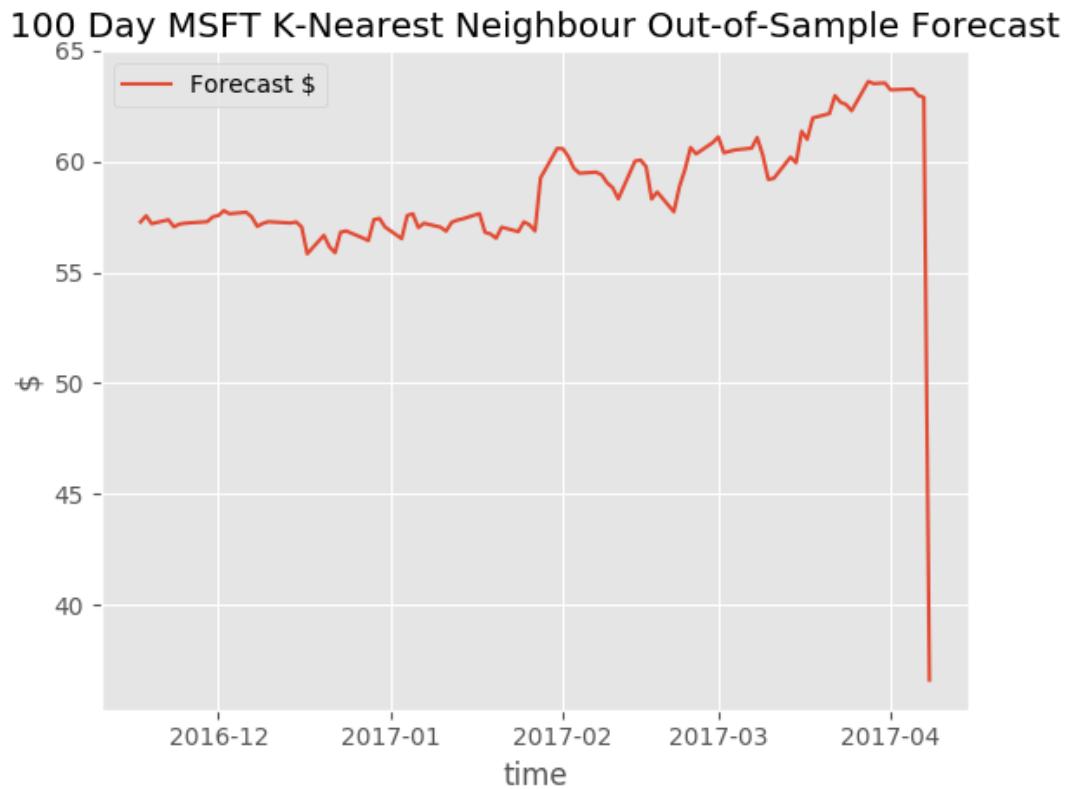


FIGURE 4.123: 100 day MSFT K-Nearest Neighbour out-of-sample forecast

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

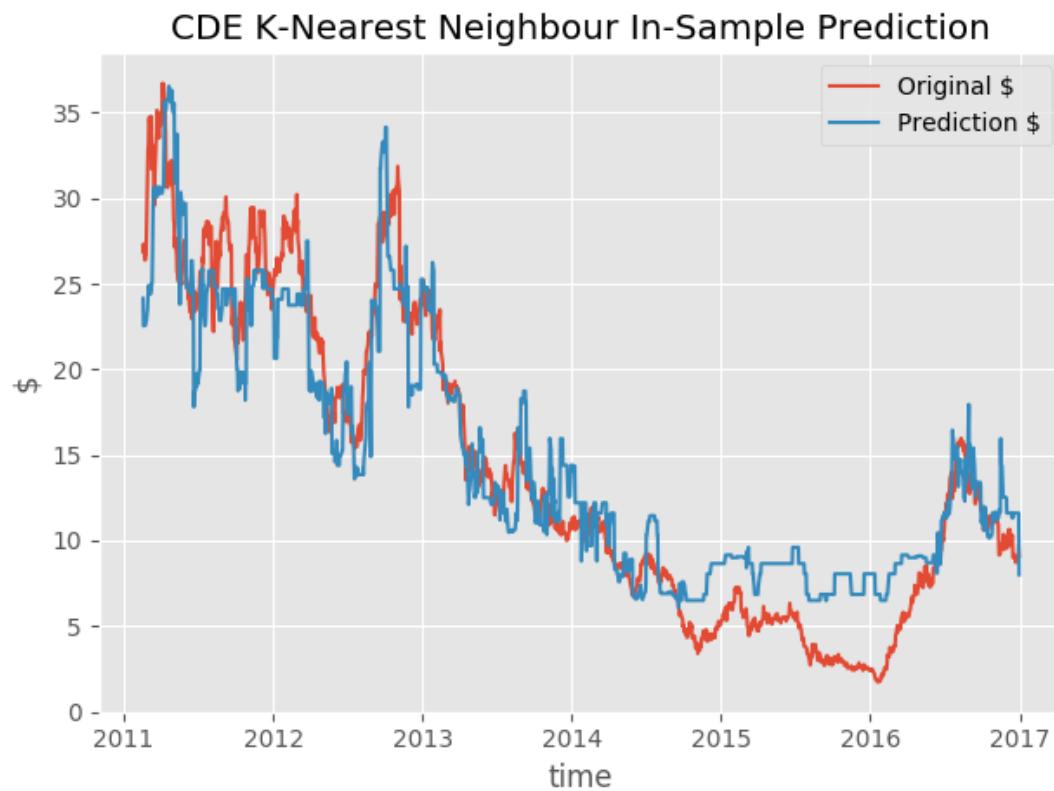


FIGURE 4.124: CDE K-Nearest Neighbour in-sample prediction

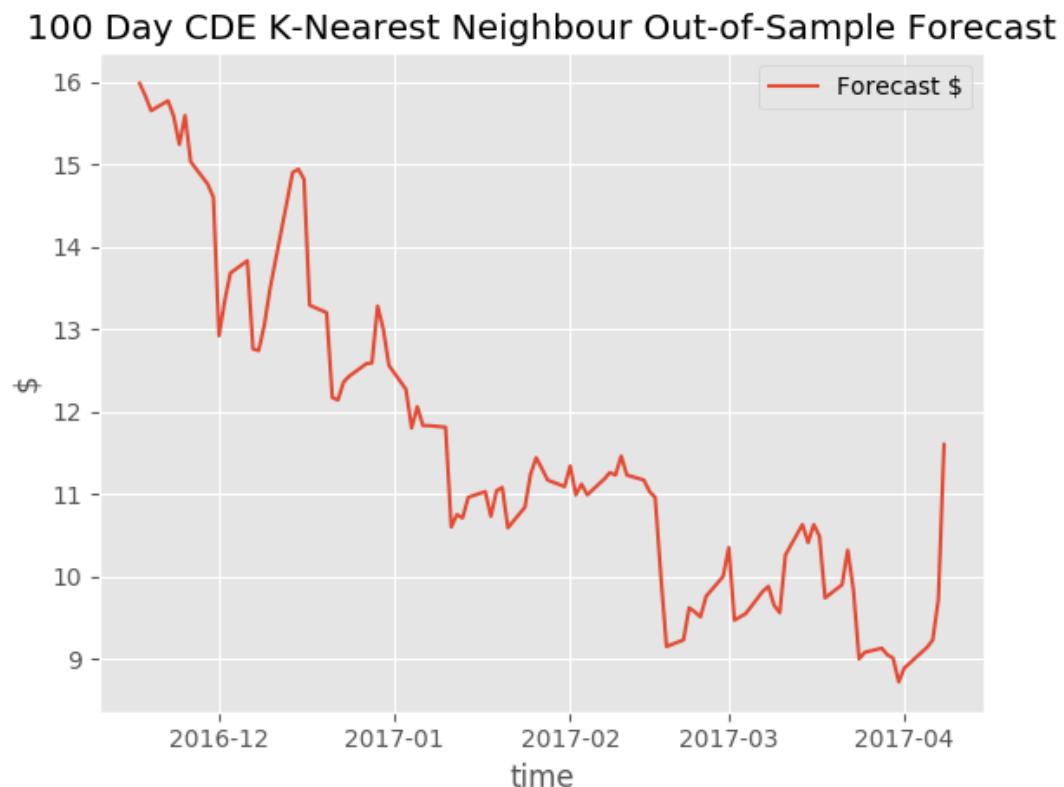


FIGURE 4.125: 100 day CDE K-Nearest Neighbour out-of-sample forecast

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

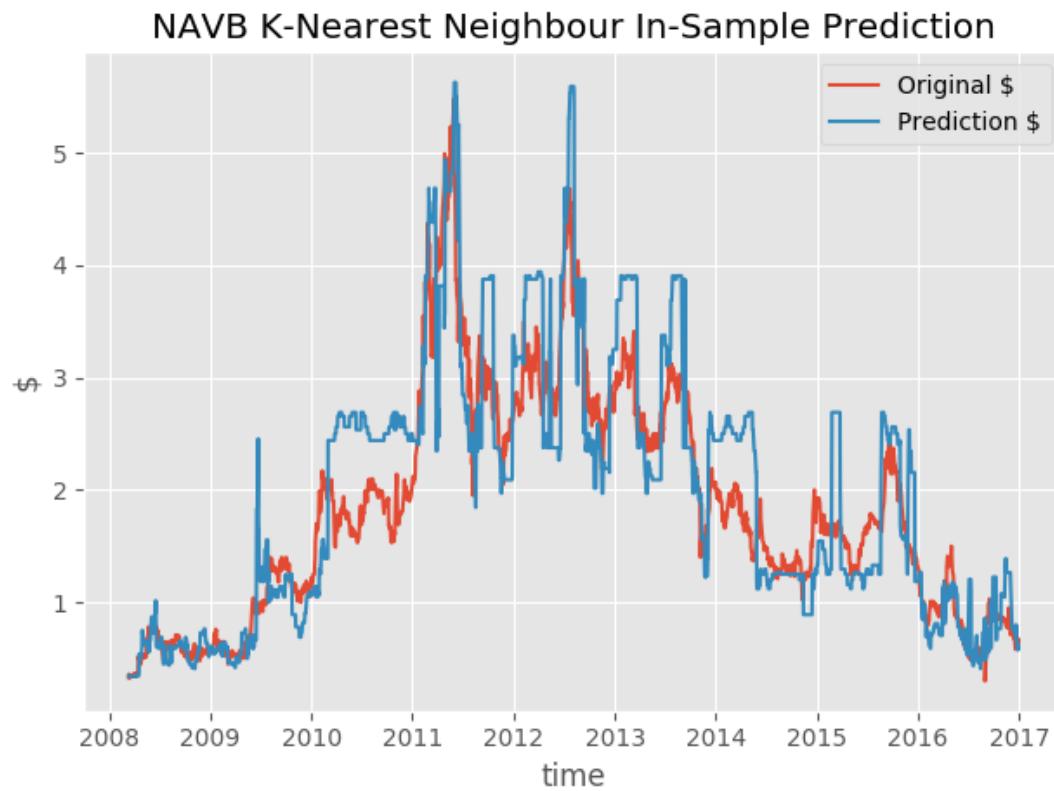


FIGURE 4.126: NAVB K-Nearest Neighbour in-sample prediction

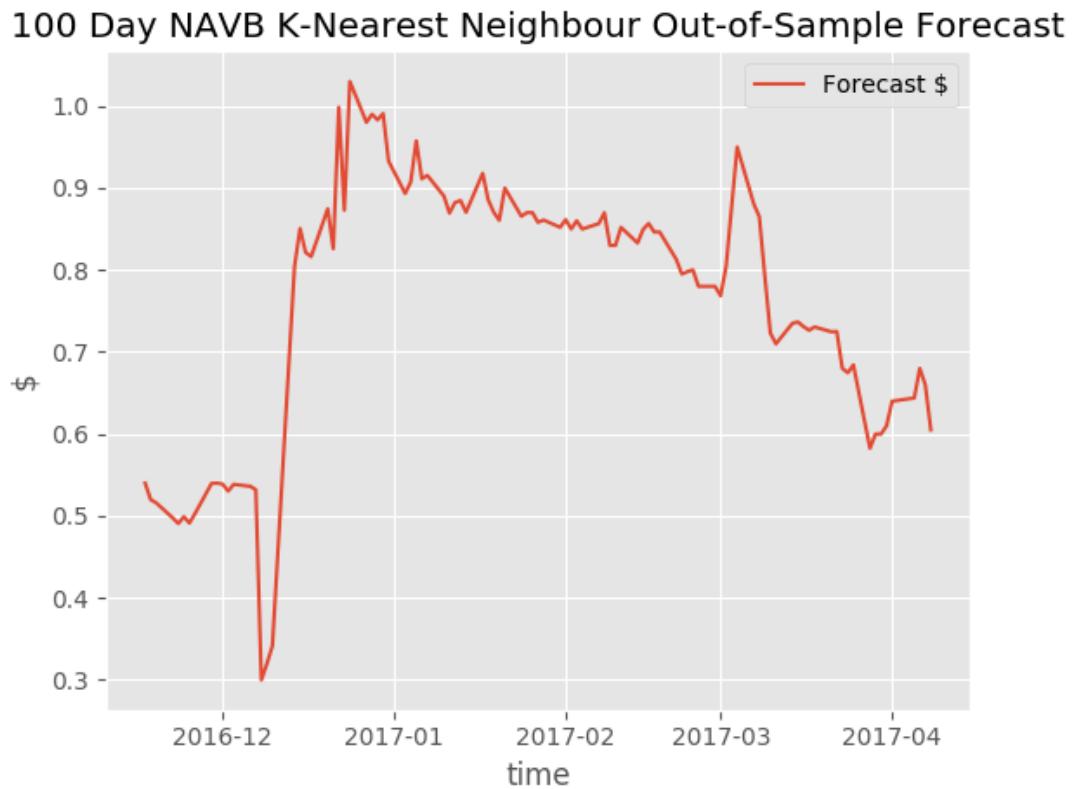


FIGURE 4.127: 100 day NAVB K-Nearest Neighbour out-of-sample forecast

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

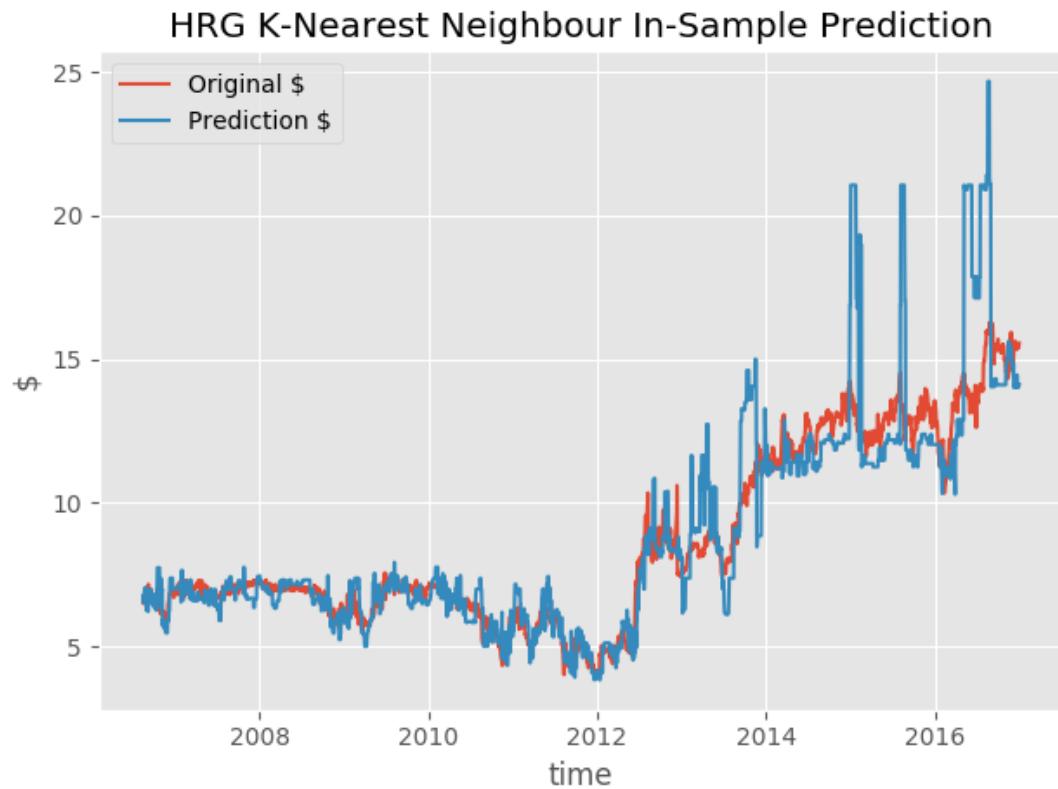


FIGURE 4.128: HRG K-Nearest Neighbour in-sample prediction

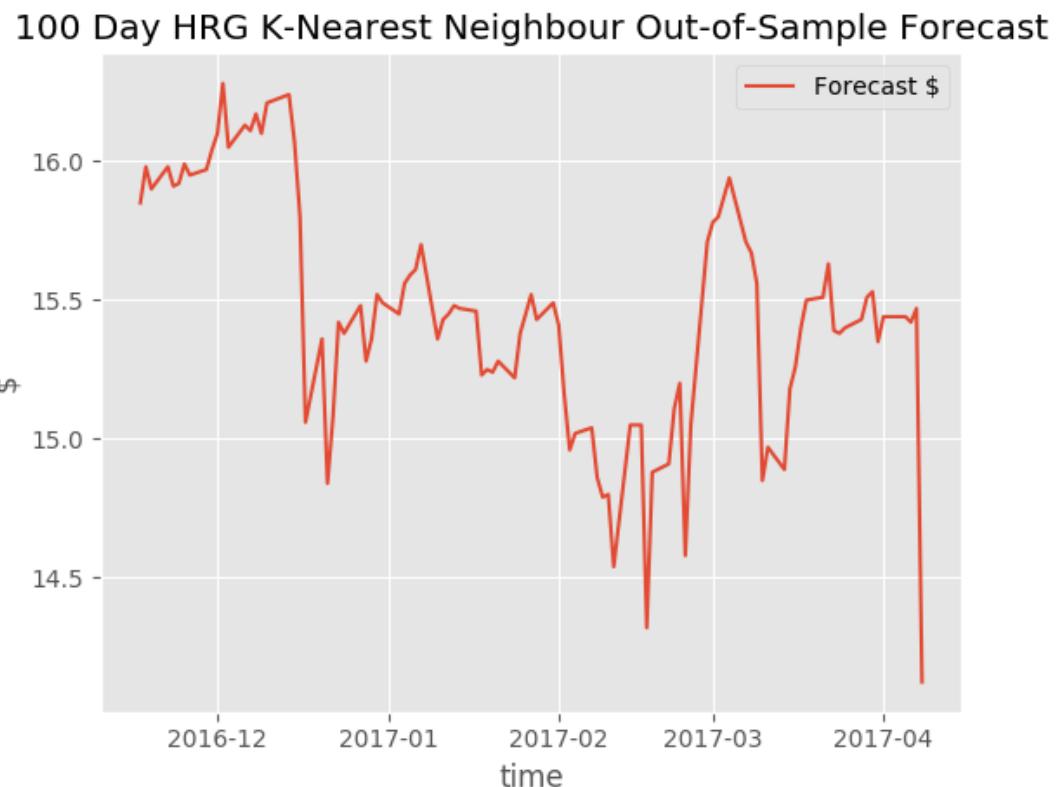


FIGURE 4.129: 100 day HRG K-Nearest Neighbour out-of-sample forecast

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

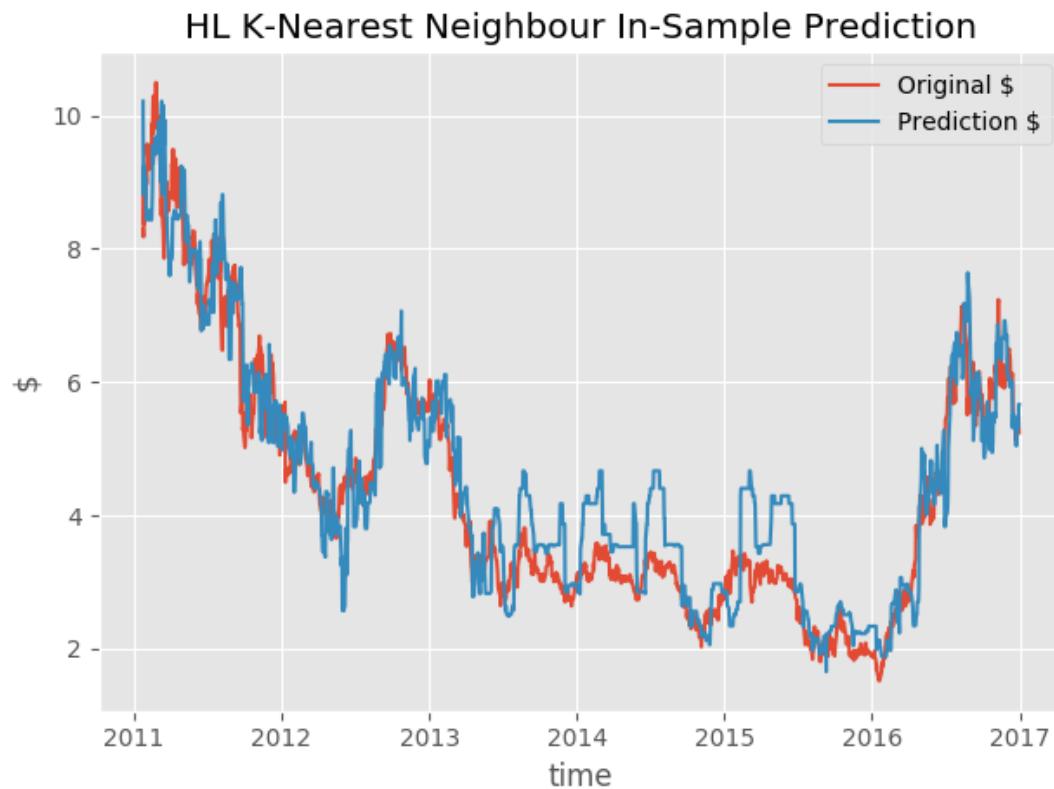


FIGURE 4.130: HL K-Nearest Neighbour in-sample prediction

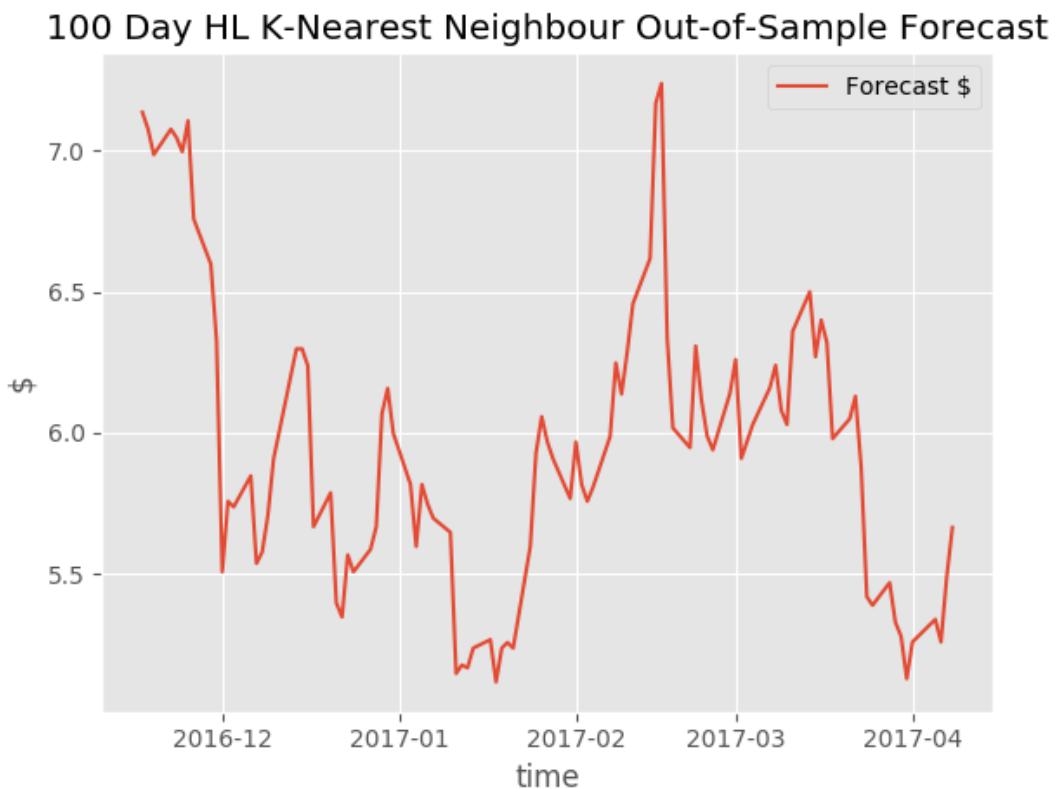


FIGURE 4.131: 100 day HL K-Nearest Neighbour out-of-sample forecast

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.

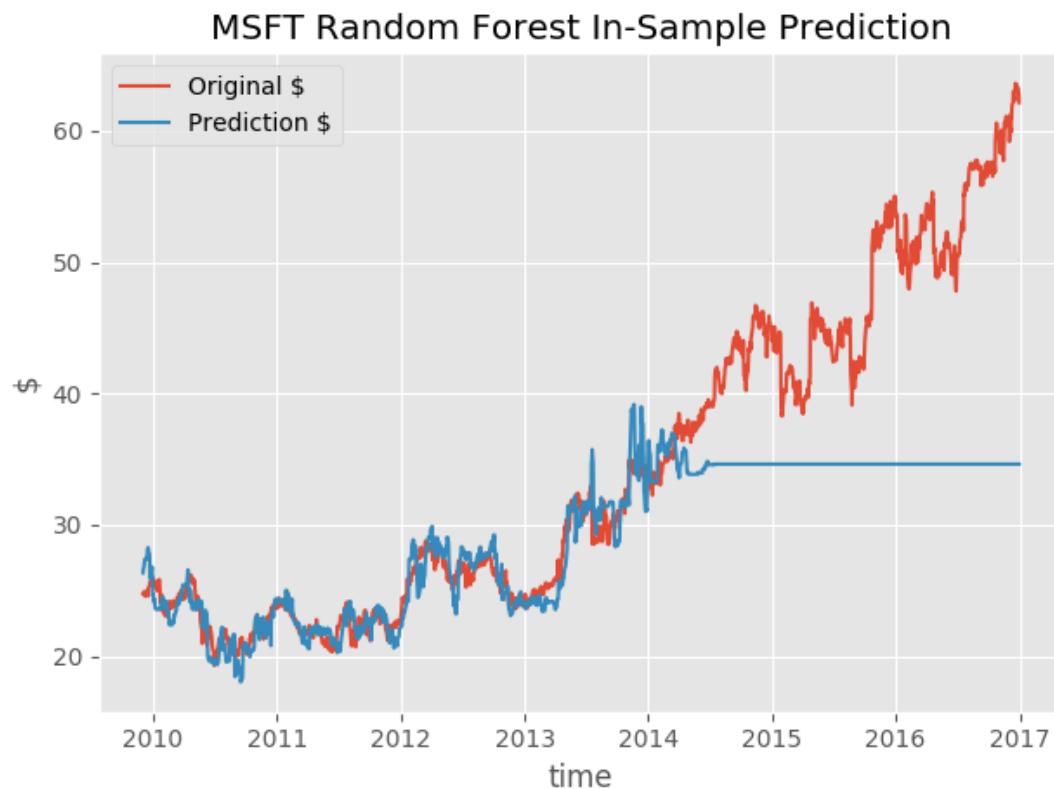


FIGURE 4.132: MSFT Random Forest in-sample prediction

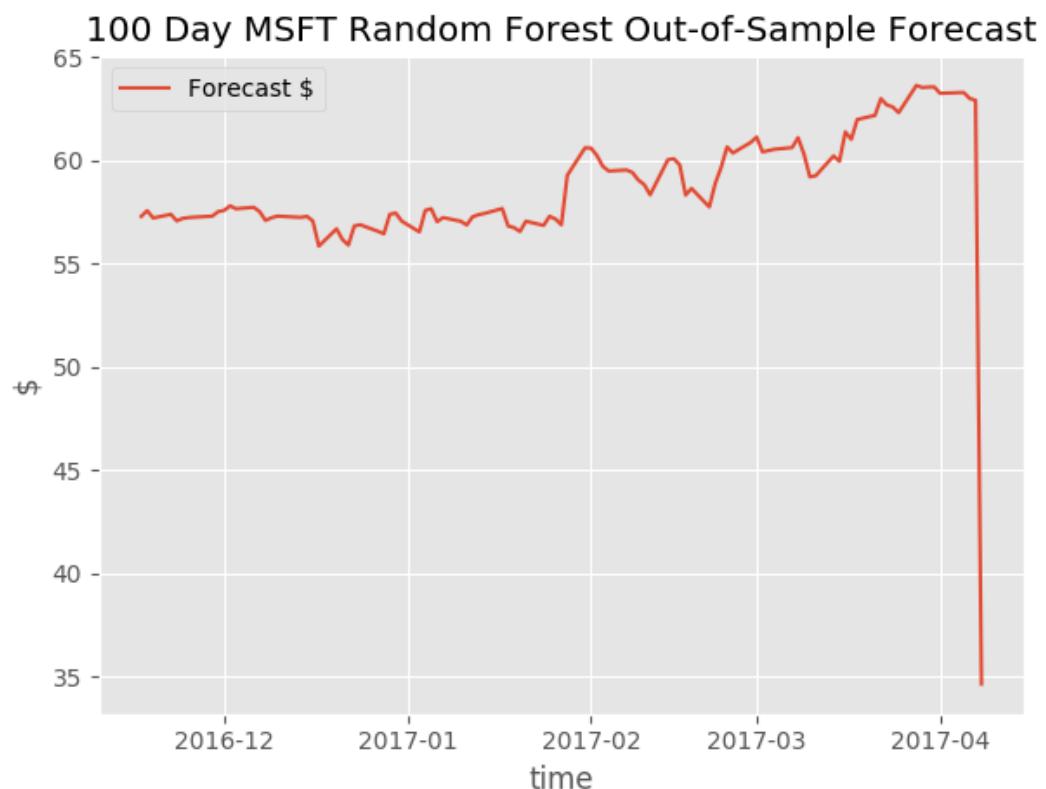


FIGURE 4.133: 100 day MSFT Random Forest out-of-sample forecast

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

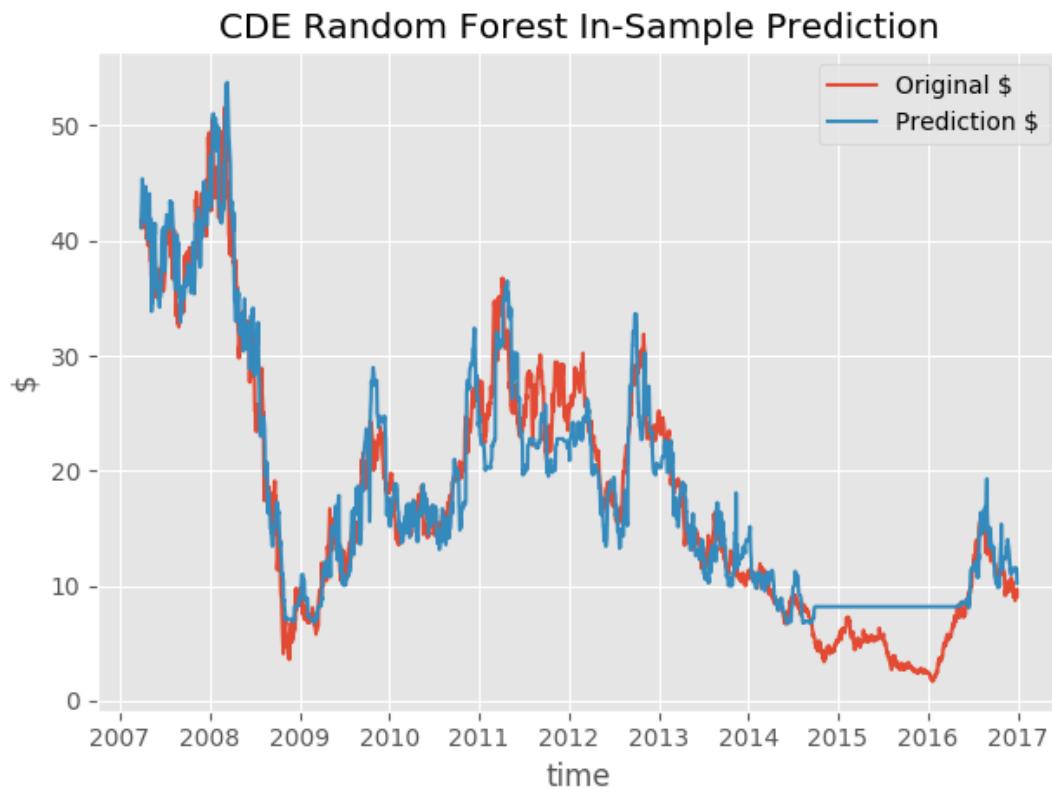


FIGURE 4.134: CDE Random Forest in-sample prediction

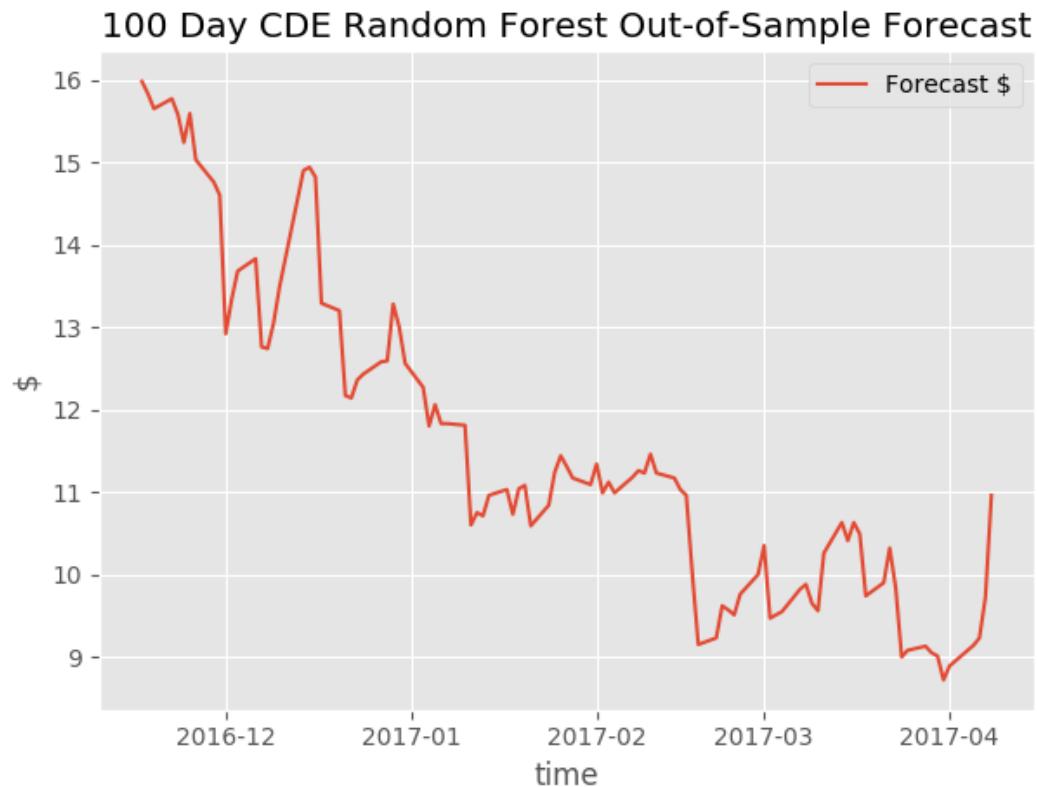


FIGURE 4.135: 100 day CDE Random Forest out-of-sample forecast

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

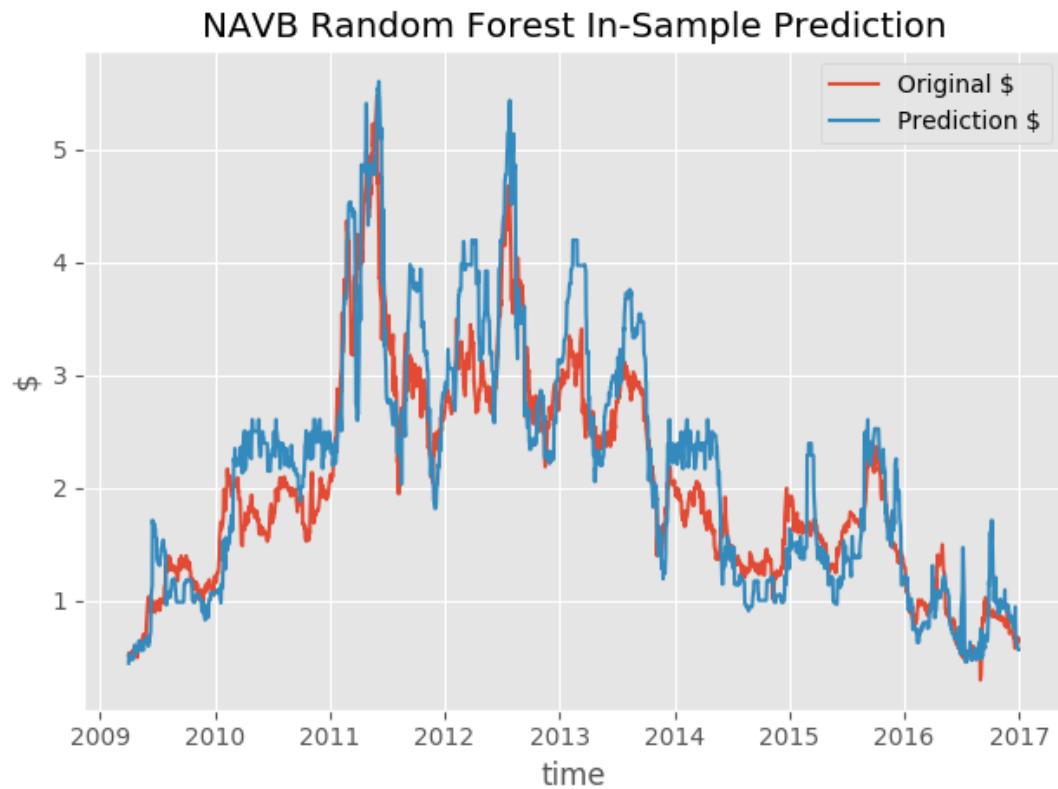


FIGURE 4.136: NAVB Random Forest in-sample prediction

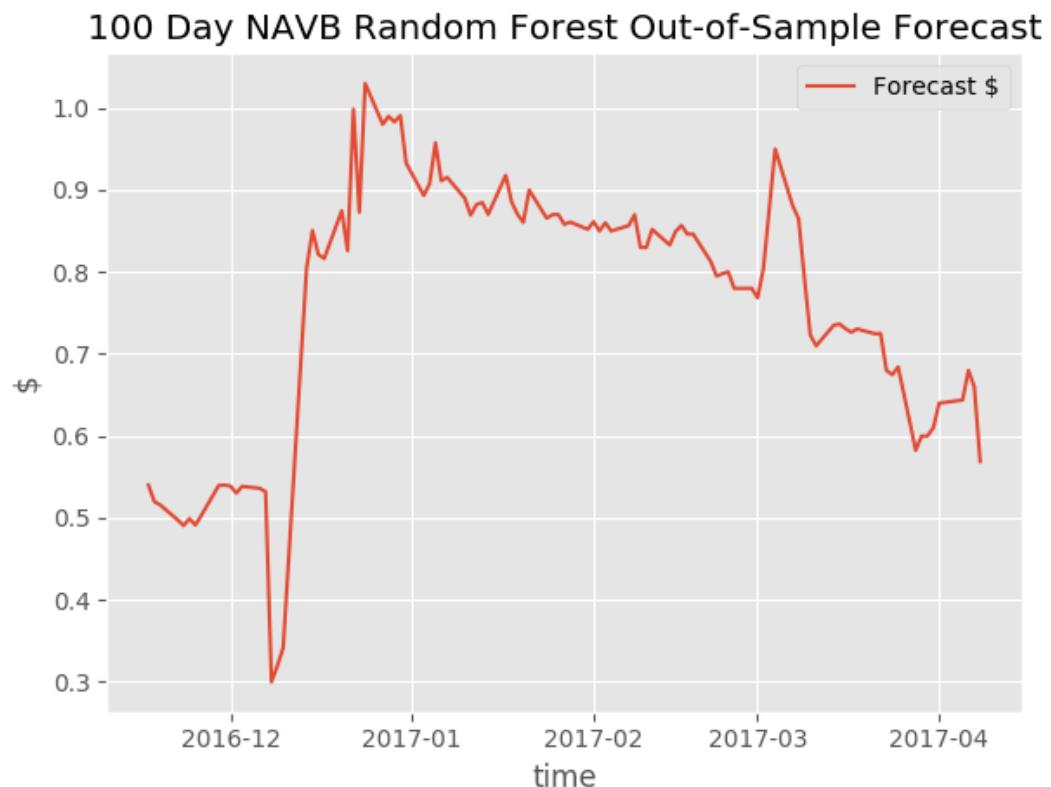


FIGURE 4.137: 100 day NAVB Random Forest out-of-sample forecast

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

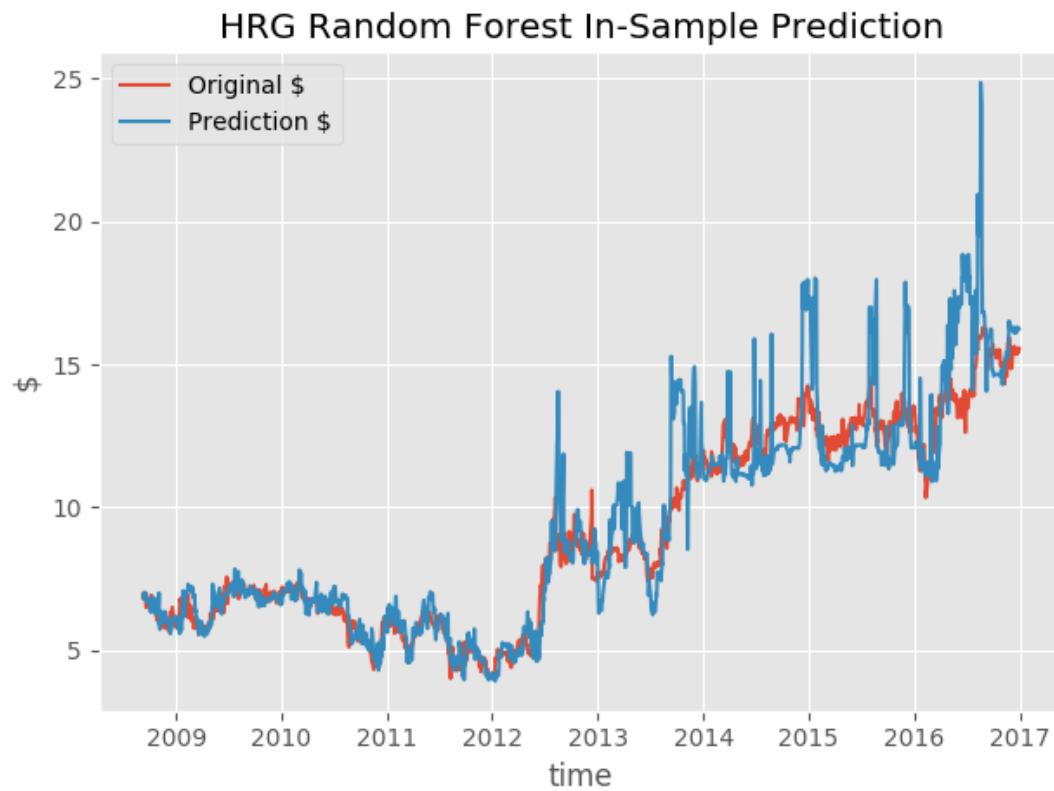


FIGURE 4.138: HRG Random Forest in-sample prediction

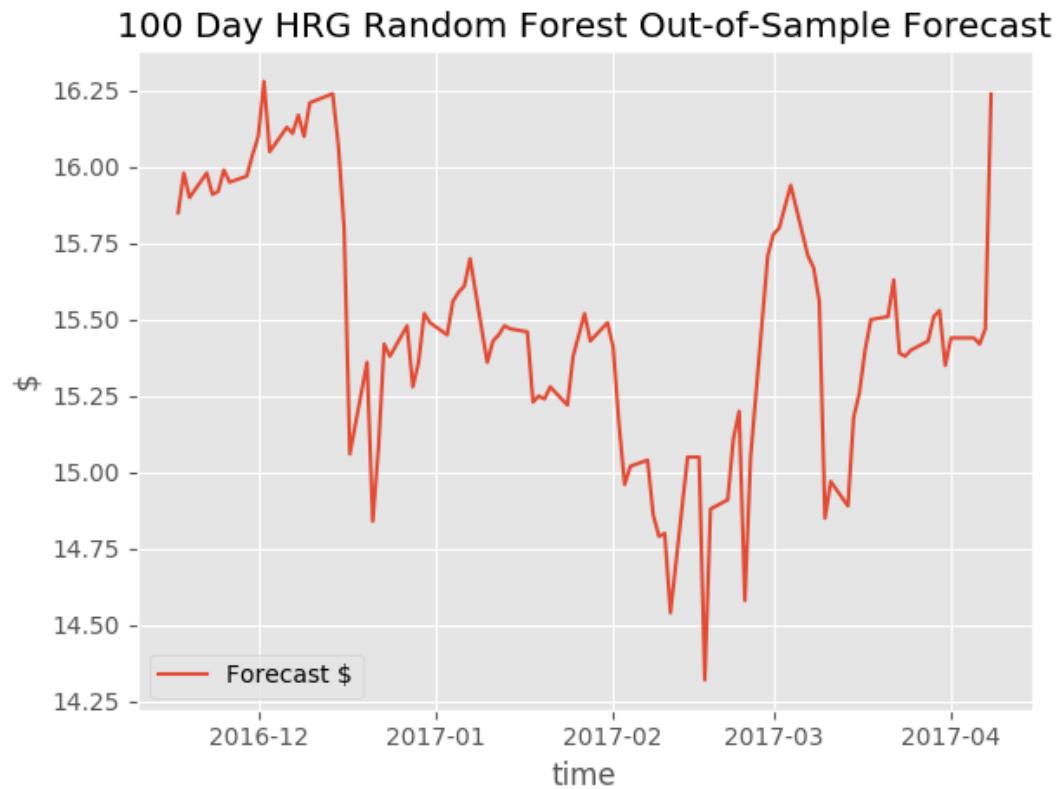


FIGURE 4.139: 100 day HRG Random Forest out-of-sample forecast

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

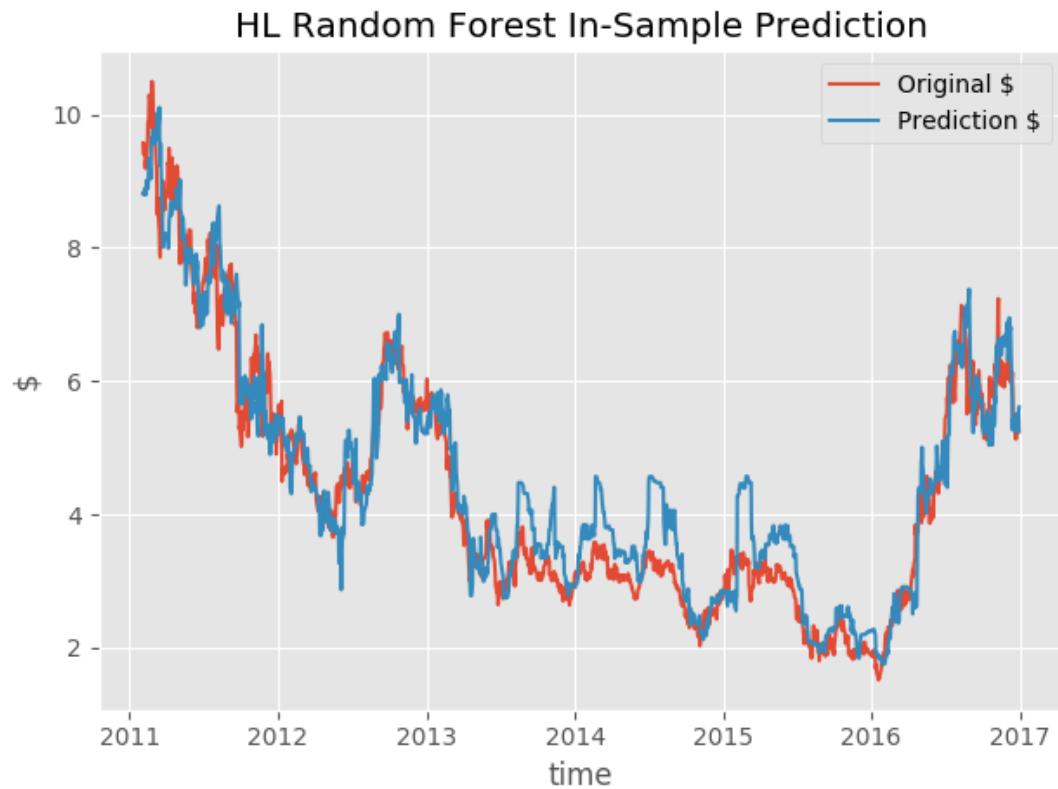


FIGURE 4.140: HL Random Forest in-sample prediction

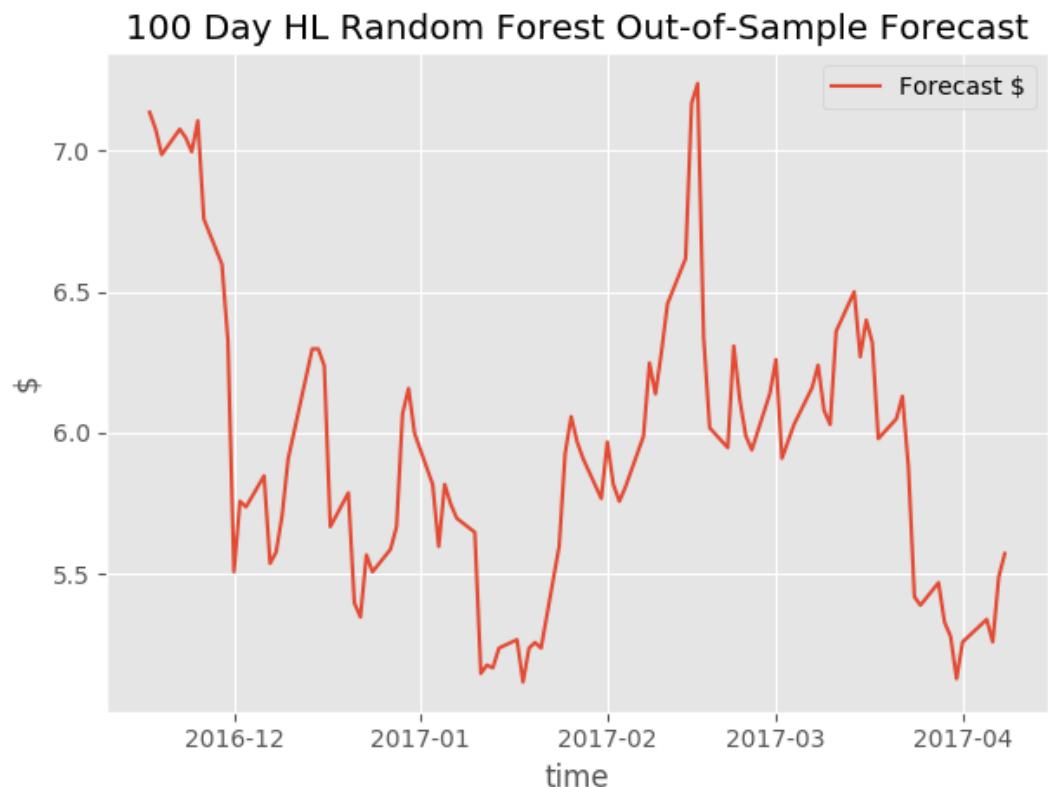


FIGURE 4.141: 100 day HL Random Forest out-of-sample forecast

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.

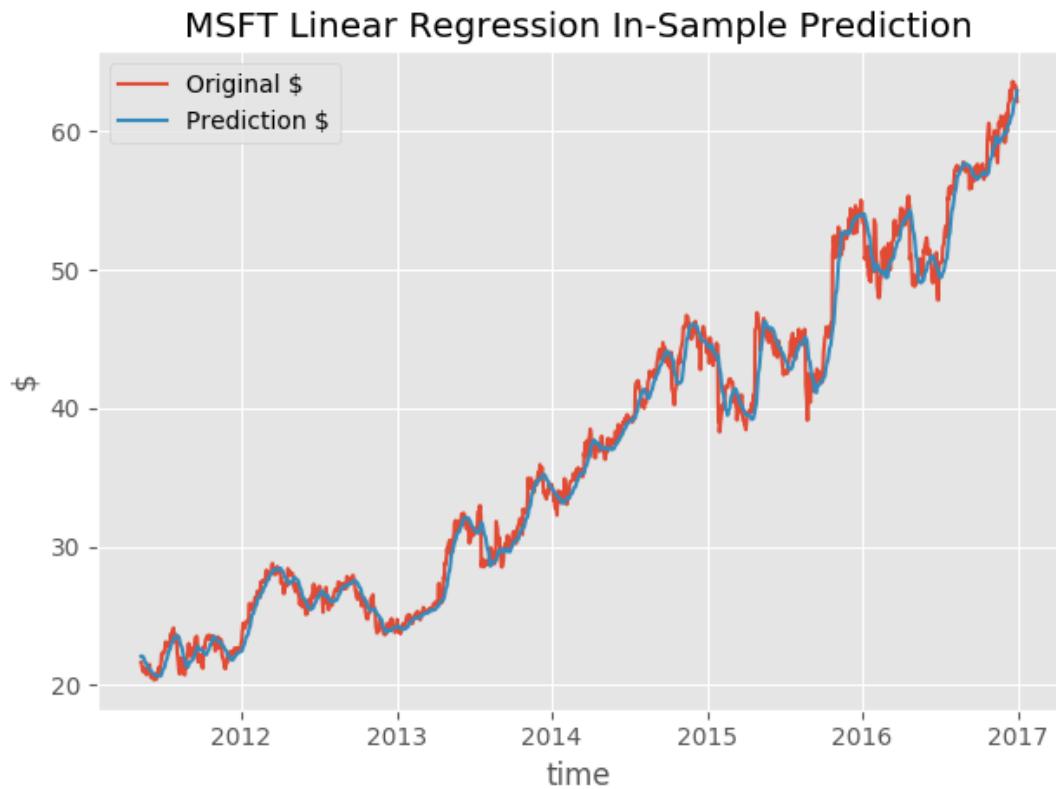


FIGURE 4.142: MSFT Linear Regression in-sample prediction

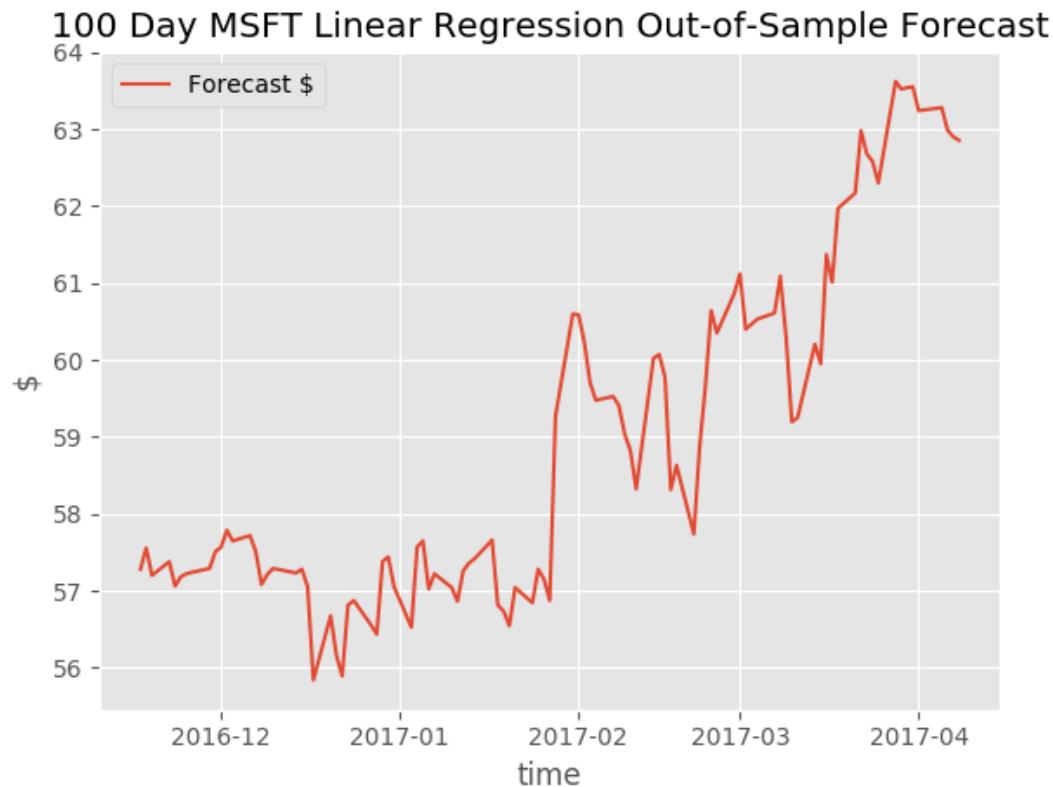


FIGURE 4.143: 100 day MSFT Linear Regression out-of-sample forecast

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

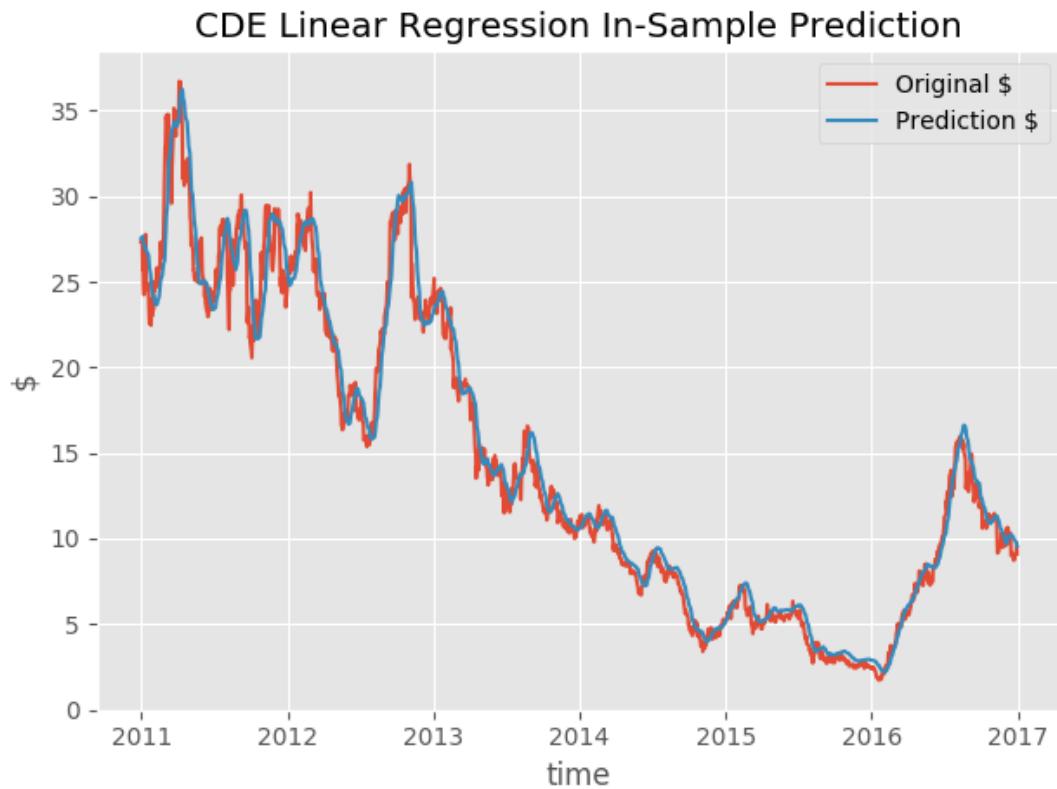


FIGURE 4.144: CDE Linear Regression in-sample prediction

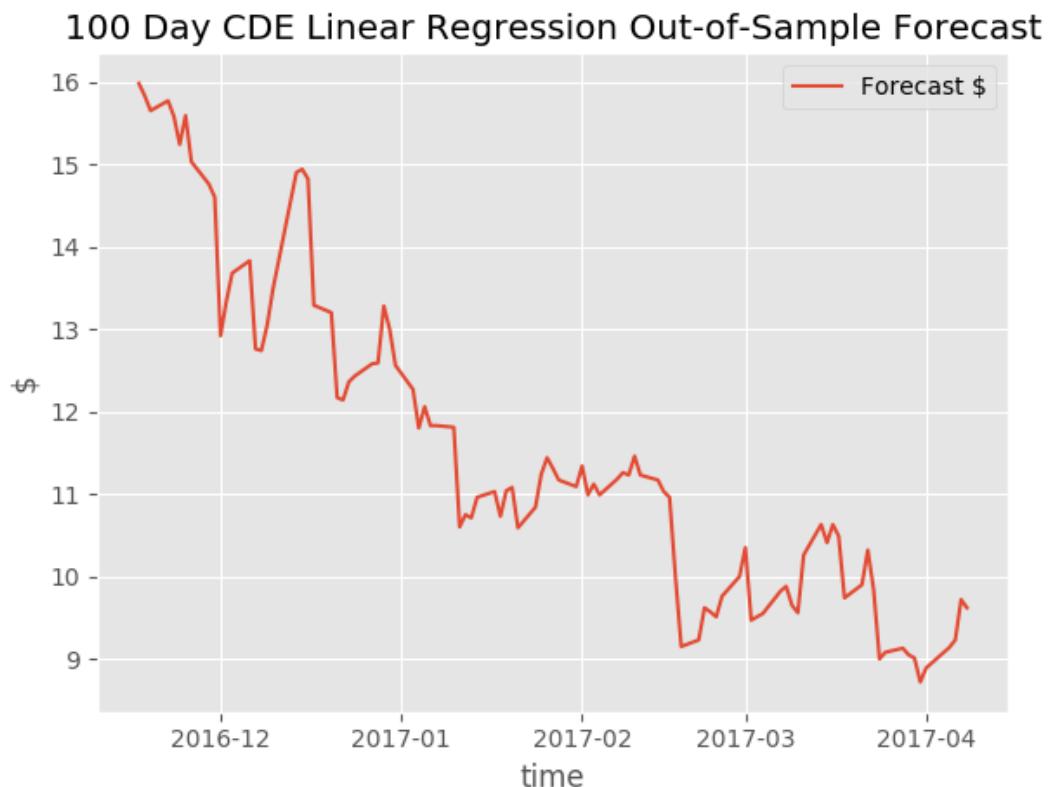
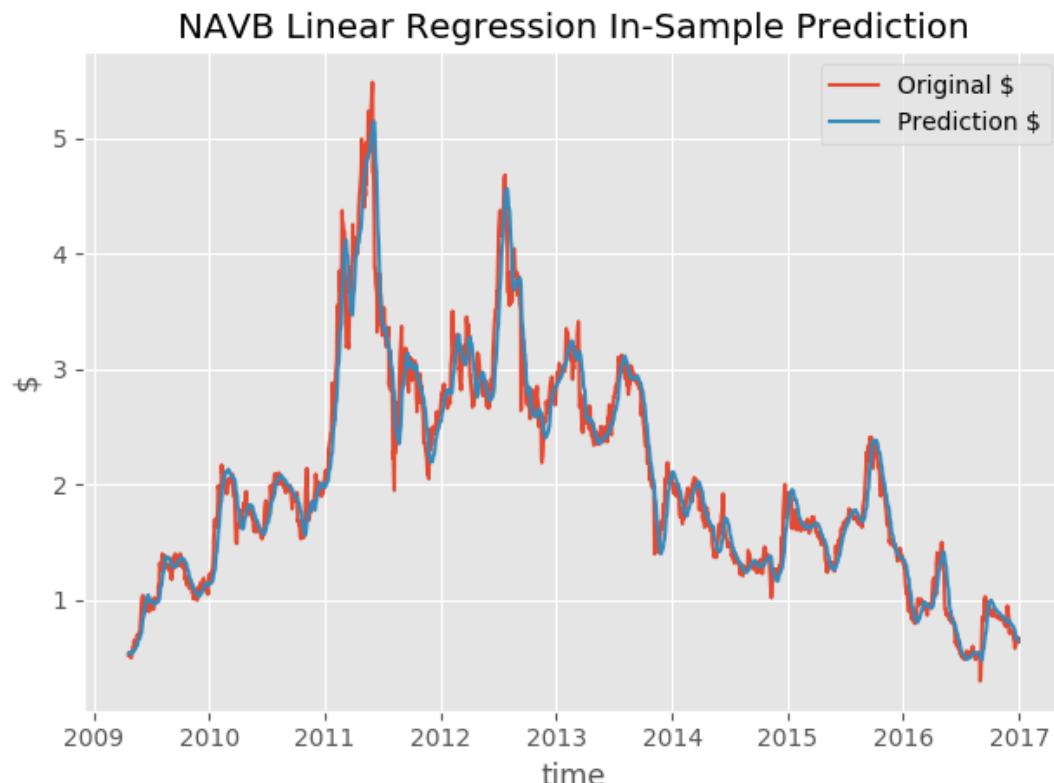
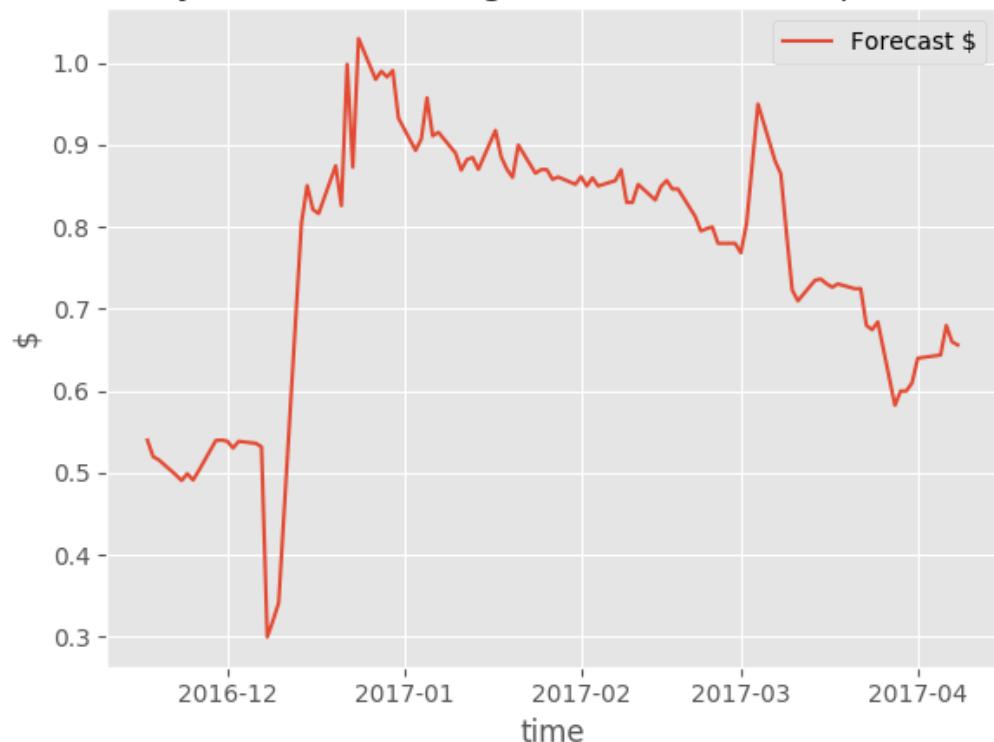


FIGURE 4.145: 100 day CDE Linear Regression out-of-sample forecast

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



100 Day NAVB Linear Regression Out-of-Sample Forecast



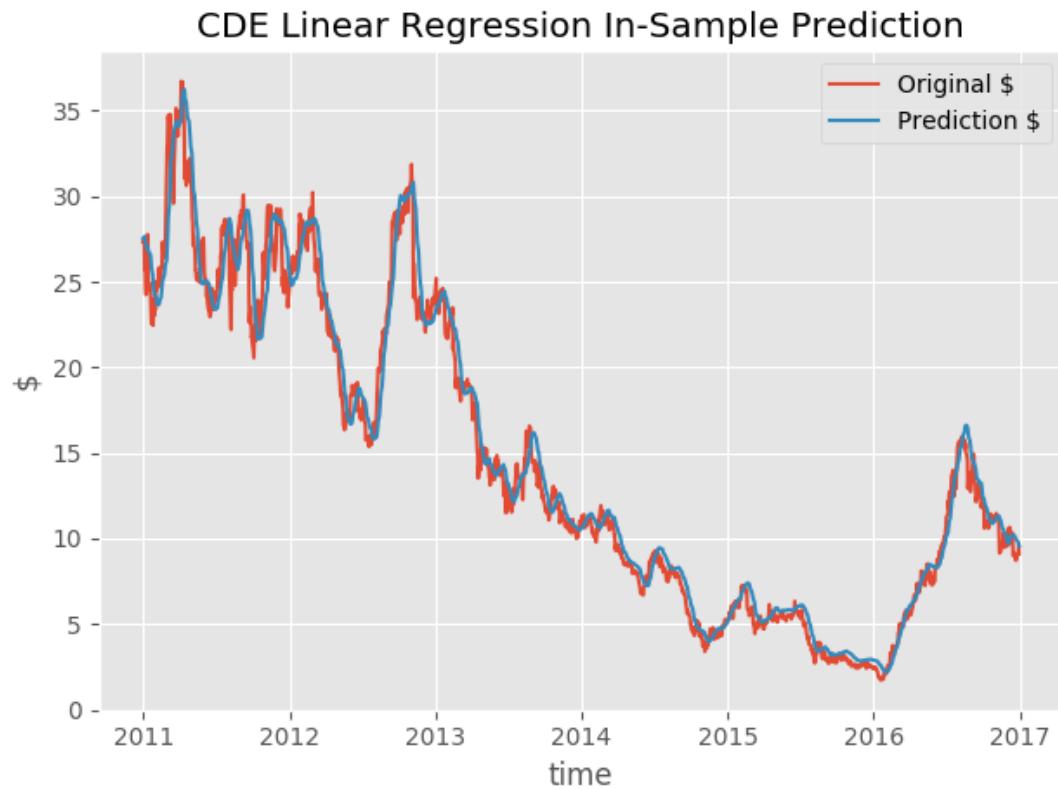


FIGURE 4.146: CDE Linear Regression in-sample prediction

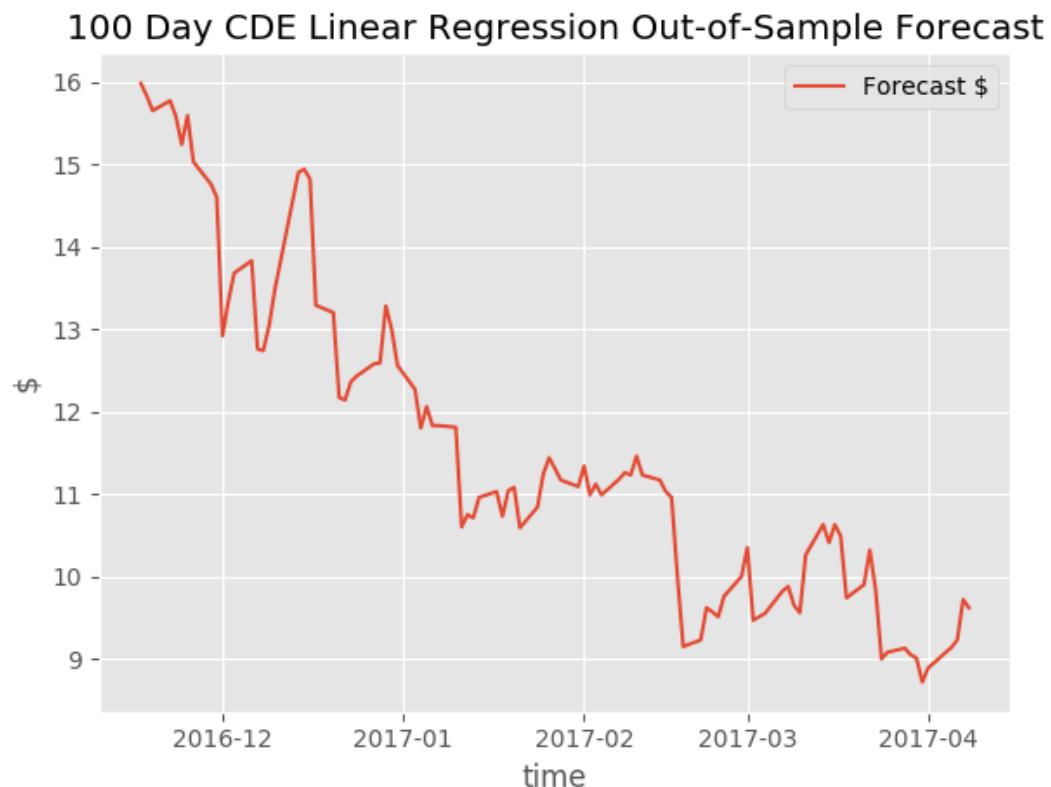


FIGURE 4.147: 100 day CDE Linear Regression out-of-sample forecast

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

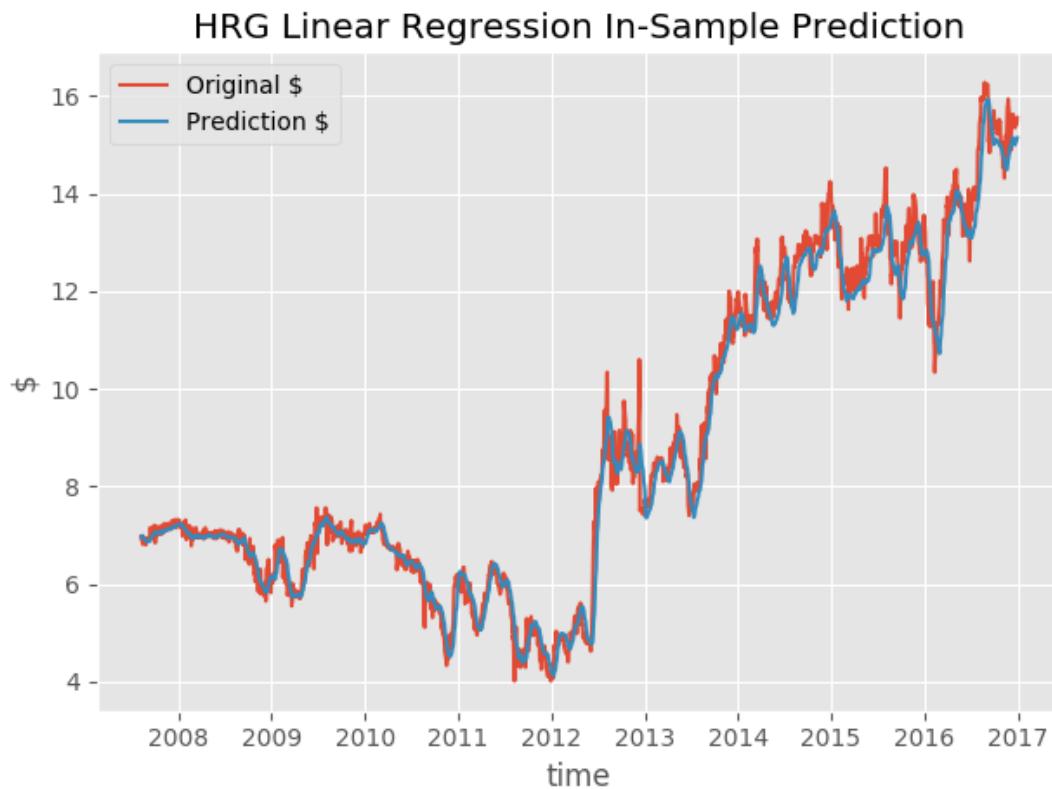


FIGURE 4.148: HRG Linear Regression in-sample prediction

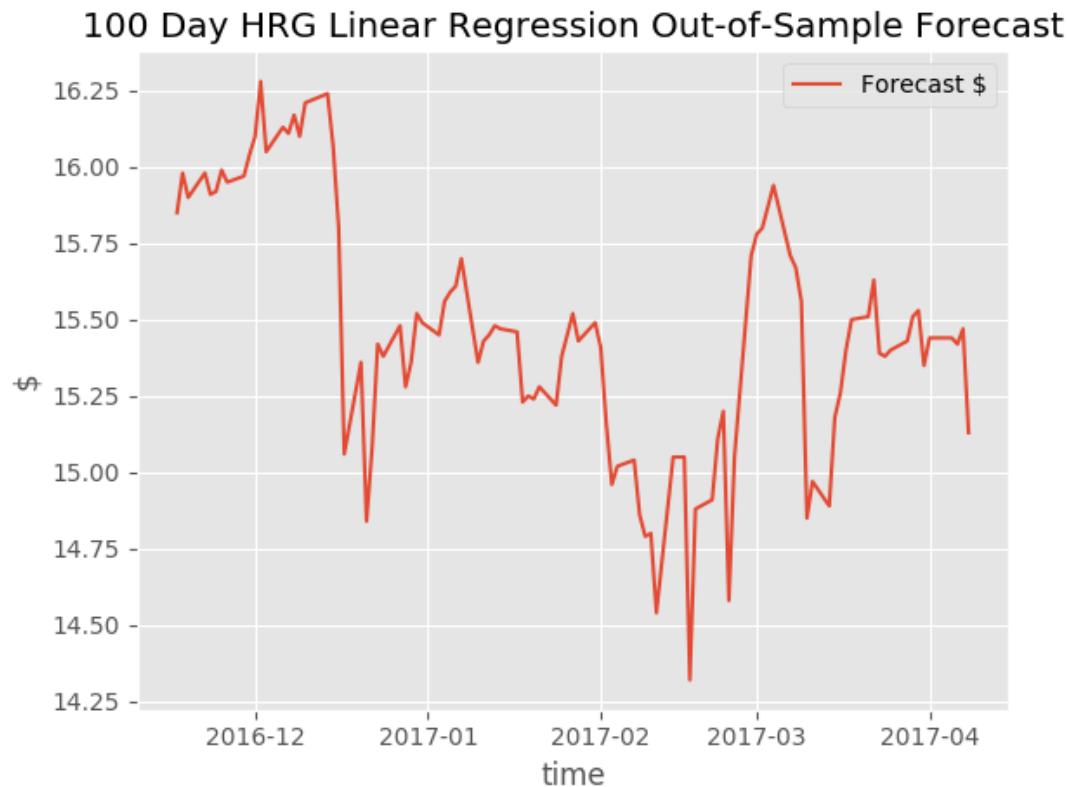


FIGURE 4.149: 100 day HRG Linear Regression out-of-sample forecast

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

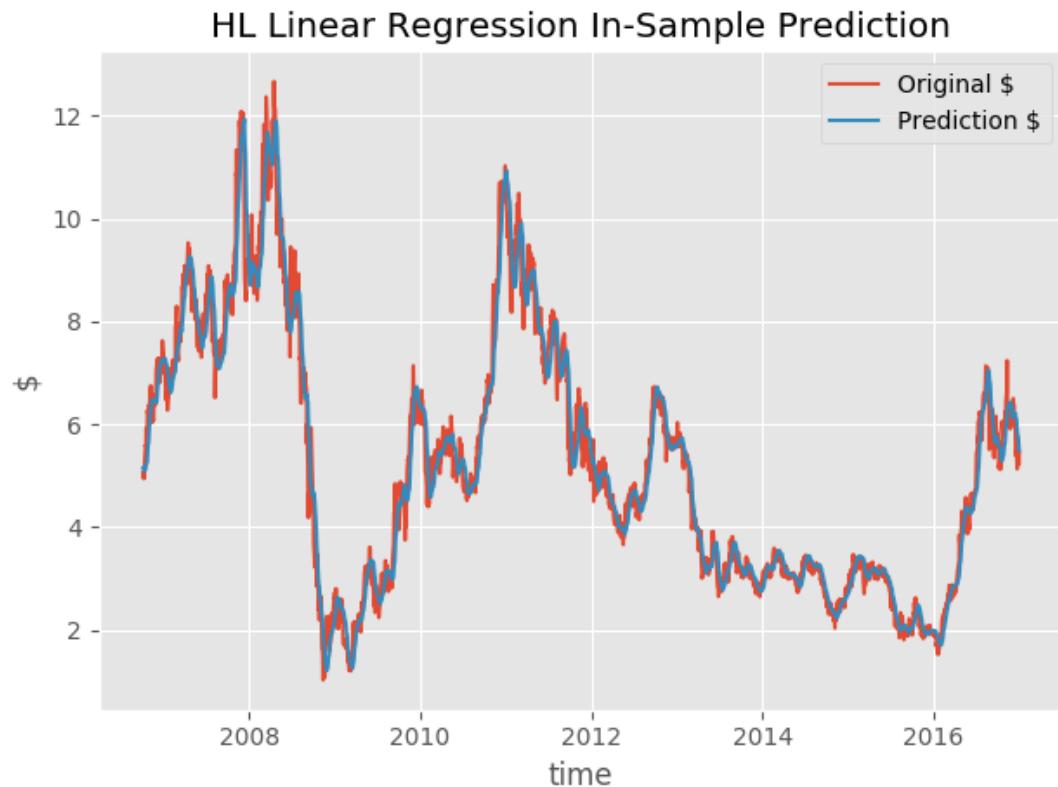


FIGURE 4.150: HL Linear Regression in-sample prediction

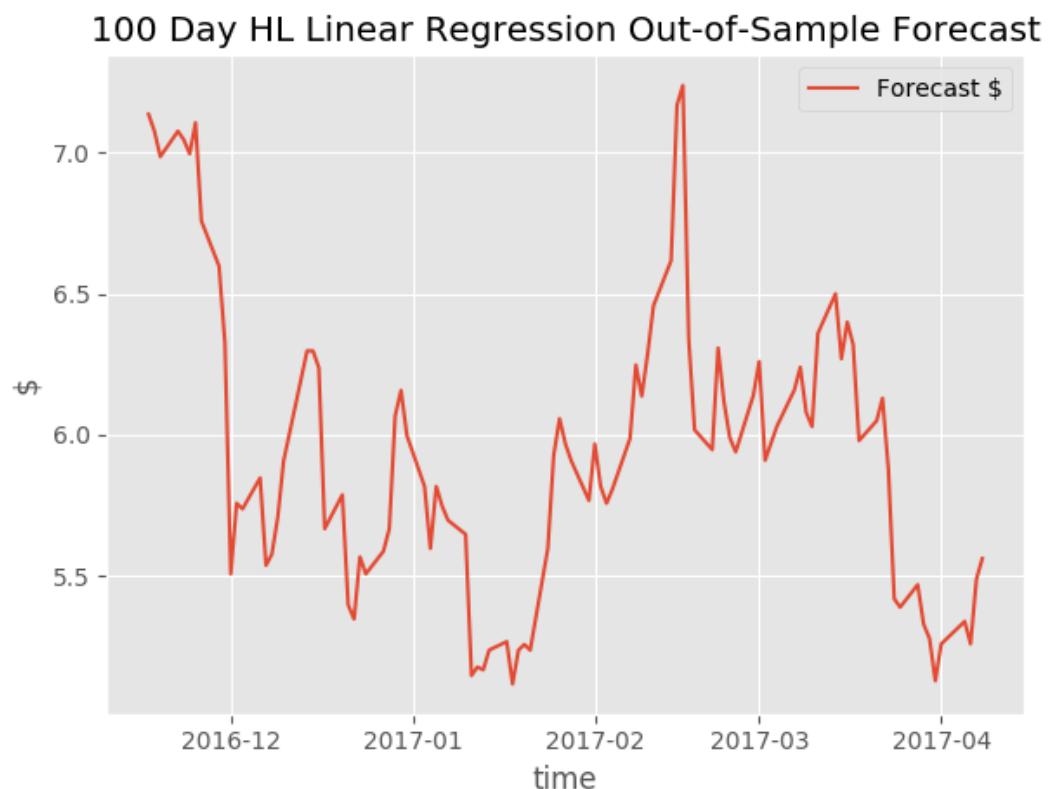


FIGURE 4.151: 100 day HL Linear Regression out-of-sample forecast

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.

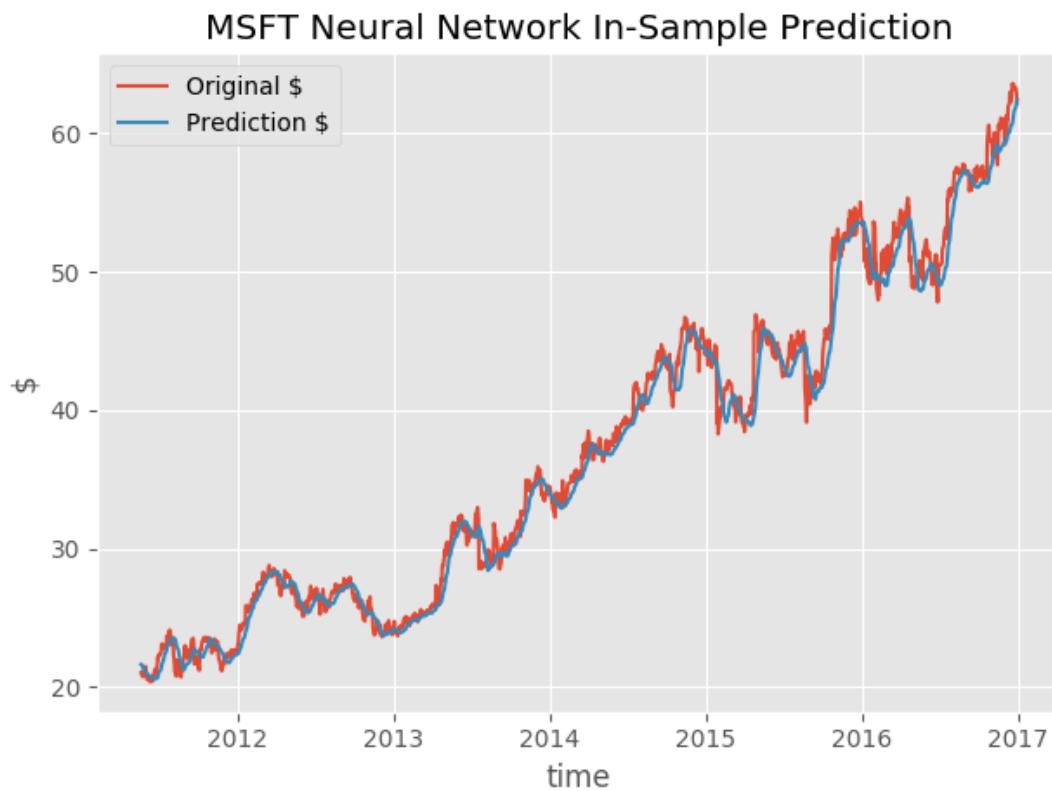


FIGURE 4.152: MSFT Neural Network in-sample prediction

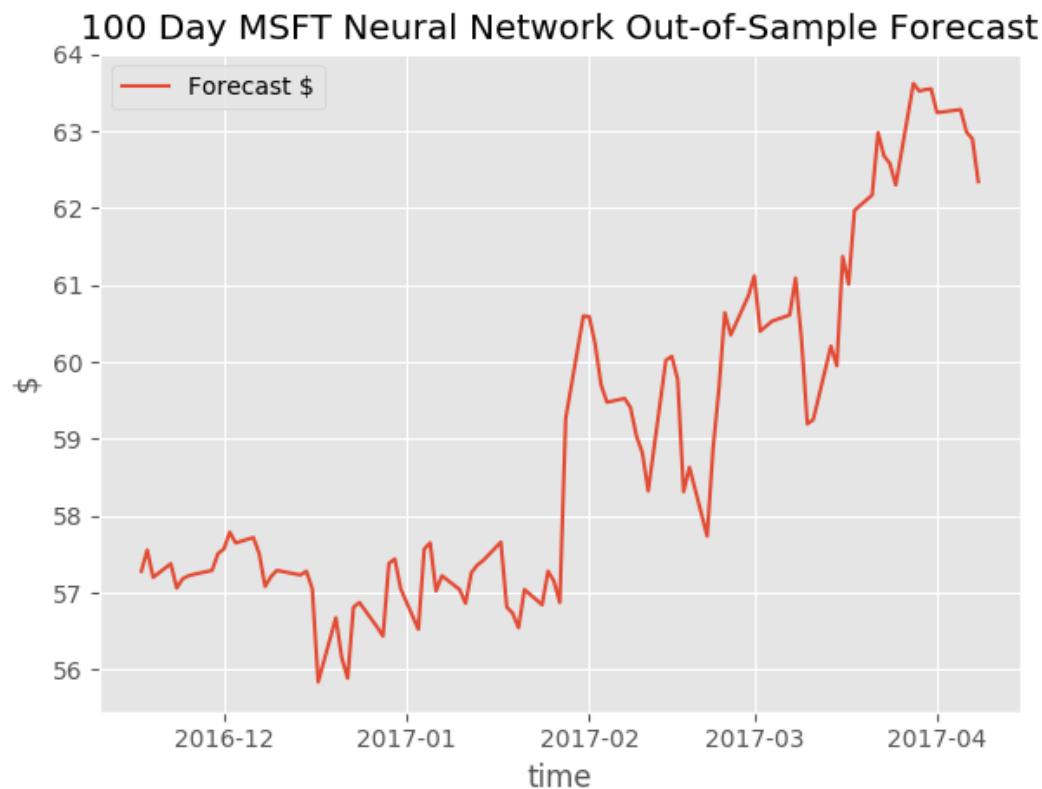
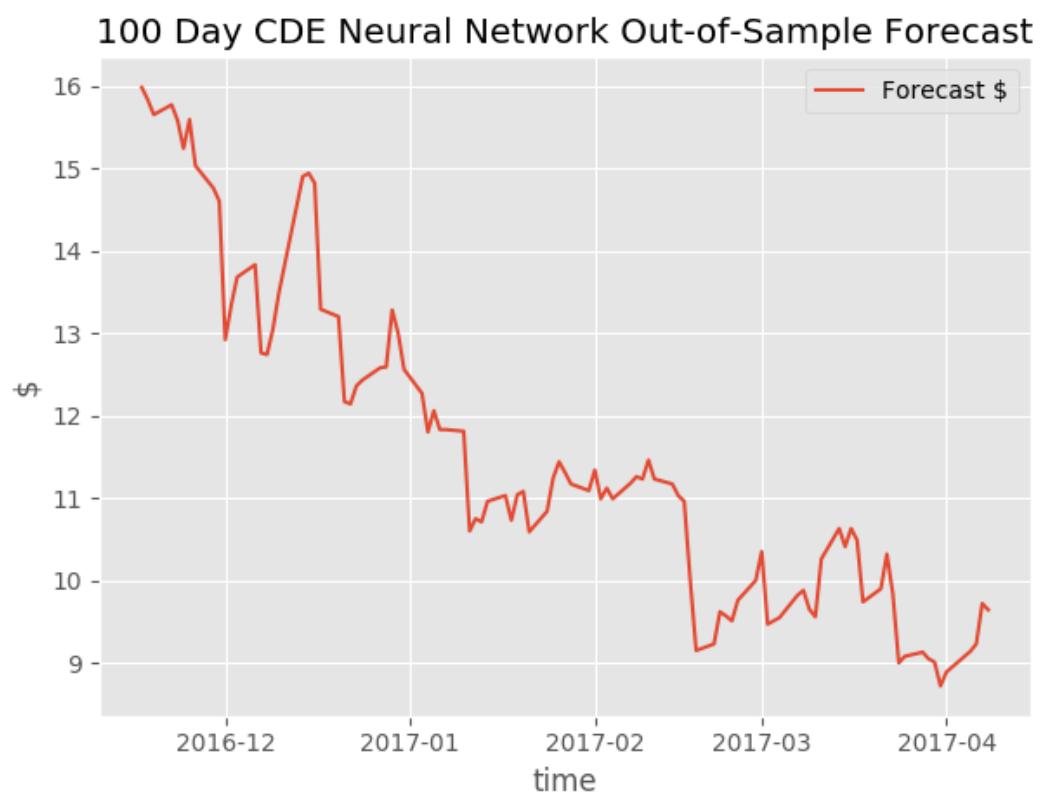
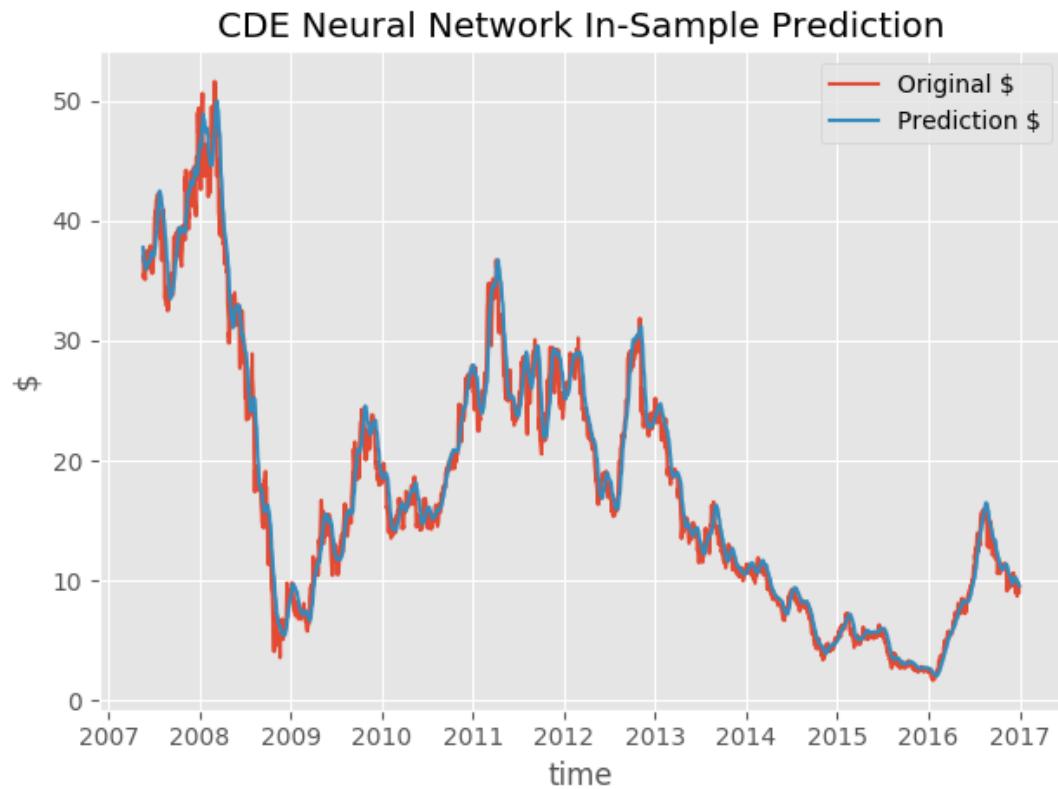


FIGURE 4.153: 100 day MSFT Neural Network out-of-sample forecast

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



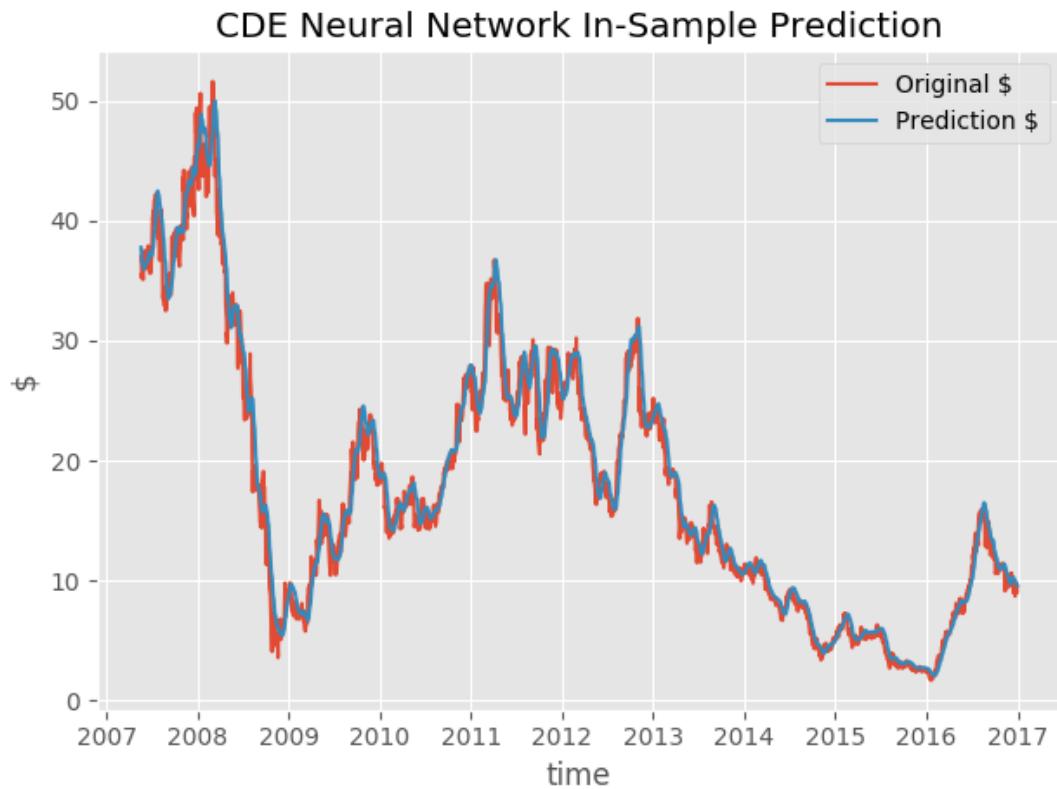


FIGURE 4.154: CDE Neural Network in-sample prediction

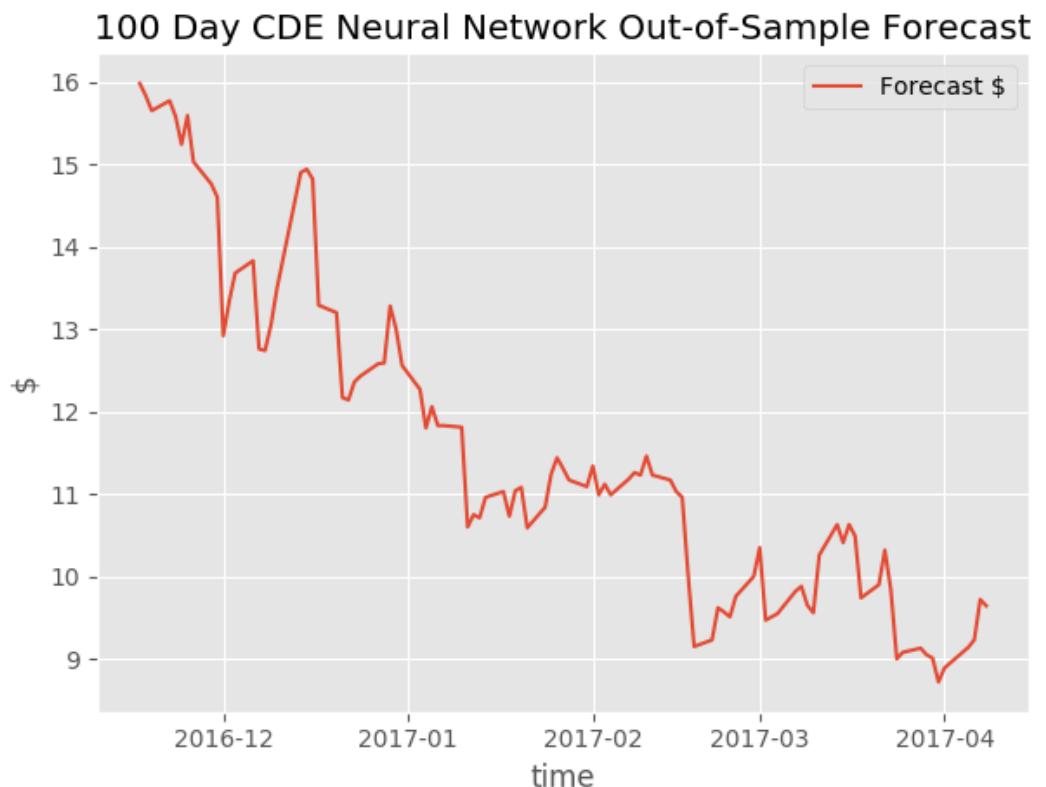


FIGURE 4.155: 100 day CDE Neural Network out-of-sample forecast

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

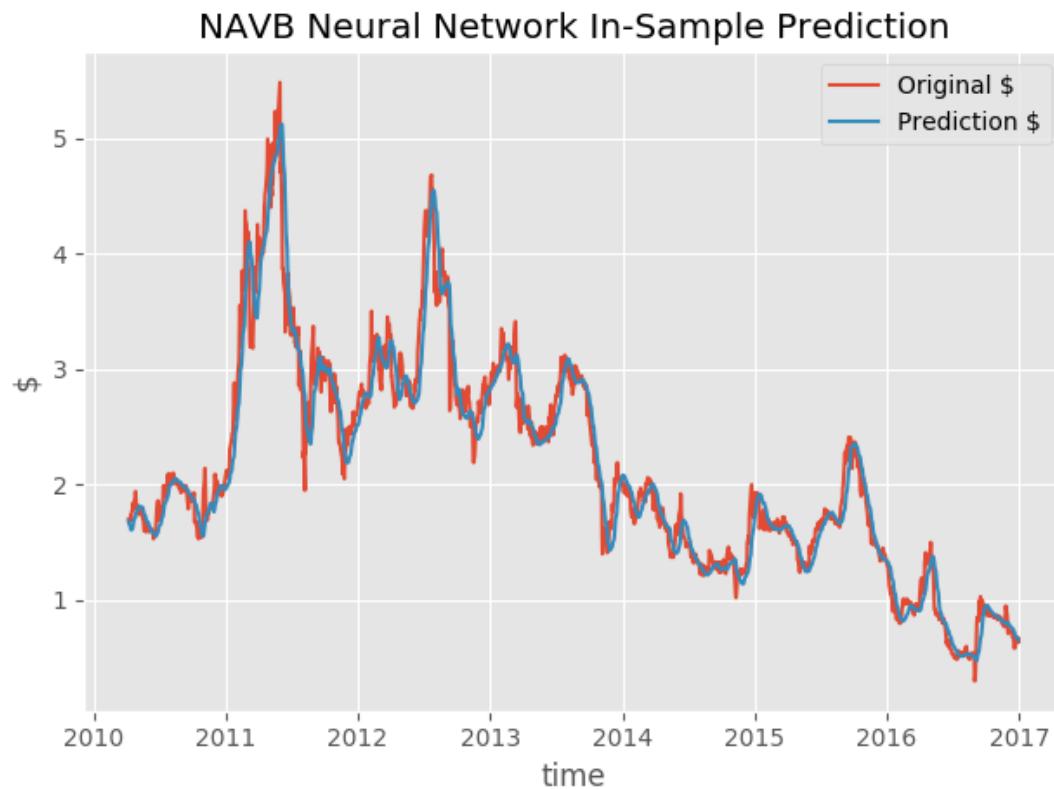


FIGURE 4.156: NAVB Neural Network in-sample prediction

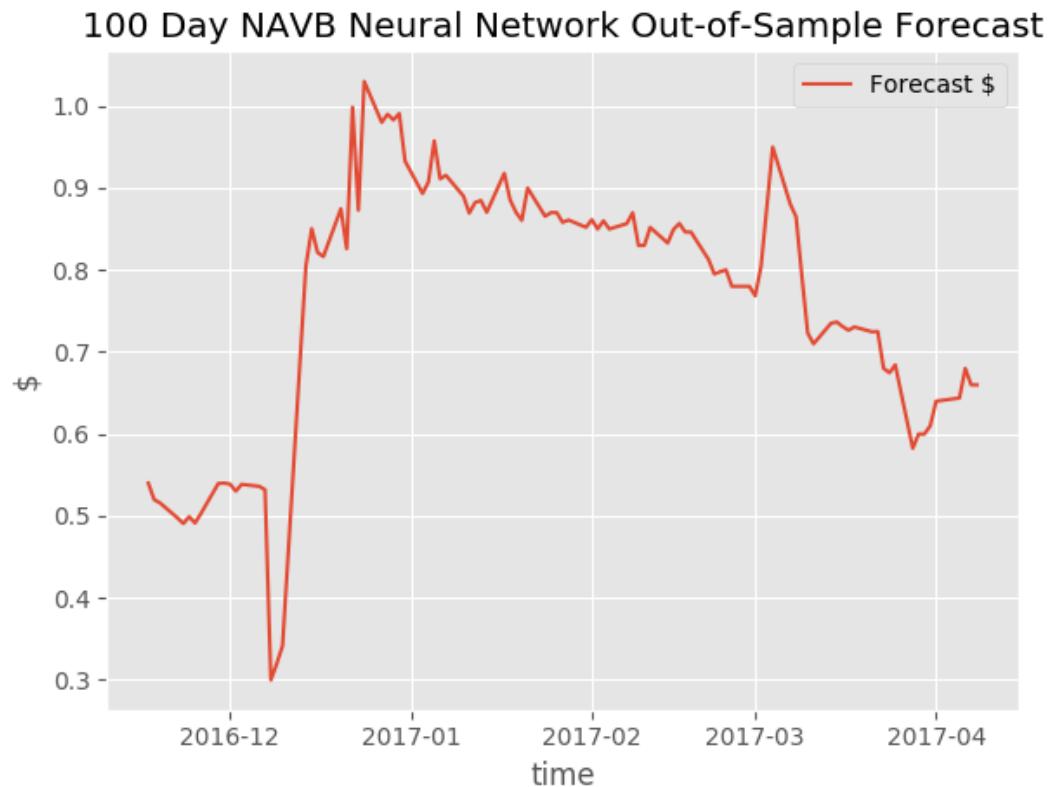


FIGURE 4.157: 100 day NAVB Neural Network out-of-sample forecast

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

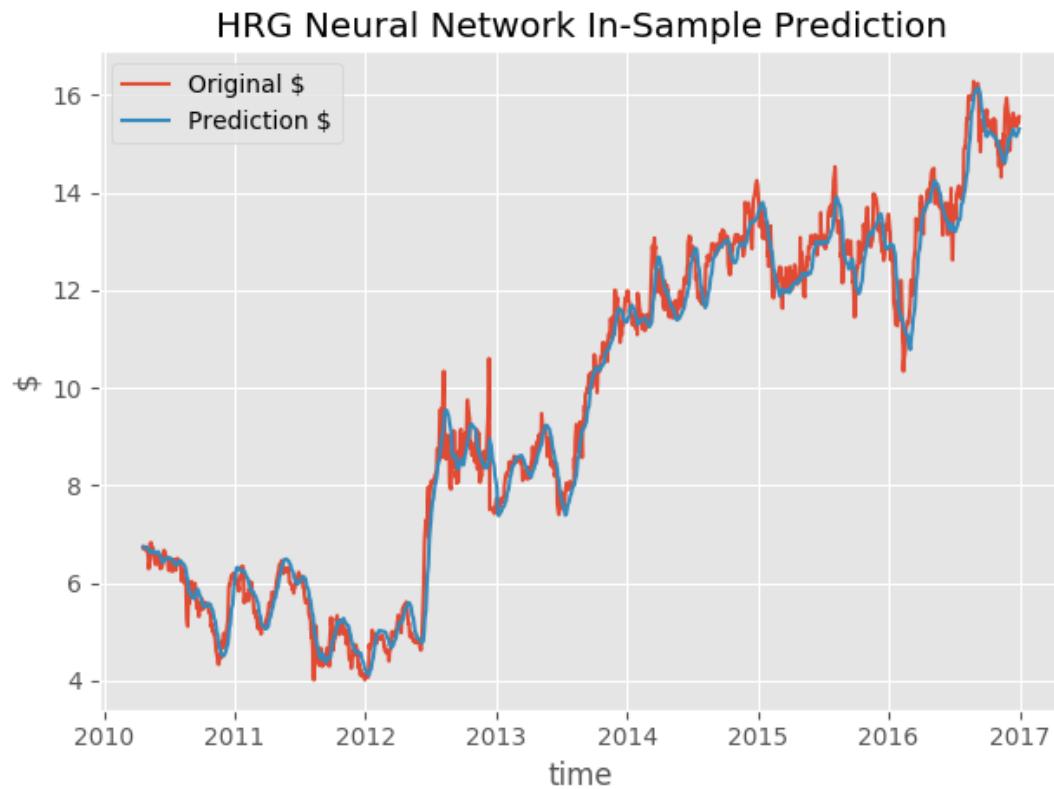


FIGURE 4.158: HRG Neural Network in-sample prediction

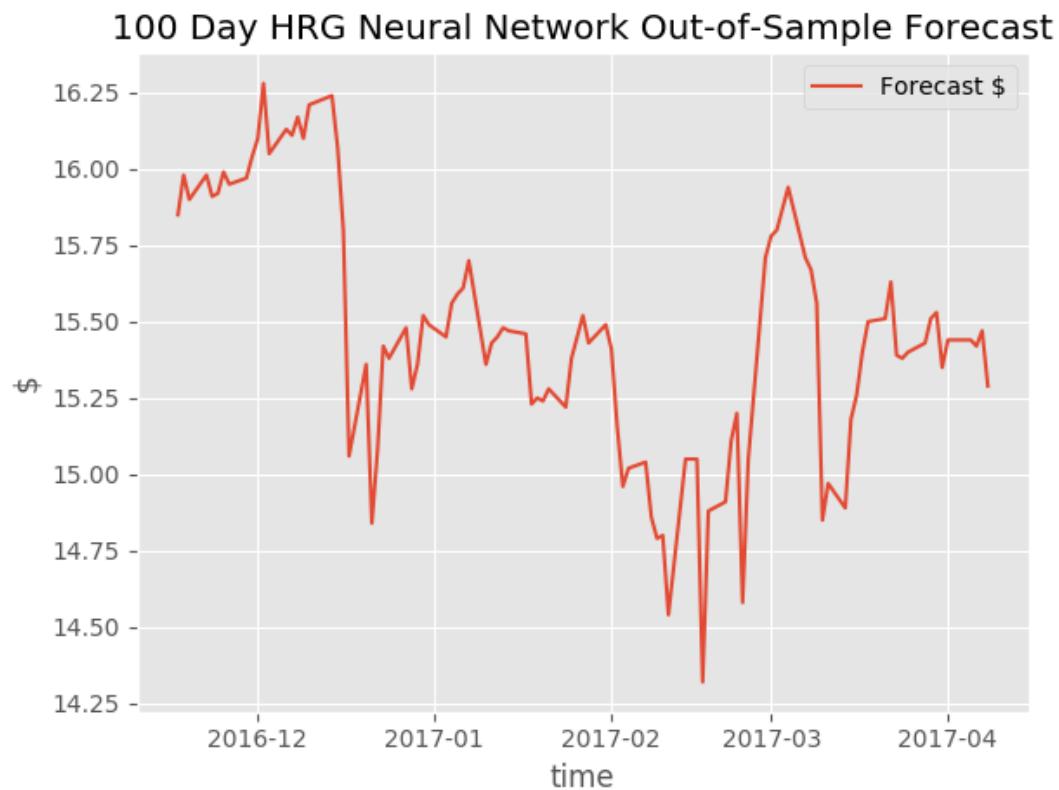


FIGURE 4.159: 100 day HRG Neural Network out-of-sample forecast

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

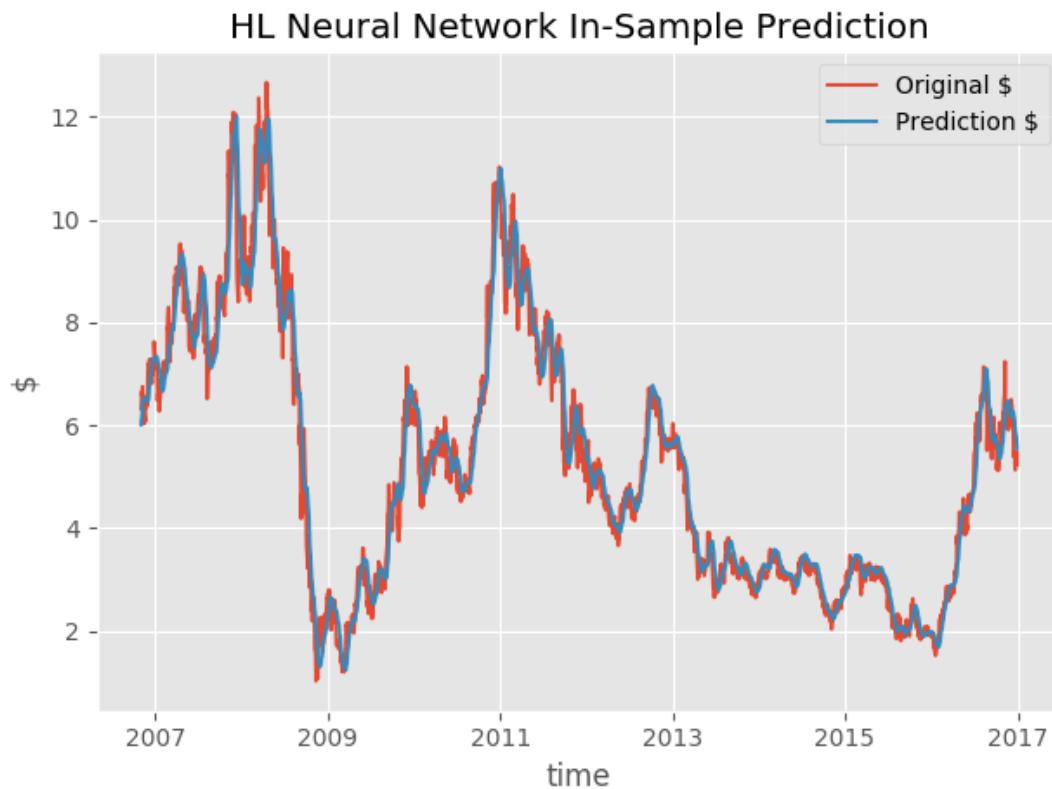


FIGURE 4.160: HL Neural Network in-sample prediction

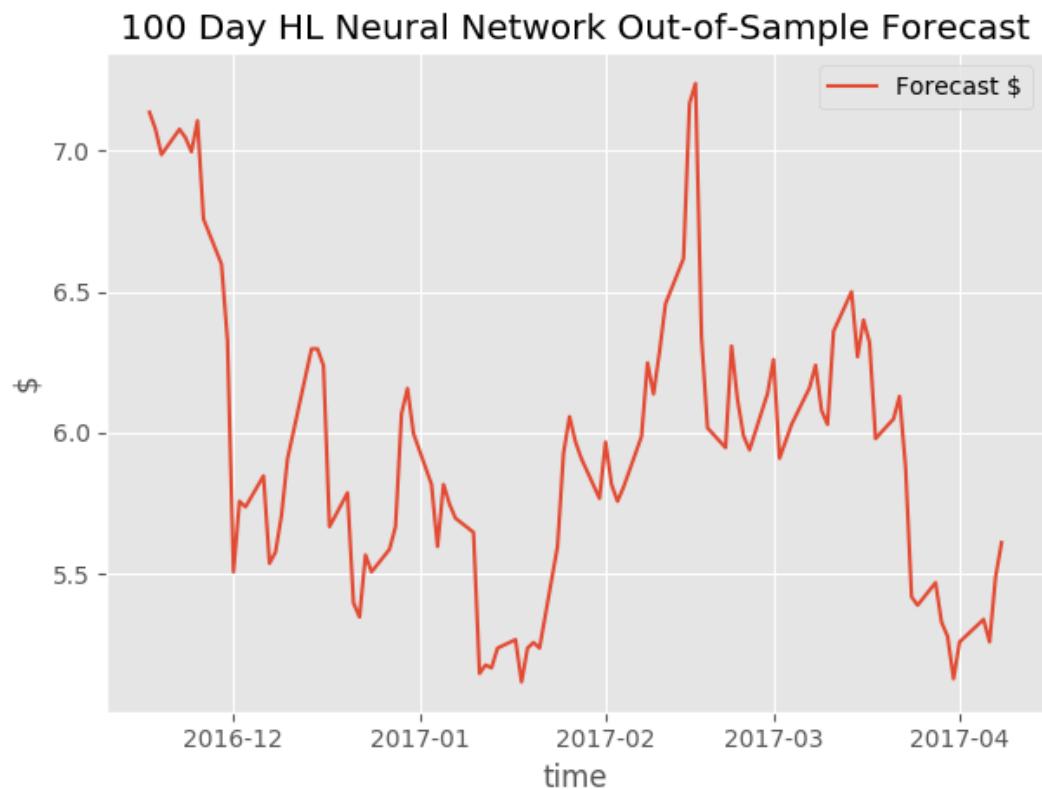


FIGURE 4.161: 100 day HL Neural Network out-of-sample forecast

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.

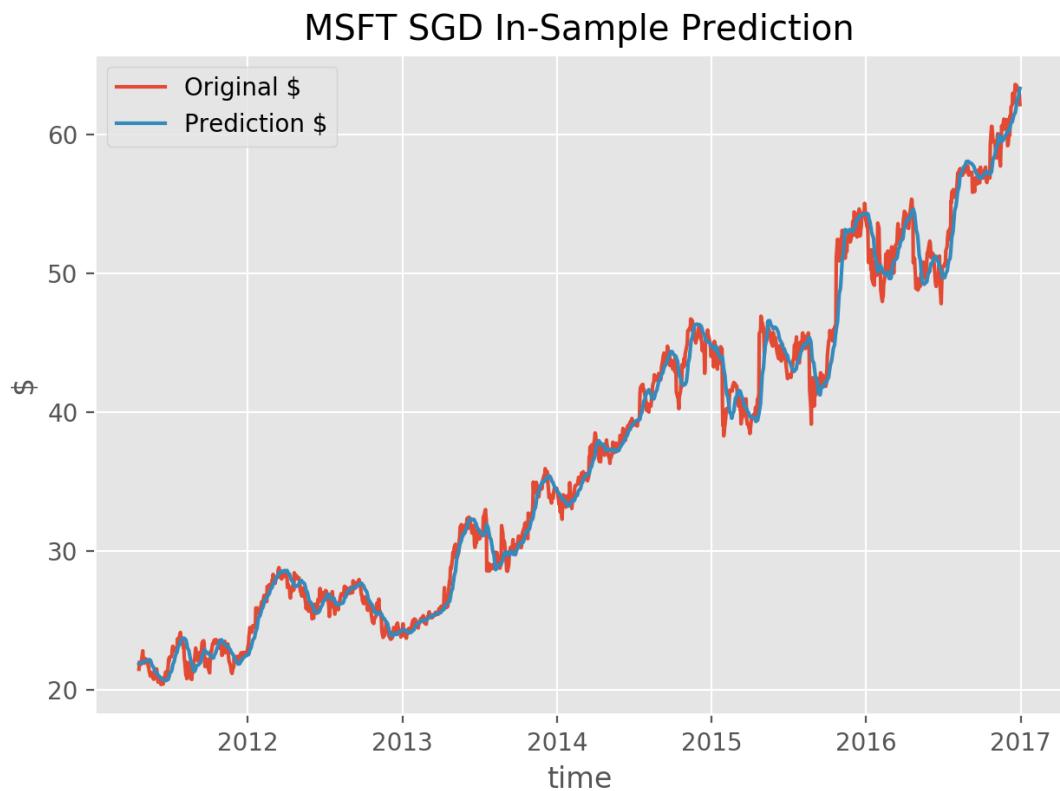


FIGURE 4.162: MSFT SGD in-sample prediction

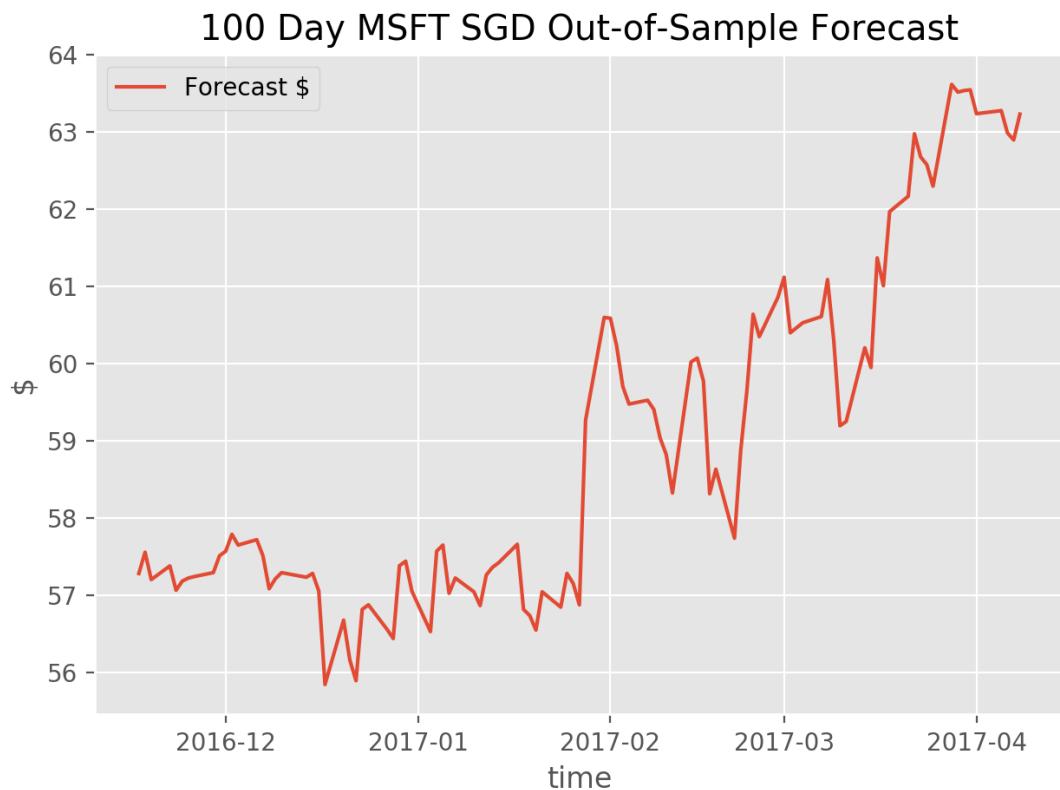


FIGURE 4.163: 100 day MSFT SGD out-of-sample forecast

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

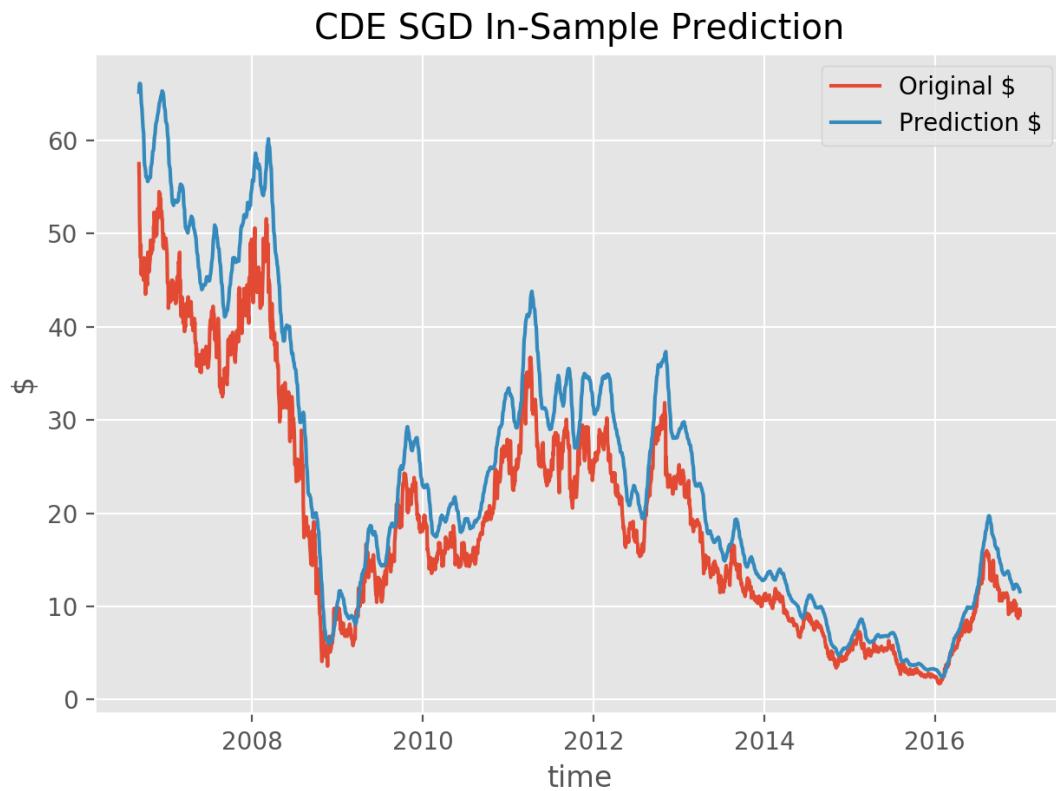


FIGURE 4.164: CDE SGD in-sample prediction

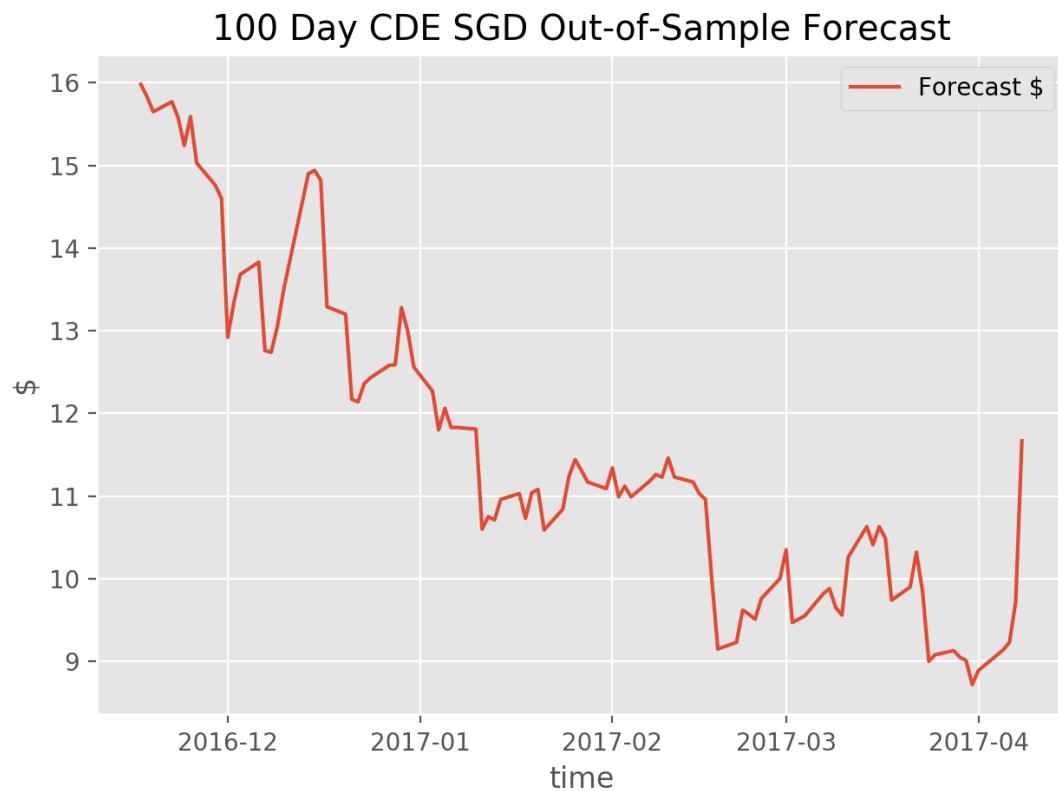


FIGURE 4.165: 100 day CDE SGD out-of-sample forecast

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

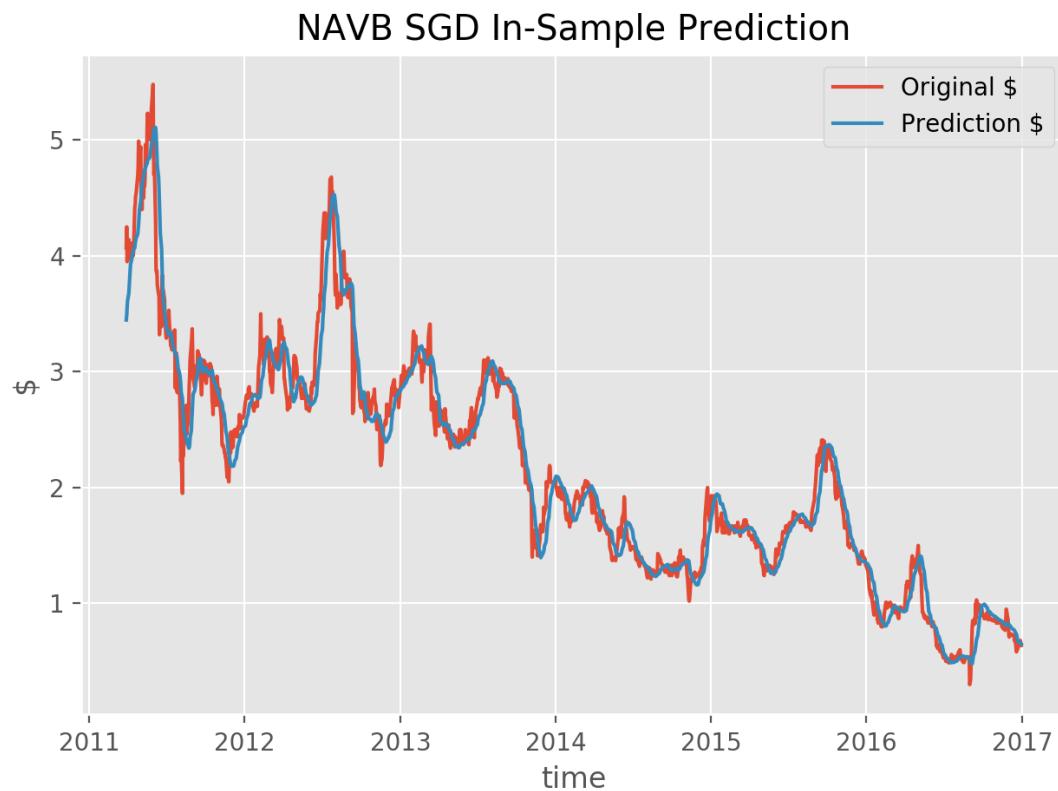


FIGURE 4.166: NAVB SGD in-sample prediction

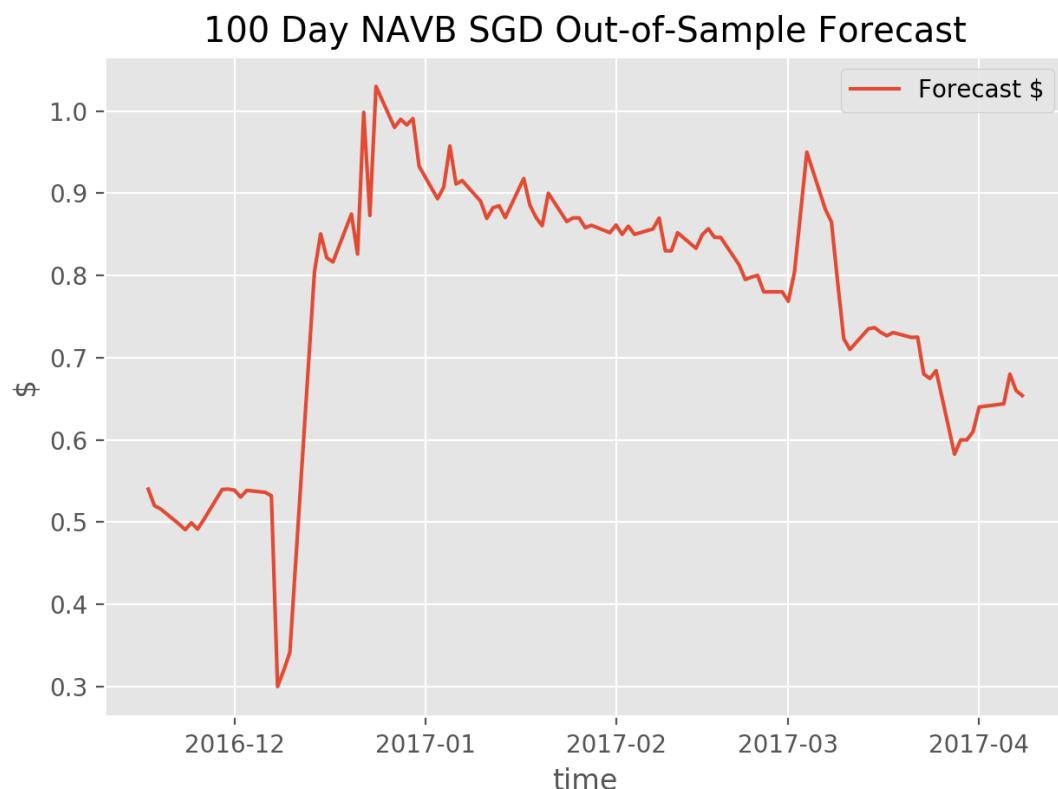


FIGURE 4.167: 100 day NAVB SGD out-of-sample forecast

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

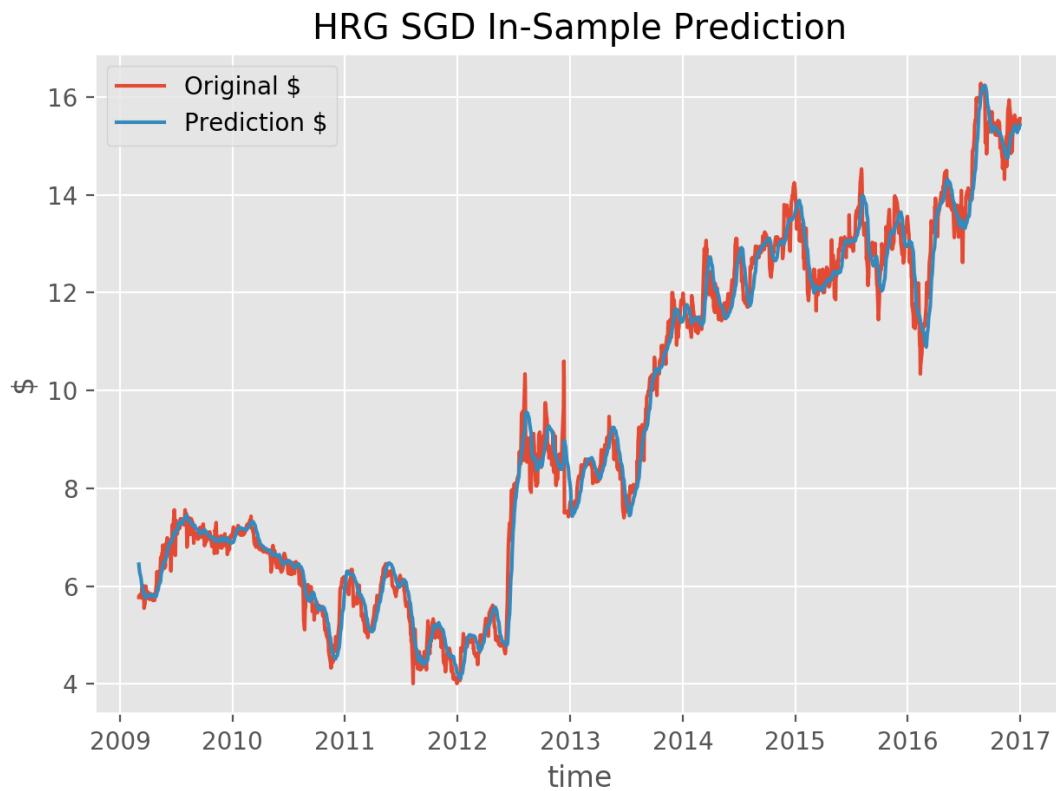


FIGURE 4.168: HRG SGD in-sample prediction

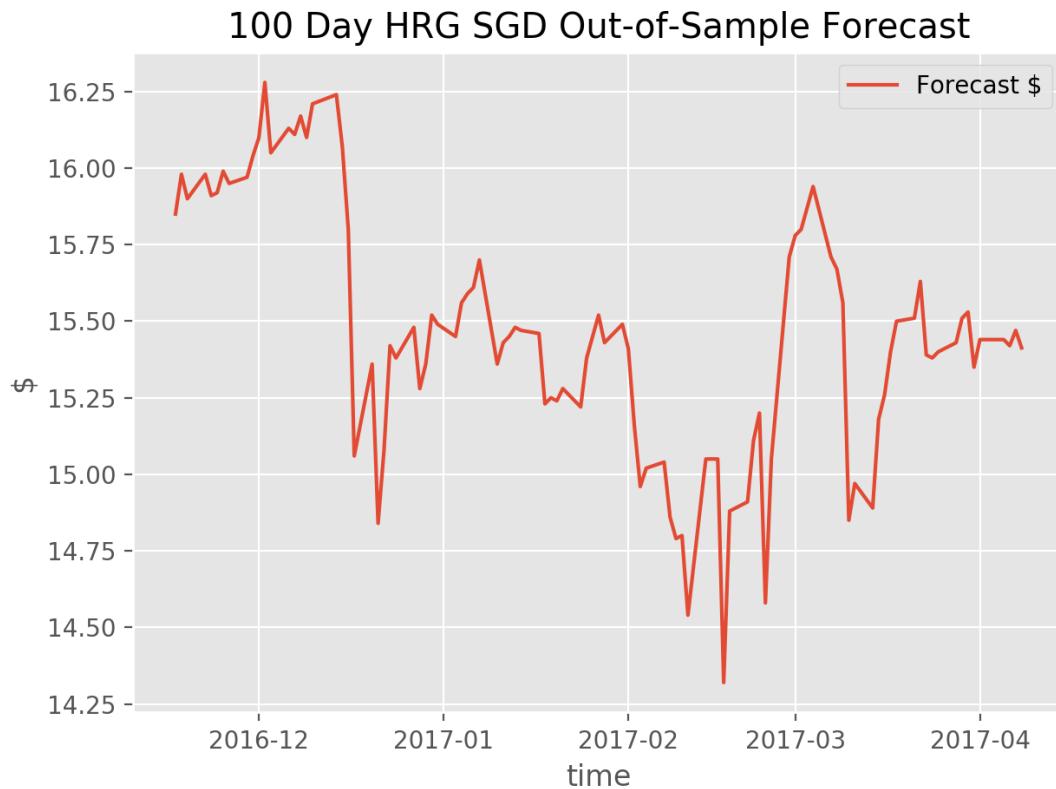


FIGURE 4.169: 100 day HRG SGD out-of-sample forecast

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

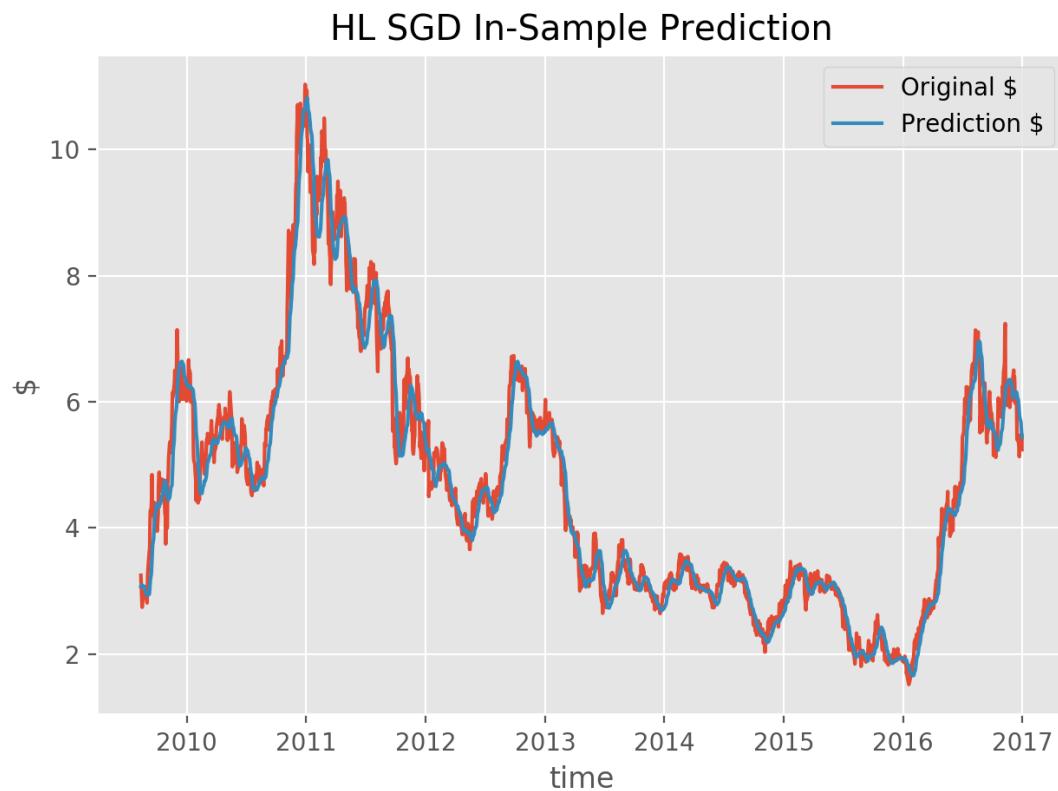


FIGURE 4.170: HL SGD in-sample prediction

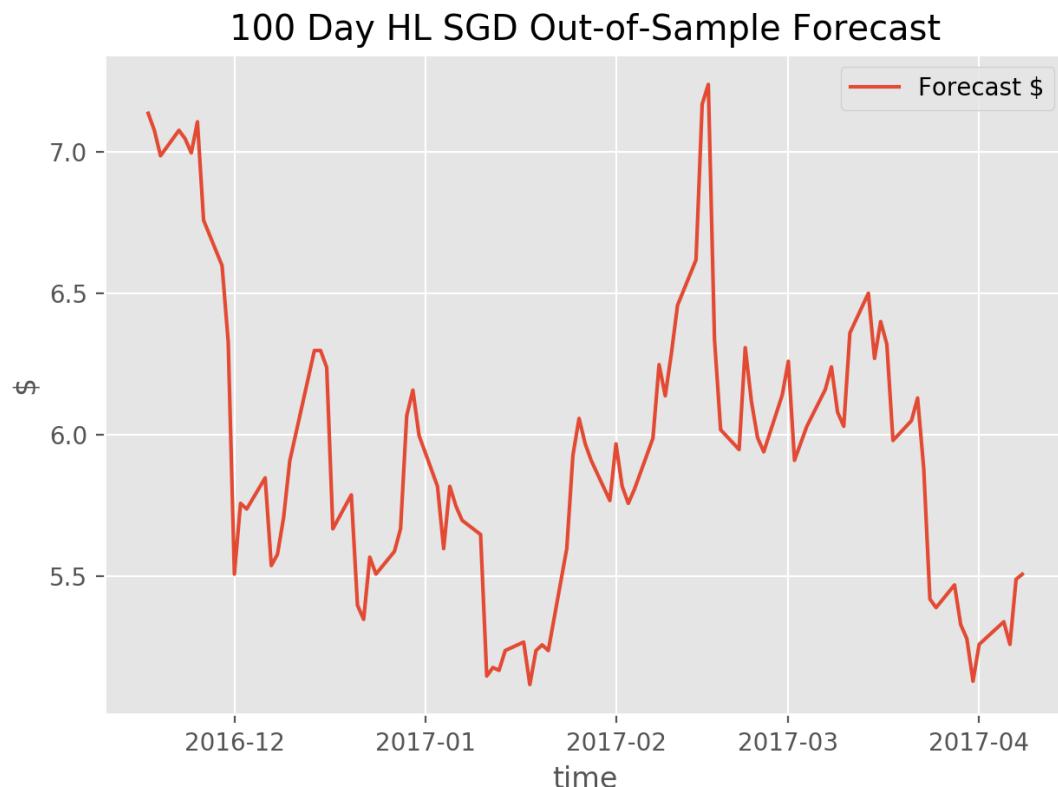


FIGURE 4.171: 100 day HL SGD out-of-sample forecast

4.3 Bayesian Statistics

4.3.1 Metropolis-Hastings

MSFT scored sharpe ratios of 0.439 for the original returns, and -4.796 for the predicted returns in the in-sample test.

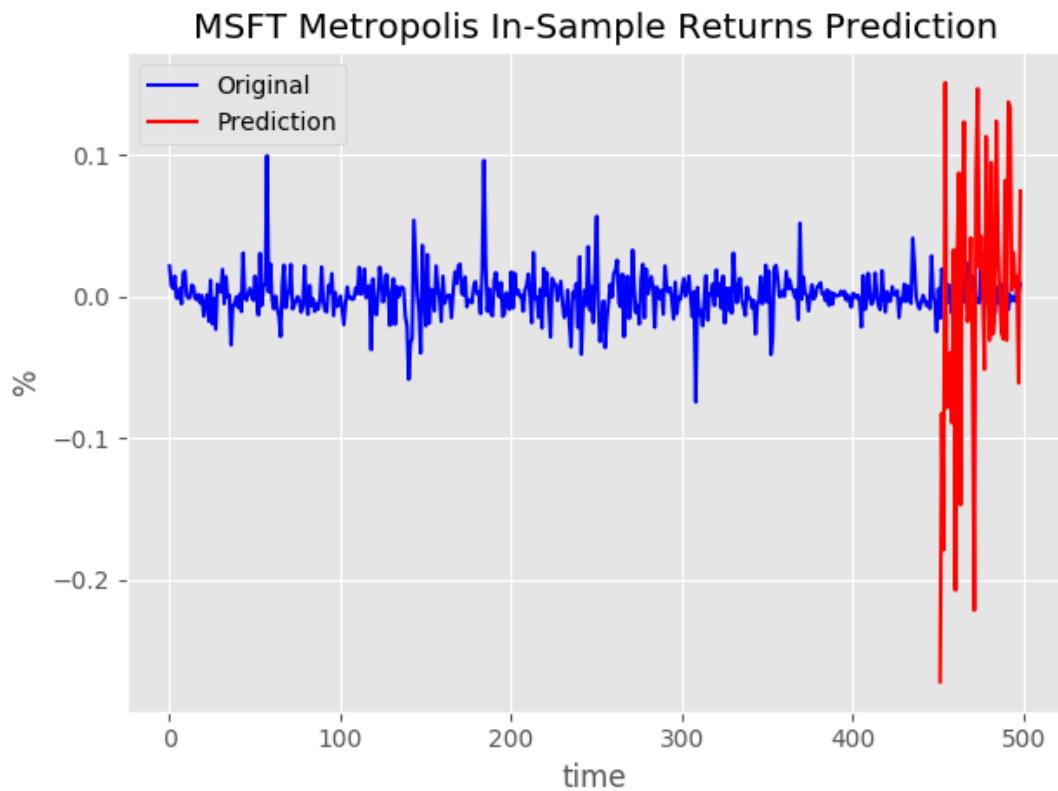


FIGURE 4.172: MSFT Metropolis-Hastings in-sample prediction

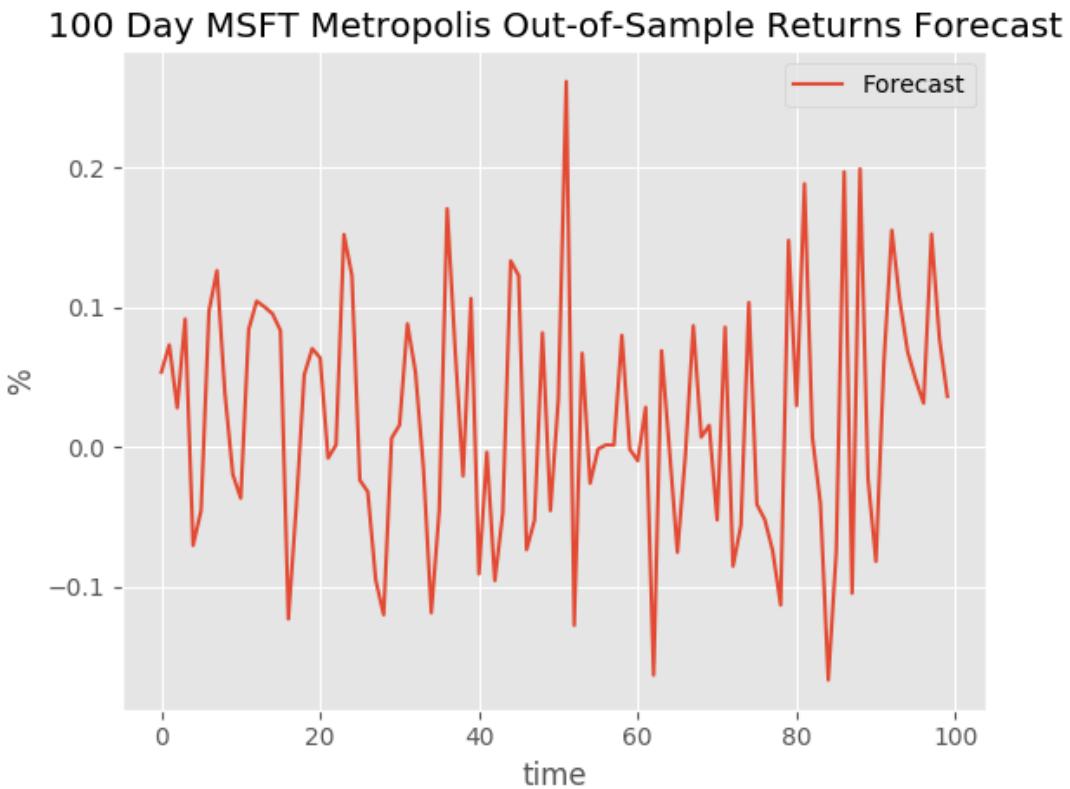


FIGURE 4.173: 100 day MSFT Metropolis-Hastings out-of-sample forecast

CDE scored sharpe ratios of 2.028 for the original returns, and 4.075 for the predicted returns in the in-sample test.

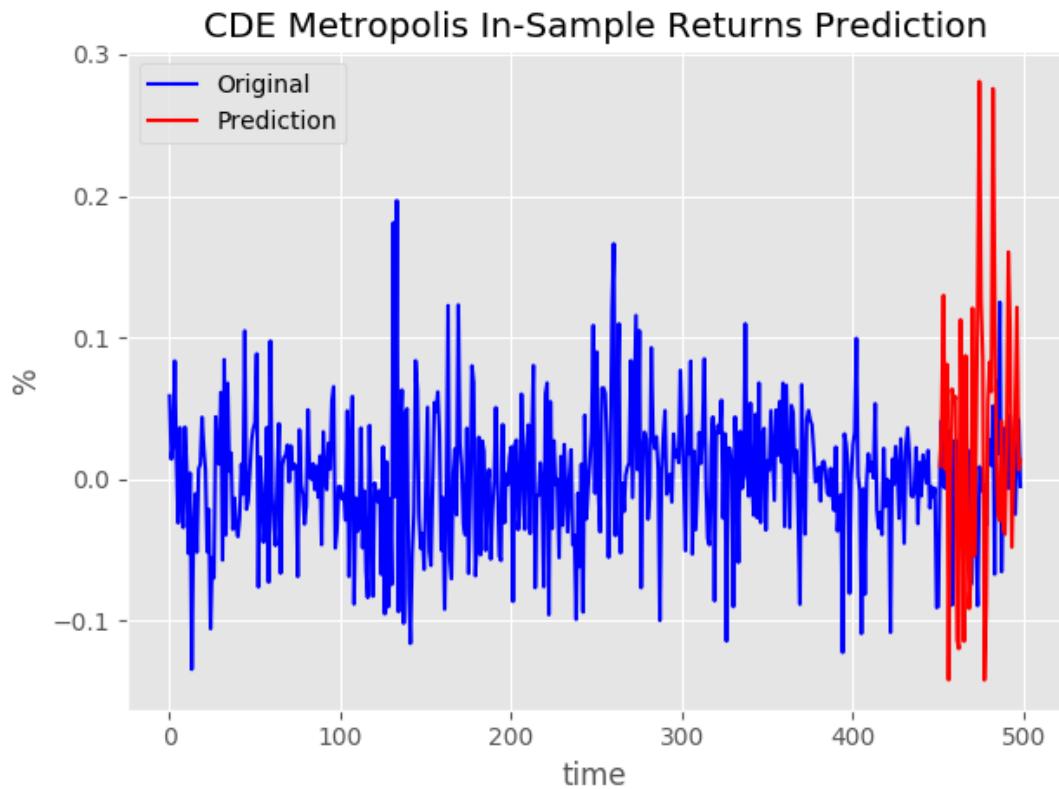


FIGURE 4.174: CDE Metropolis-Hastings in-sample prediction

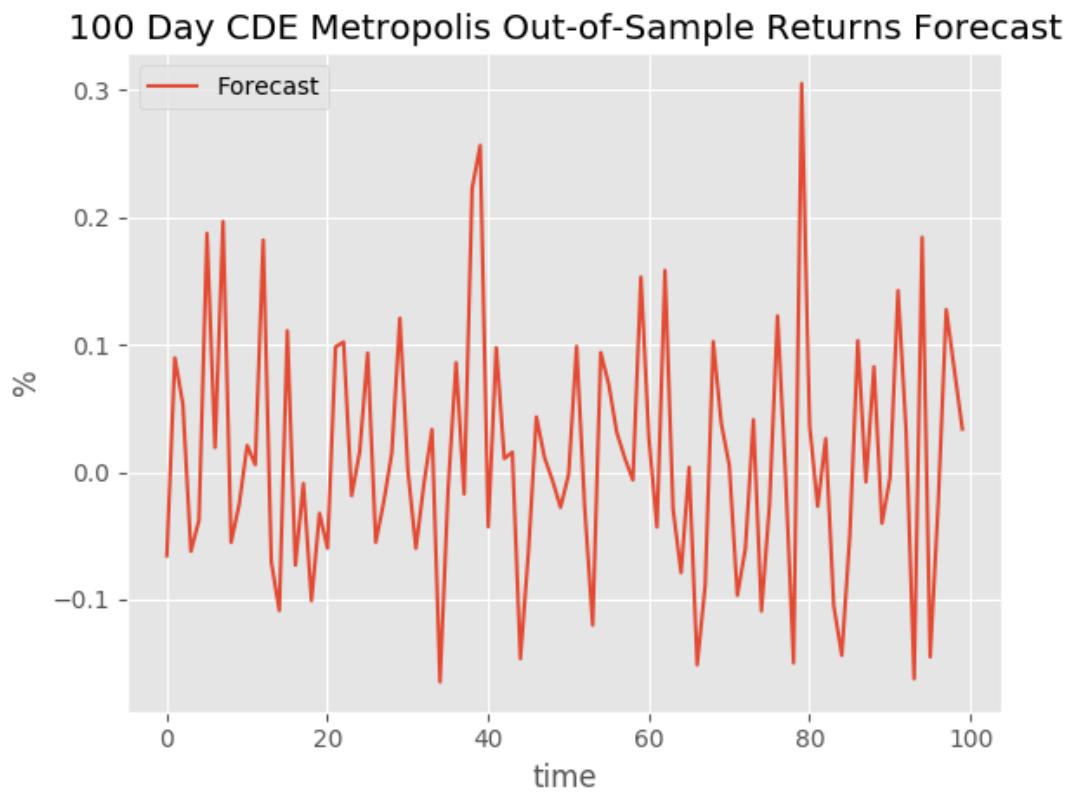


FIGURE 4.175: 100 day CDE Metropolis-Hastings out-of-sample forecast

NAVB scored sharpe ratios of -2.800 for the original returns, and 2.998 for the predicted returns in the in-sample test.

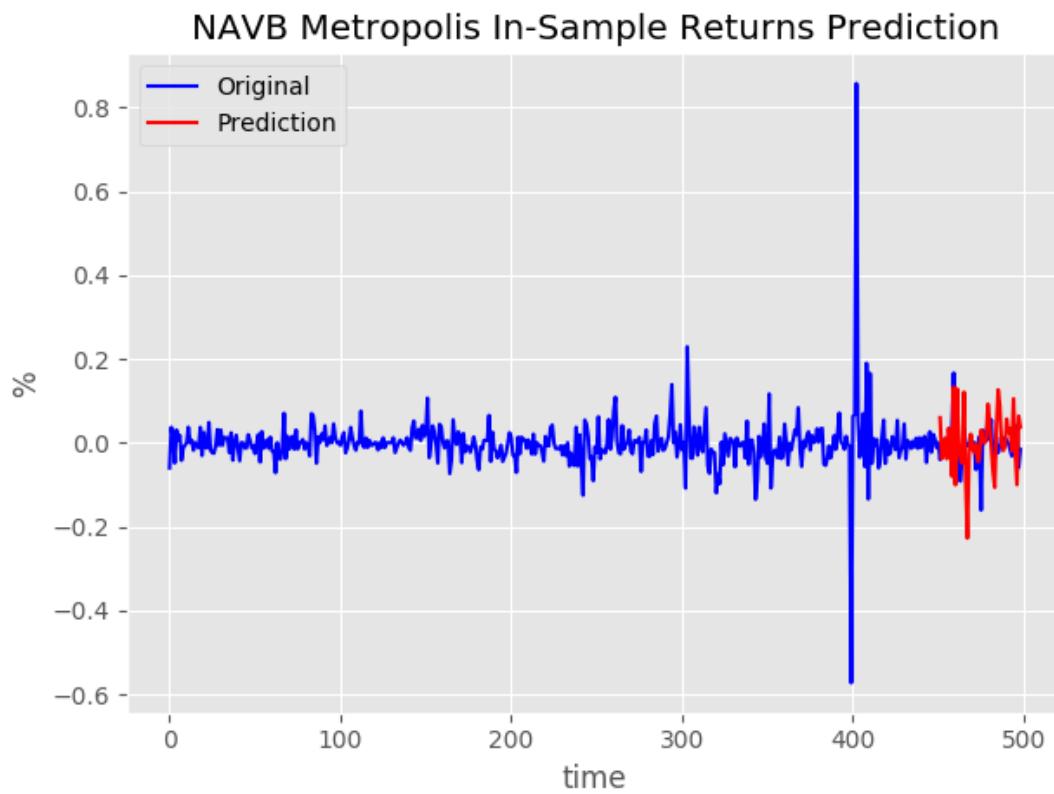


FIGURE 4.176: NAVB Metropolis-Hastings in-sample prediction

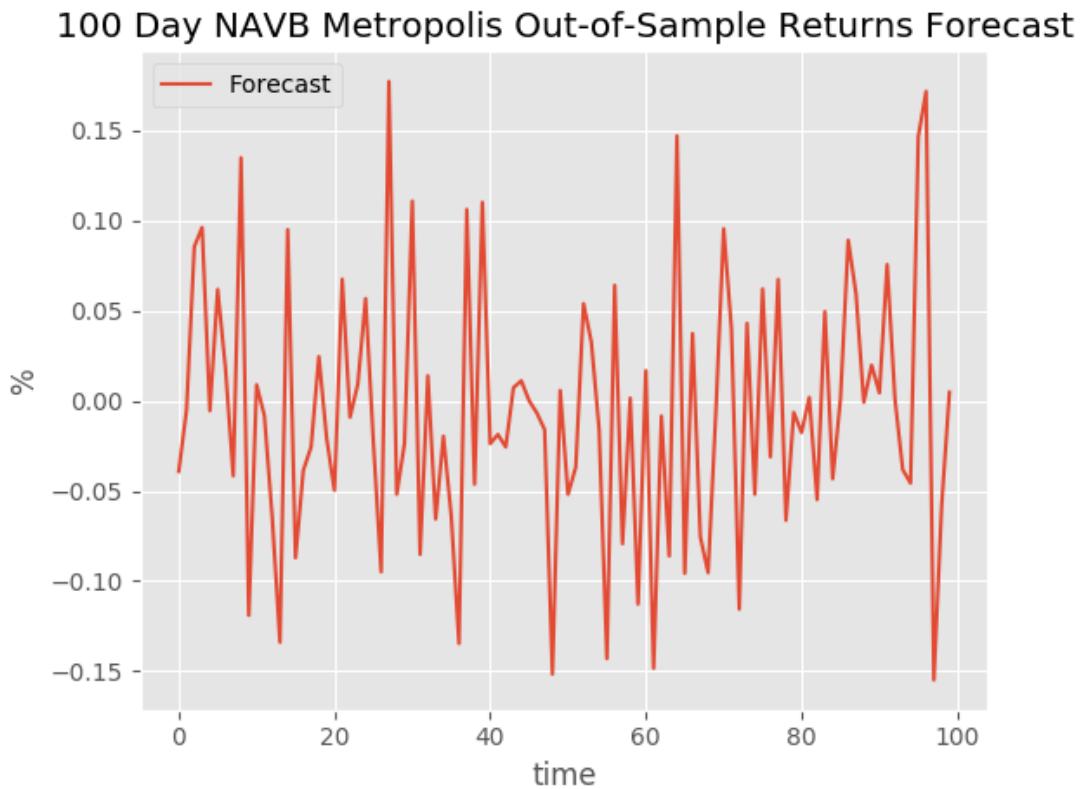


FIGURE 4.177: 100 day NAVB Metropolis-Hastings out-of-sample forecast

HRG scored sharpe ratios of 1.362 for the original returns, and -2.173 for the predicted returns in the in-sample test.

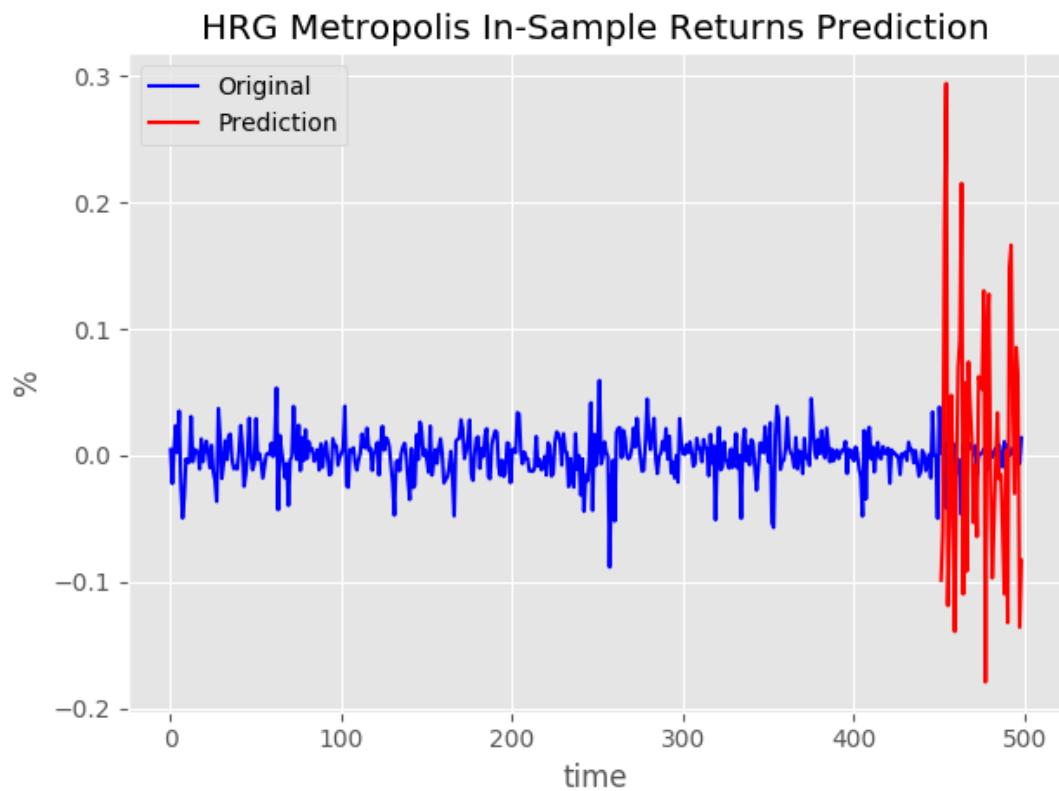


FIGURE 4.178: HRG Metropolis-Hastings in-sample prediction

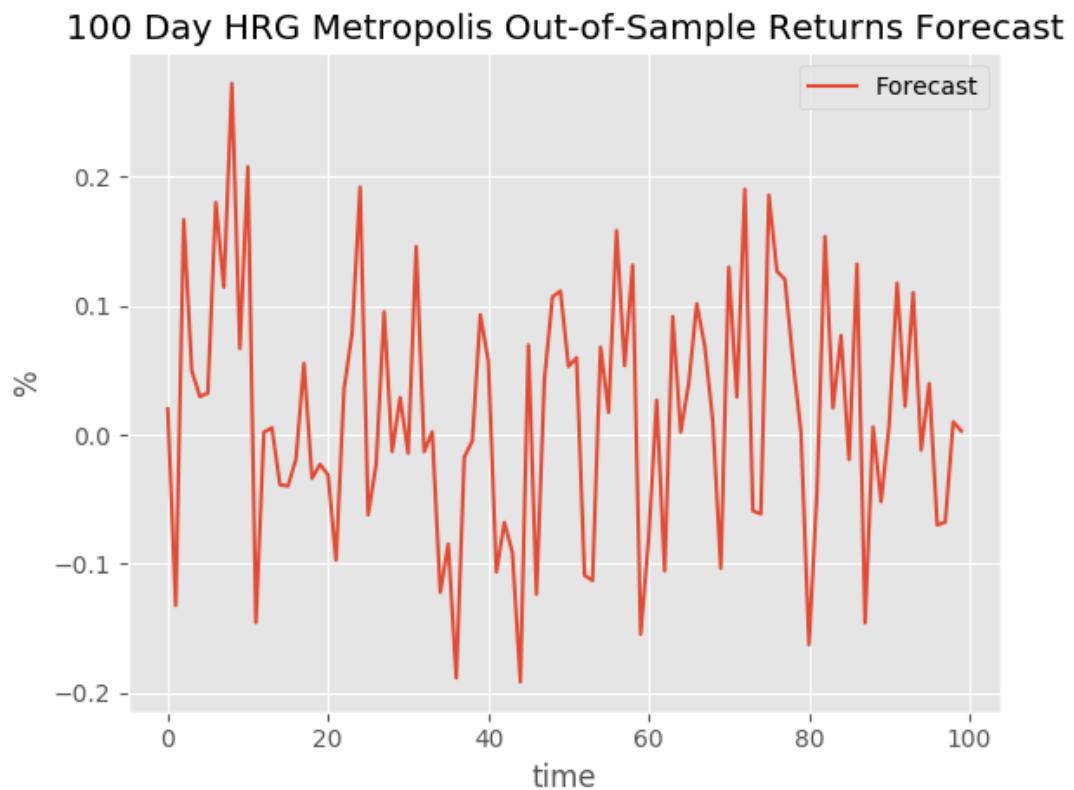


FIGURE 4.179: 100 day HRG Metropolis-Hastings out-of-sample forecast

HL scored sharpe ratios of 0.439 for the original returns, and -2.283 for the predicted returns in the in-sample test.

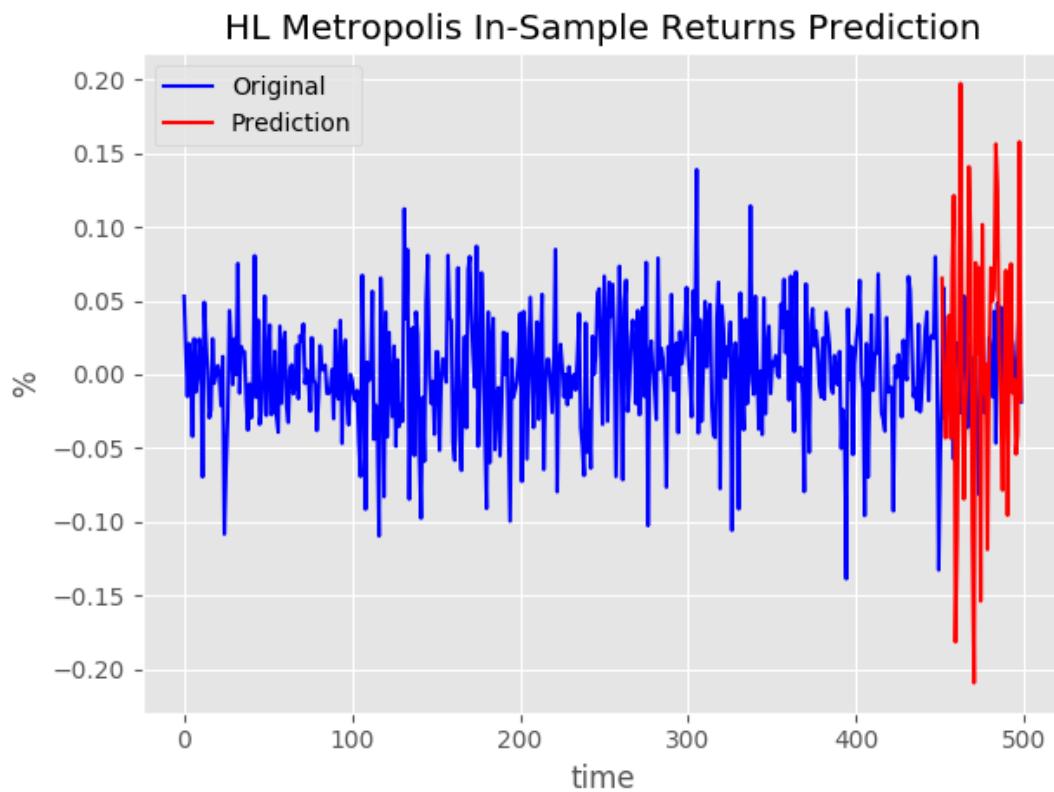


FIGURE 4.180: HL Metropolis-Hastings in-sample prediction

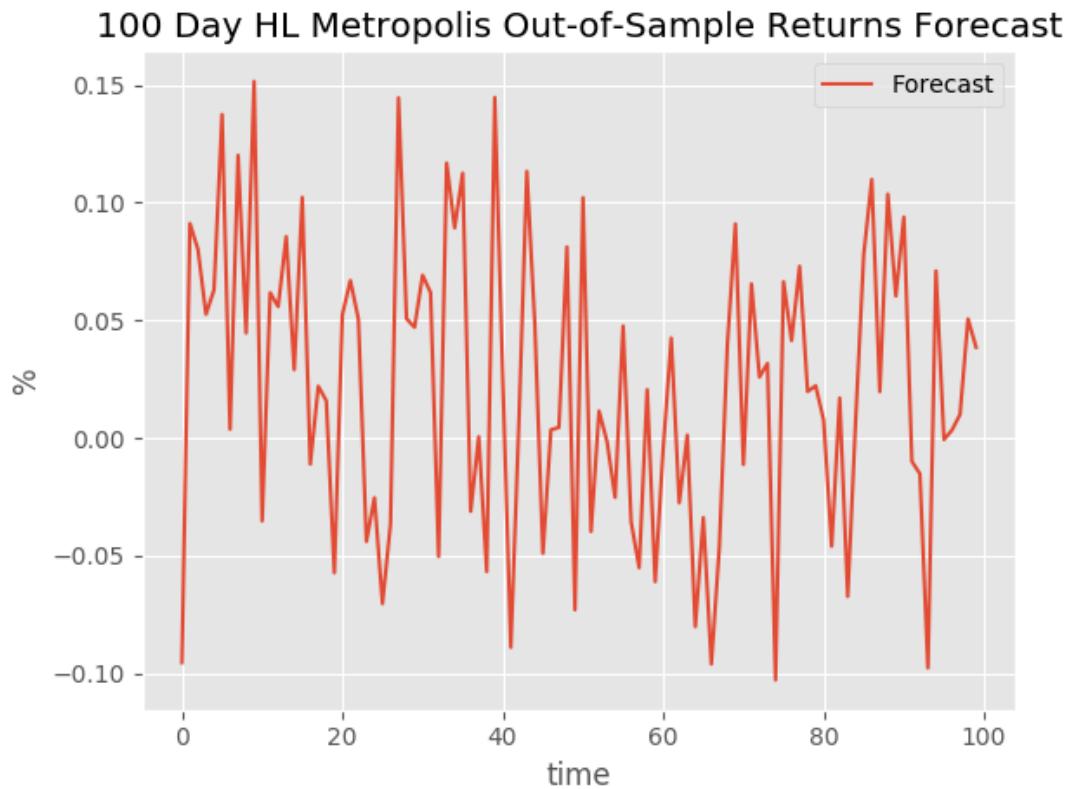


FIGURE 4.181: 100 day HL Metropolis-Hastings out-of-sample forecast

4.3.2 No-U-Turn Sampler (NUTS)

MSFT scored sharpe ratios of 2.499 for the original returns, and -1.355 for the predicted returns in the in-sample test.

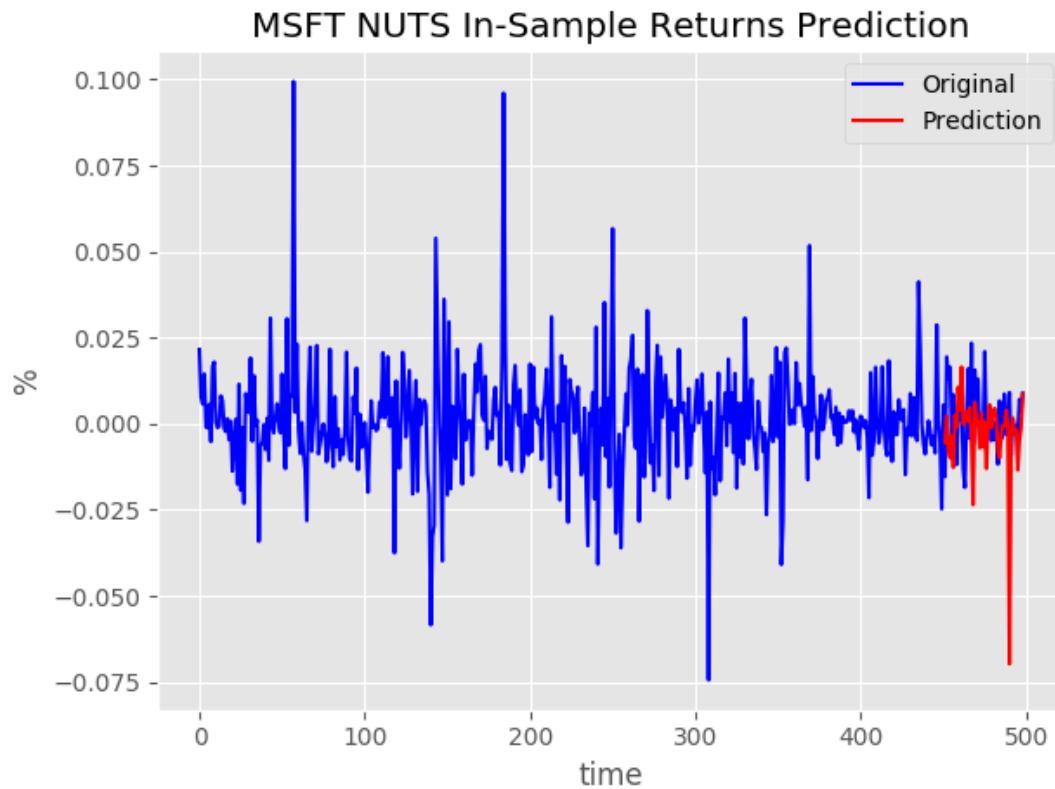


FIGURE 4.182: MSFT NUTS in-sample prediction

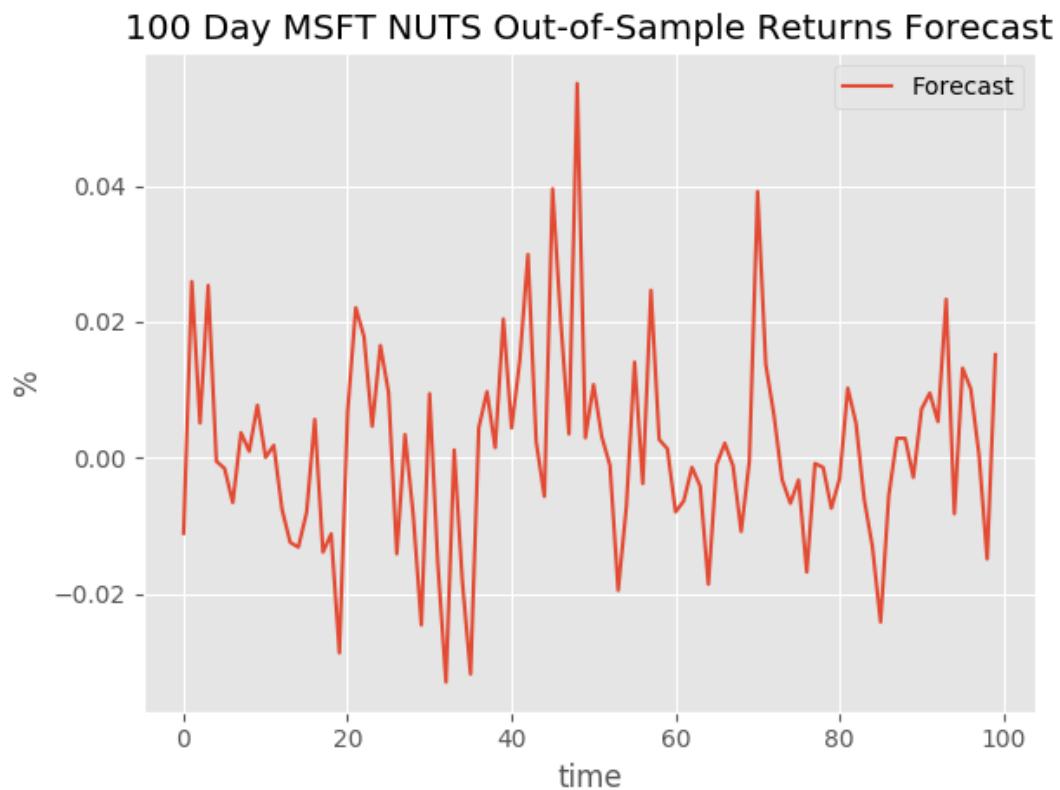


FIGURE 4.183: 100 day MSFT NUTS out-of-sample forecast

CDE scored sharpe ratios of 2.028 for the original returns, and 3.066 for the predicted returns in the in-sample test.

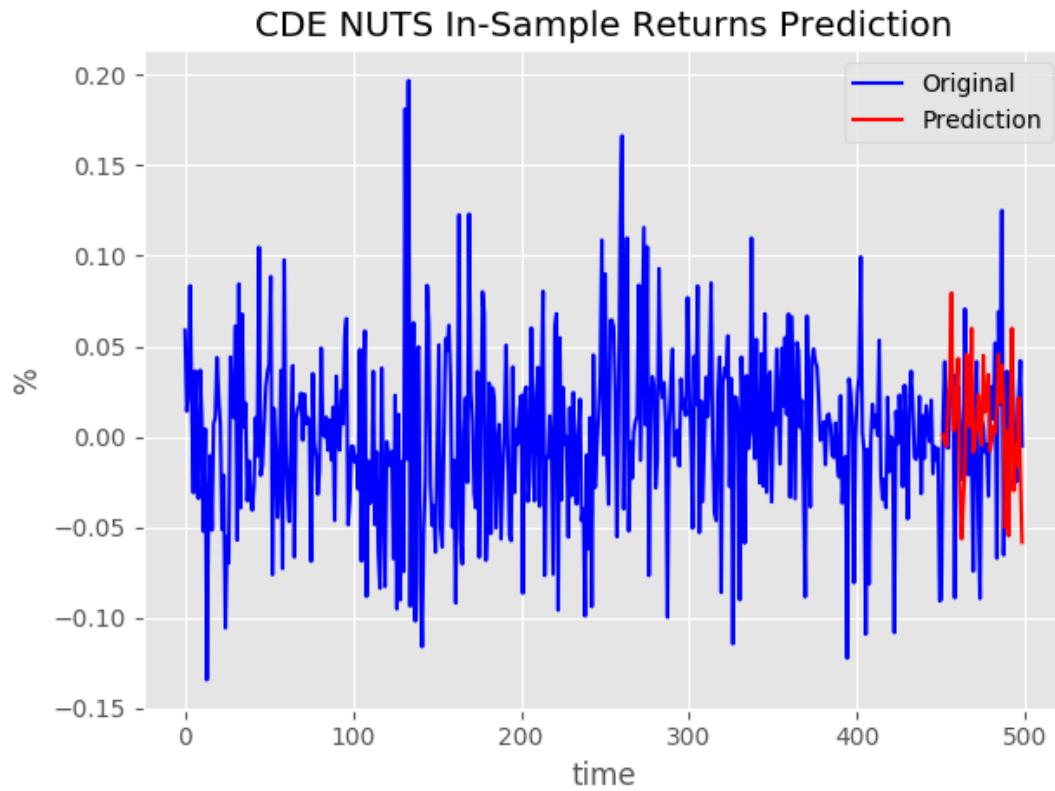


FIGURE 4.184: CDE NUTS in-sample prediction

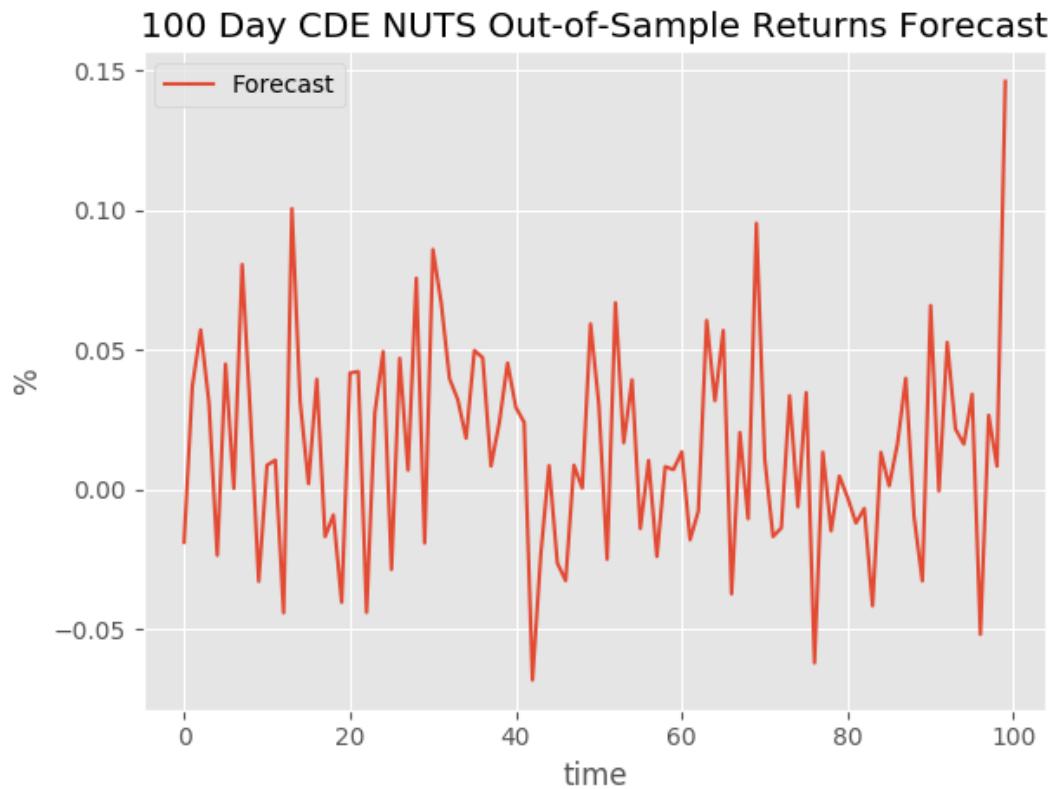


FIGURE 4.185: 100 day CDE NUTS out-of-sample forecast

NAV B scored sharpe ratios of -2.800 for the original returns, and -2.914 for the predicted returns in the in-sample test.

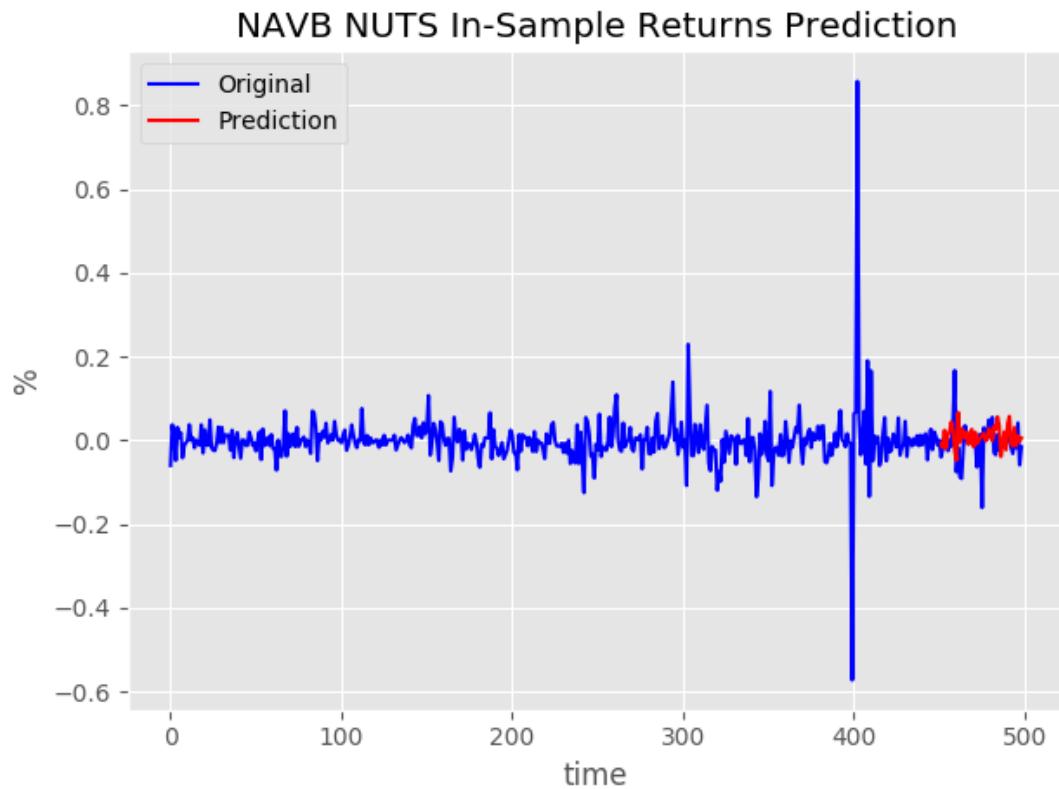


FIGURE 4.186: NAVB NUTS in-sample prediction

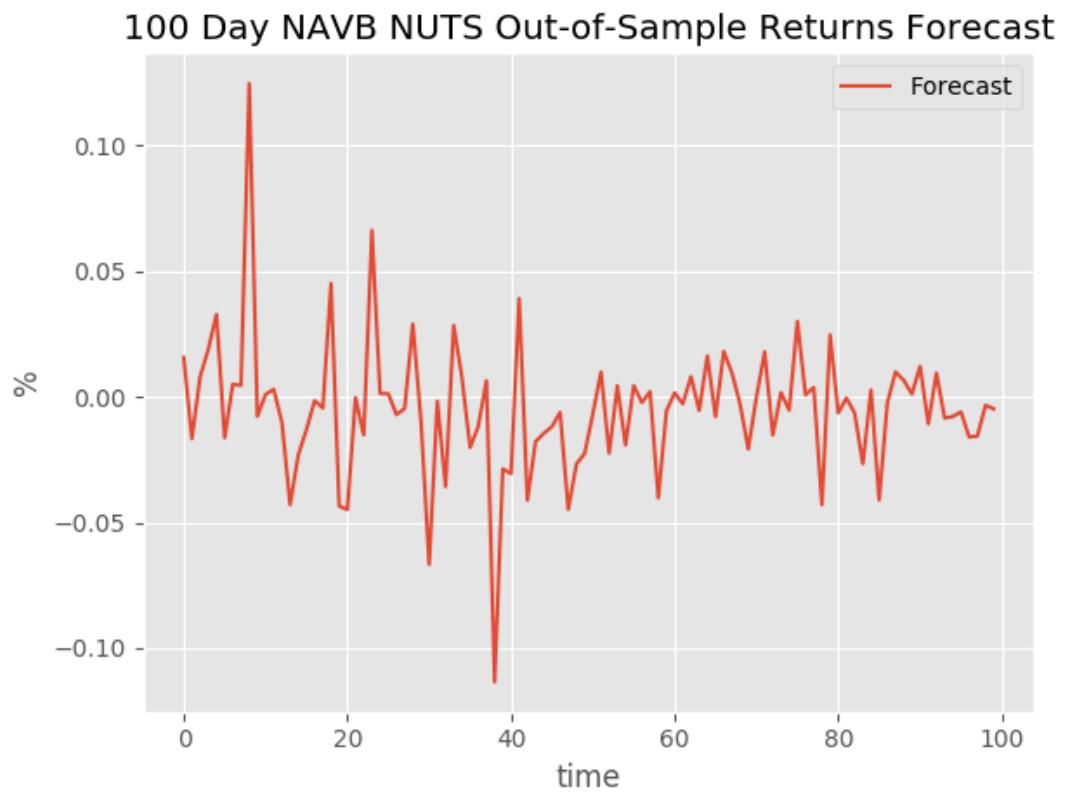


FIGURE 4.187: 100 day NAVB NUTS out-of-sample forecast

HRG scored sharpe ratios of 1.362 for the original returns, and -1.812 for the predicted returns in the in-sample test.

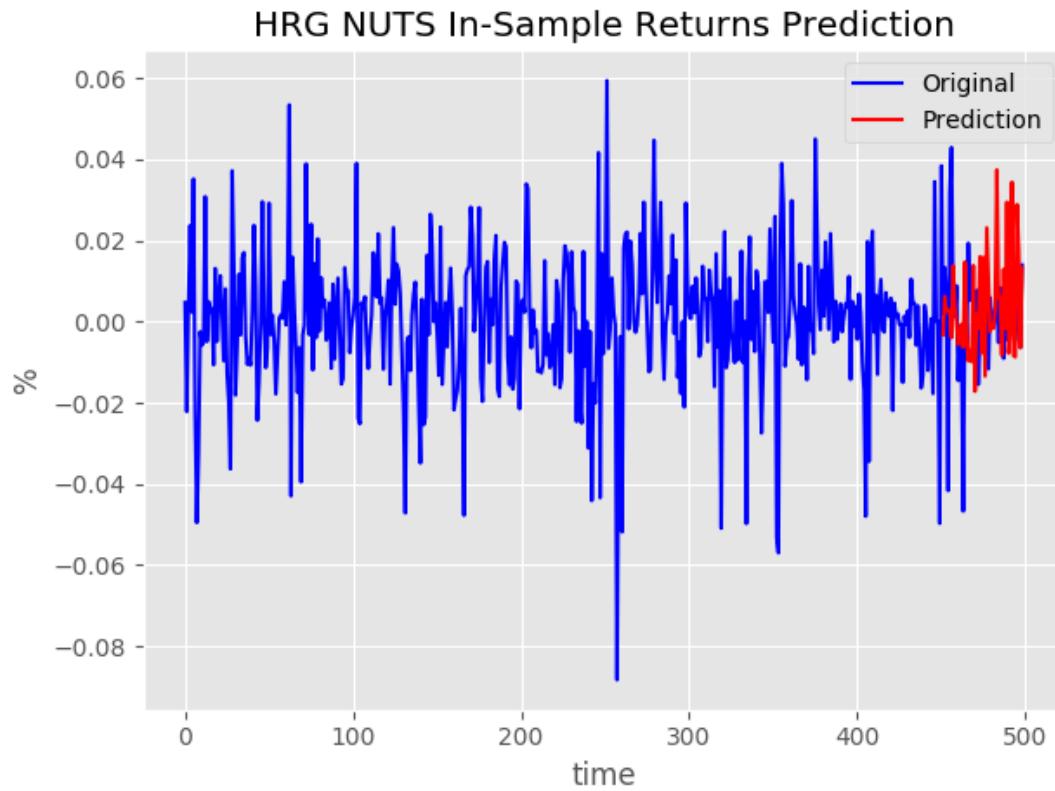


FIGURE 4.188: HRG NUTS in-sample prediction

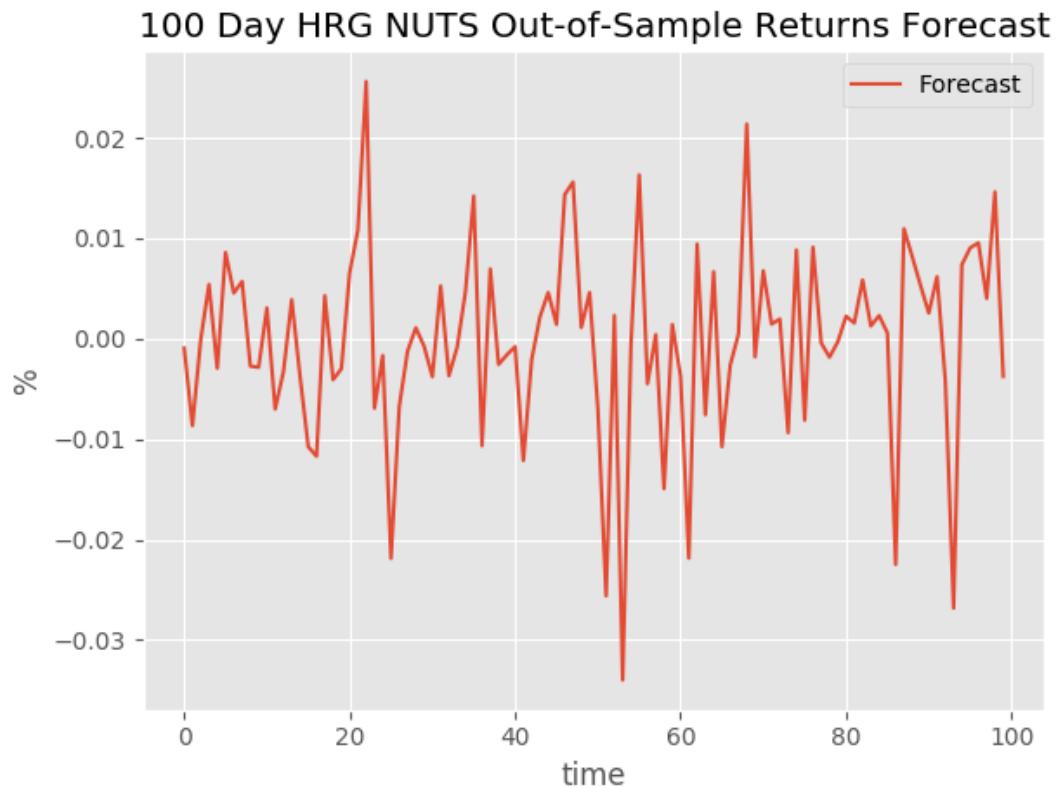


FIGURE 4.189: 100 day HRG NUTS out-of-sample forecast

HL scored sharpe ratios of 0.506 for the original returns, and 0.923 for the predicted returns in the in-sample test.

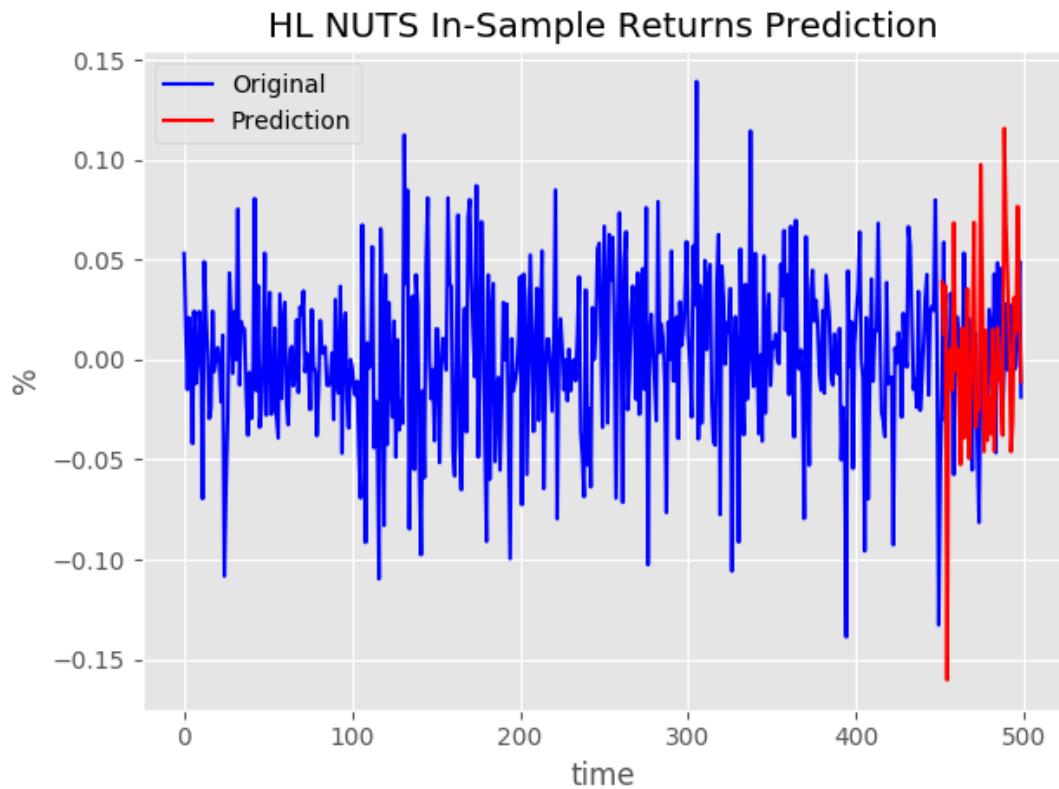


FIGURE 4.190: HL NUTS in-sample prediction

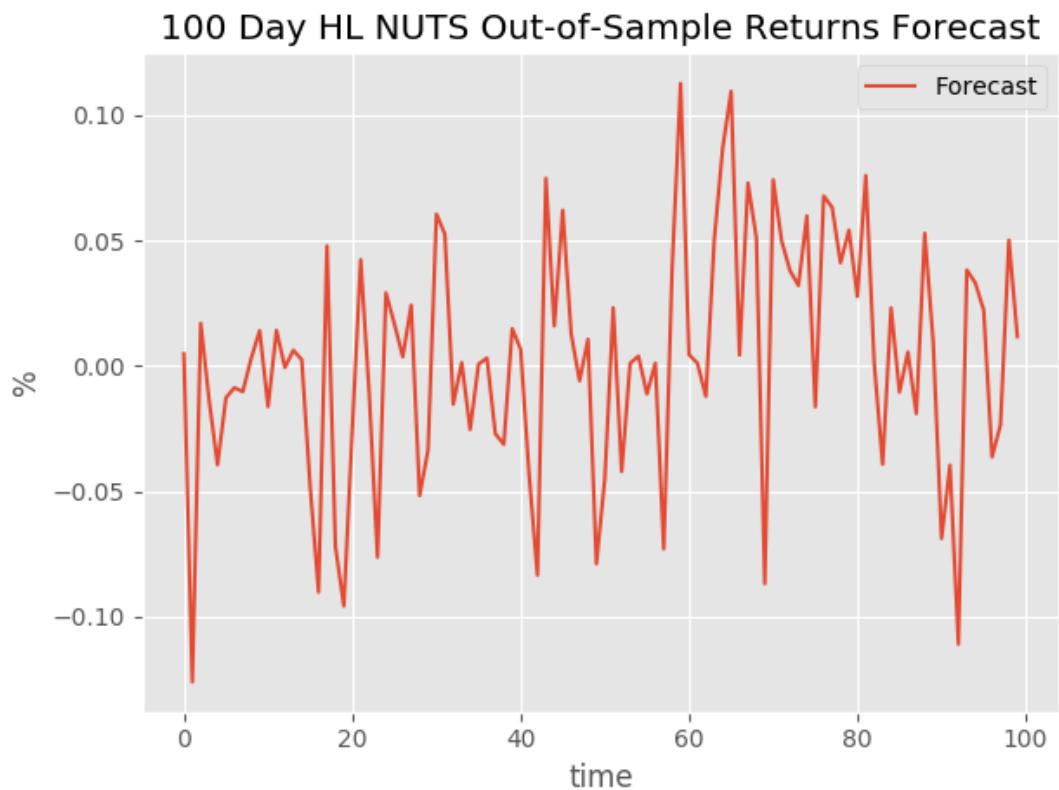


FIGURE 4.191: 100 day HL NUTS out-of-sample forecast

4.4 Strategy

4.4.1 Classification

Starting Capital	\$100,000
Total Capital Used	\$103,110.65
Sharpe Ratio	0.346
Portfolio Value	\$149,126.306
Algorithm Period Return	0.491
Benchmark Period Return	1.008
Algorithm Volatility	0.272
Benchmark Volatility	0.156

TABLE 4.12: Machine Learning Classifier strategy with only upwards forecasts

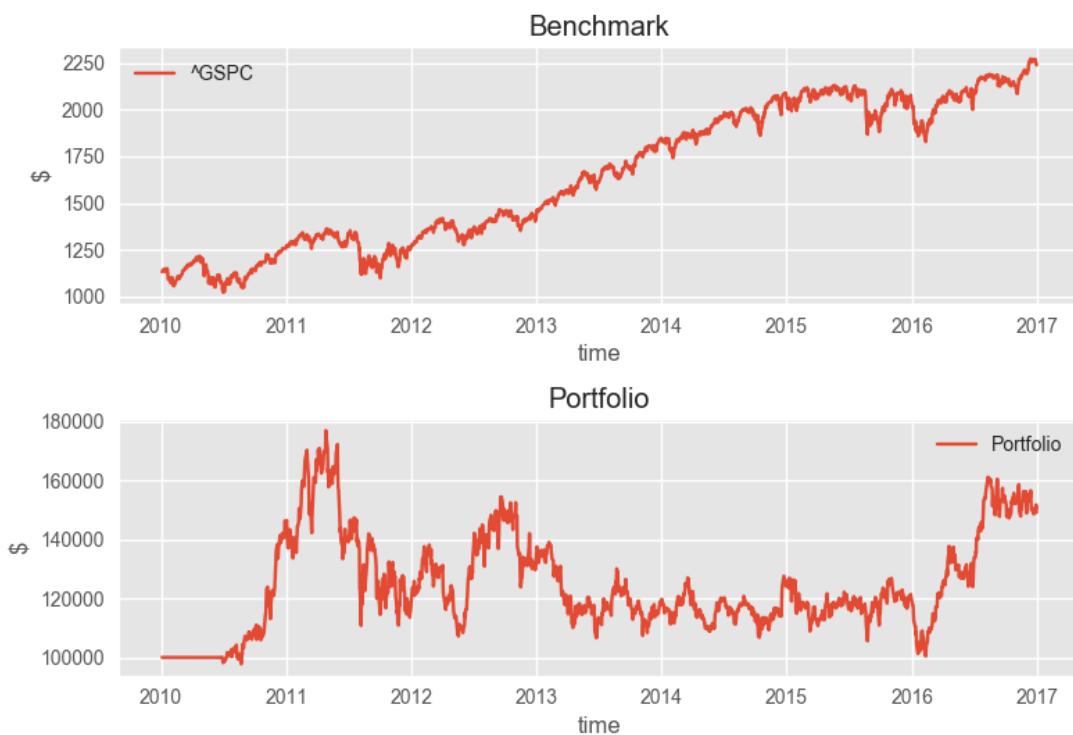


FIGURE 4.192: Machine Learning Classifier strategy with only upwards forecasts

Starting Capital	\$100,000
Total Capital Used	\$76,853.53
Sharpe Ratio	-0.548
Portfolio Value	\$20,134.519
Algorithm Period Return	-0.799
Benchmark Period Return	1.008
Algorithm Volatility	0.323
Benchmark Volatility	0.156

TABLE 4.13: Machine Learning Classifier strategy with upwards and downwards forecasts

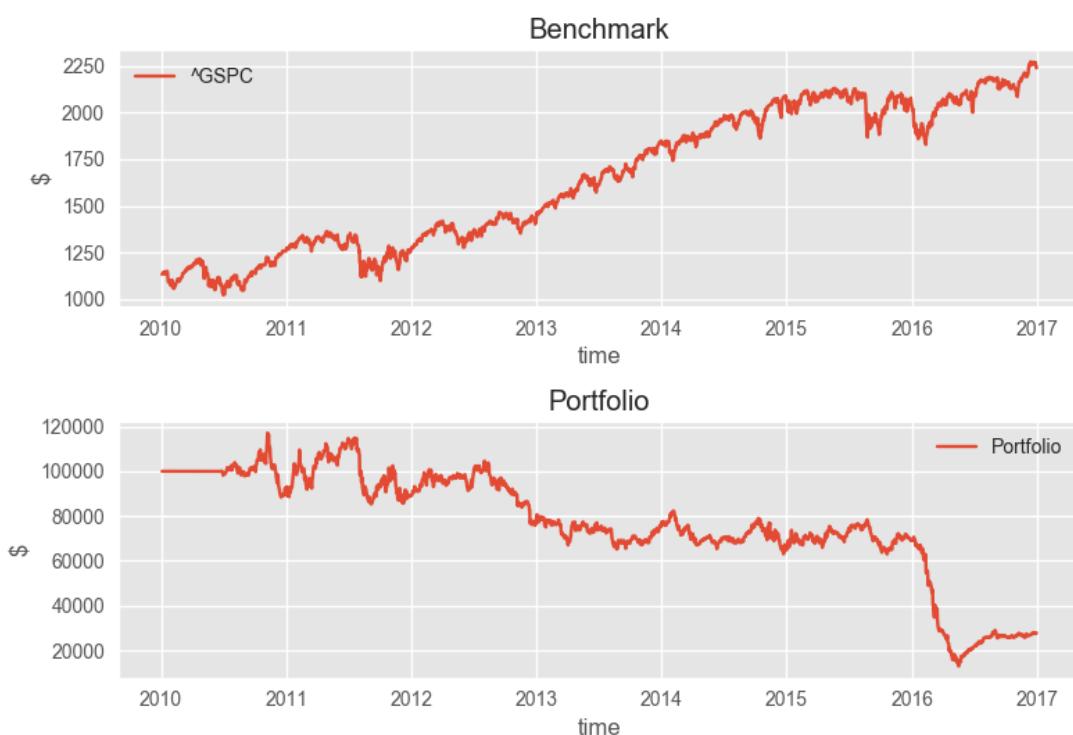


FIGURE 4.193: Machine Learning Classifier strategy with upwards and downwards forecasts

Starting Capital	\$100,000
Total Capital Used	\$295,844.66
Sharpe Ratio	0.385
Portfolio Value	\$167,474.009
Algorithm Period Return	0.675
Benchmark Period Return	1.008
Algorithm Volatility	0.393
Benchmark Volatility	0.156

TABLE 4.14: Machine Learning Classifier strategy with stop loss

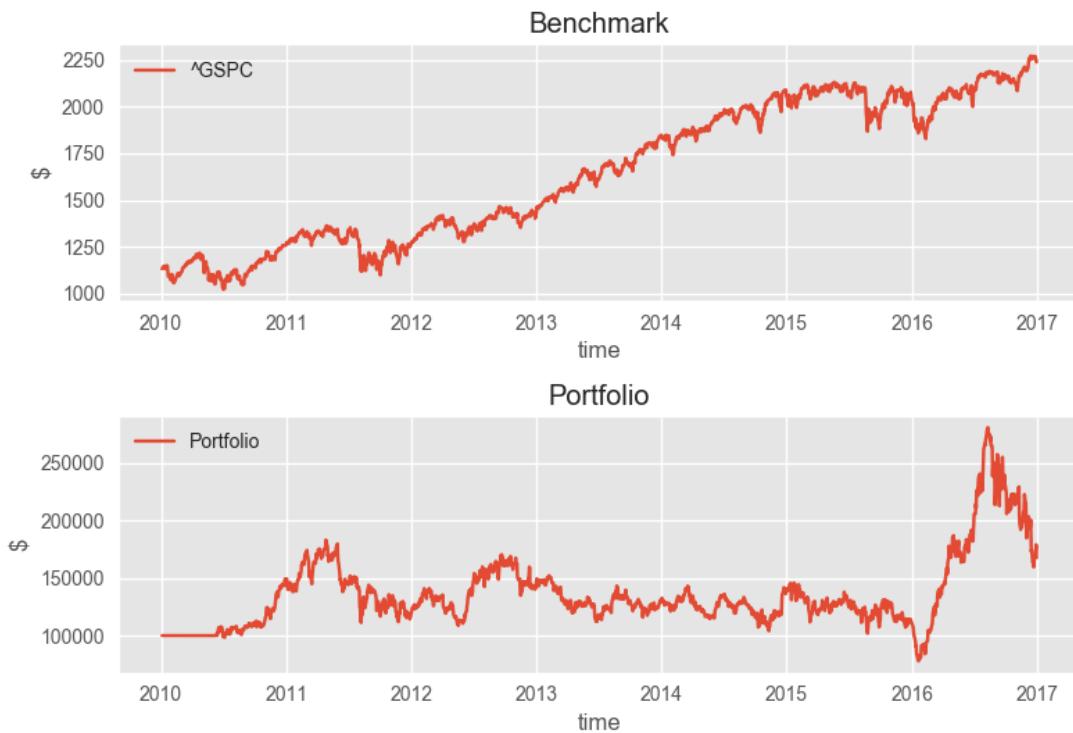


FIGURE 4.194: Machine Learning Classifier strategy with stop loss

4.4.2 Regression

Starting Capital	\$100,000
Total Capital Used	\$229,547.84
Sharpe Ratio	0.420
Portfolio Value	\$183,714.616
Algorithm Period Return	0.837
Benchmark Period Return	1.008
Algorithm Volatility	0.373
Benchmark Volatility	0.156

TABLE 4.15: Machine Learning Regression strategy with only upwards forecasts

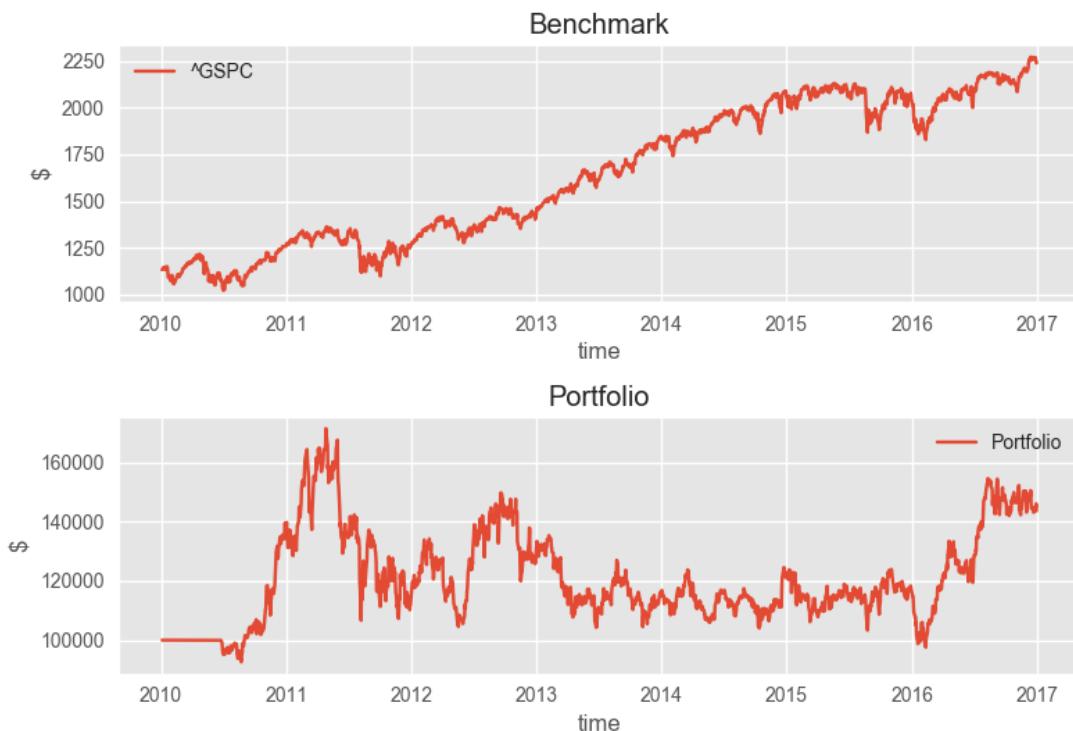


FIGURE 4.195: Machine Learning Regression strategy with only upwards forecasts

Starting Capital	\$100,000
Total Capital Used	\$80,121.51
Sharpe Ratio	-0.258
Portfolio Value	\$19,628.34
Algorithm Period Return	-0.804
Benchmark Period Return	1.008
Algorithm Volatility	0.493
Benchmark Volatility	0.156

TABLE 4.16: Machine Learning Regression strategy with upwards and downwards forecasts

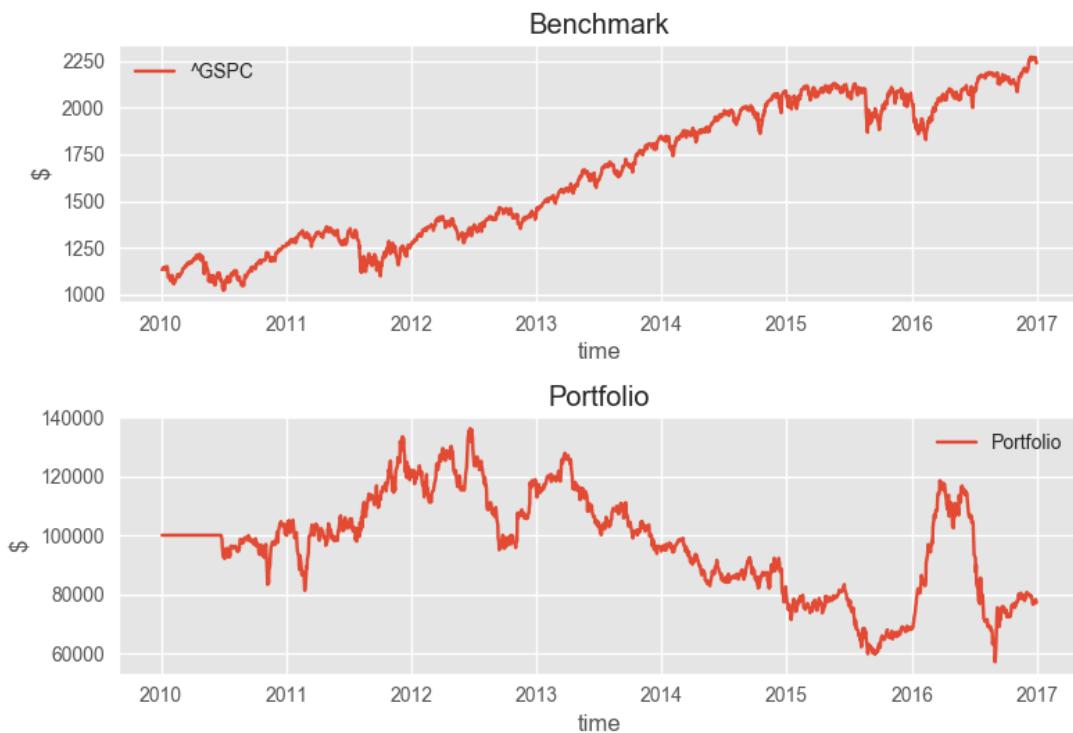


FIGURE 4.196: Machine Learning Regression strategy with upwards and downwards forecasts

Starting Capital	\$100,000
Total Capital Used	\$229,547.84
Sharpe Ratio	0.420
Portfolio Value	\$183,714.616
Algorithm Period Return	0.837
Benchmark Period Return	1.008
Algorithm Volatility	0.373
Benchmark Volatility	0.156

TABLE 4.17: Machine Learning Regression strategy with stop loss

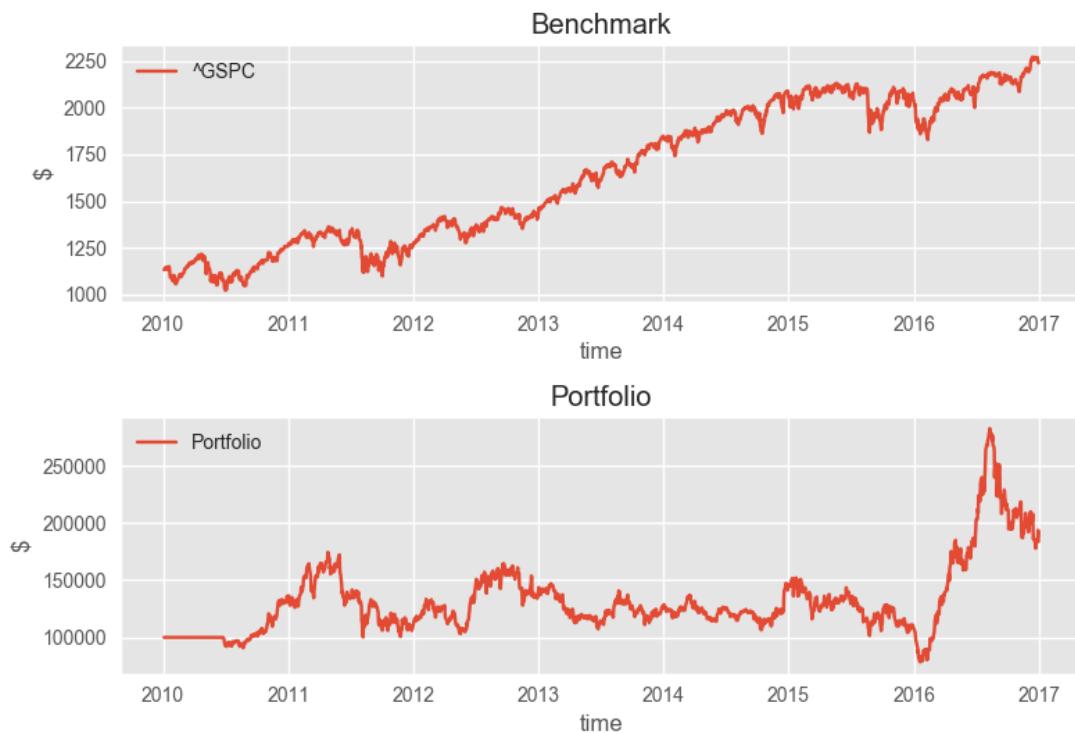


FIGURE 4.197: Machine Learning Regression strategy with stop loss

Chapter 5

Discussion and Suggestions for Future Research

5.1 Time Series Analysis

5.1.1 Random Walk

The random walk theory suggests that stock price changes have the same distribution and are independent of each other, so the past movement or trend of a stock price or market cannot be used to predict its future movement. In short, this is the idea that stocks take a random and unpredictable path.

As can be seen in figures 4.4, 4.6, 4.8, and 4.10, the correlation plots show this theory to be false and that past movement is related to future movement.

The histogram of returns show that the stocks follow a normal distribution, where some stocks showed higher returns than others as can be seen from a wider x-axis.

5.1.2 Ordinary Least Squares (OLS)

This algorithm achieved a good fit in the in-sample tests, having a prediction rate above 96% for all the stocks tested.

5.1.3 Auto Regressive (AR)

This algorithm failed to achieve a good fit in the in-sample tests, having a huge difference in sharpe ratios based on the original price returns and in-sample predicted price returns. The algorithm failed completely in forecasting price returns in out-of-sample testing.

The histogram of returns further showed that the algorithm did not achieve a good fit, as the histogram itself was not completely symmetrical.

5.1.4 Moving Average (MA)

This algorithm faired better than AR having a lower difference in sharpe ratios based on the original price returns and in-sample predicted price returns, however still failed to achieve a good fit in the in-sample tests.

Although achieving a better fit, the histogram of returns still showed flaws as they were not perfectly symmetrical.

5.1.5 Auto Regressive Moving Average (ARMA)

This algorithm once again faired better than the previous algorithm, MA, having an even lower difference in sharpe ratios based on the original price returns and in-sample predicted price returns, however still failed to achieve a good fit in the in-sample tests. As can be seen in the time series analysis plots, ARMA showed to have very heavy tails in the QQ and probability plots. The algorithm also failed to forecast future price returns for certain stocks, as is evident in figure 4.81.

As can be seen from the histogram of returns, a better fit was achieved as the histogram was more symmetrical than those of the previous algorithms.

5.1.6 Auto Regressive Integrated Moving Average (ARIMA)

This algorithm faired worse than the previous algorithm, ARMA, having a higher difference in sharpe ratios based on the original price returns and in-sample predicted price returns.

This can also be seen from the histogram of returns, in which its distribution was less normal than that of the previous algorithm.

5.2 Machine Learning

5.2.1 Classification

5.2.1.1 Decision Tree

This algorithm faired well in predicting stock price movements in the in-sample tests, having an accuracy score above 75% for all the stocks prices that were forecast.

5.2.1.2 Boosted Decision Tree

This algorithm yielded slightly better results when compared to the previous algorithm in predicting stock price movements in the in-sample tests.

5.2.1.3 Support Vector Machine (SVM)

This algorithm once again yielded slightly better results when compared to the previous algorithm in predicting stock price movements in the in-sample tests.

5.2.1.4 Random Forest

This algorithm scored the same accuracy as the previous algorithm.

5.2.1.5 K-Nearest Neighbour

This algorithm's accuracy was not consistent as it scored mixed results when tested against all the stocks, where the accuracy is some was much lower than others.

5.2.1.6 Logistic Regression

This algorithm faired worse than the others having not achieved an accuracy score of above 80% when tested against all the stocks.

5.2.1.7 Bernoulli Naive Bayes

This algorithm faired the worst from the algorithms tested so far, scoring accuracy scores in the lower 70th percentile.

5.2.1.8 Gaussian Naive Bayes

This algorithm also scored a lower accuracy score, showing that Naive Bayes isn't the best form of forecasting stock prices.

5.2.1.9 Neural Network

This algorithm, although did not score the best accuracy score from the other algorithms, still faired well in correctly predicting the price movements of the stocks.

5.2.1.10 Stochastic Gradient Descent

This algorithm faired the worst from the lot, varying heavily in accuracy scores, in which one stock had an accuracy score of 59%. Even though it faired much better in correctly predicting the price movements of other stocks, this algorithm proved to be unreliable.

5.2.2 Regression

5.2.2.1 Decision Tree

As can be seen in figure 4.102, the algorithm not only failed to achieve a good fit in the in-sample test, but also failed completely in predicting any values at all. Figure 4.103 also shows that the algorithm predicted a sharp fall at the end of the out-of-sample test, which could be related to the incorrect fit achieved in the in-sample test. This was reflected in the metric score, having a high number of data losses, and low accuracy scores.

5.2.2.2 Boosted Decision Tree

This algorithm also did not do very well, as can be seen in figures 4.112, 4.114, 4.116, and 4.118, the algorithm not only failed to achieve a good fit in the in-sample test, but also failed completely in predicting any values at all. The algorithm also predicted some sharp inclines at the end of the test, as can be seen in figure 4.115. The bad fit achieved is also reflected in the metric scores, in which there was a high number of data losses, and low accuracy scores.

5.2.2.3 K-Nearest Neighbour

This algorithm also didn't fair too well, showing bad fits in figures 4.122, 4.124, 4.126, 4.128, and 4.130. Similarly to the previous algorithm, the alforithm failed completely in predicting any values at all in some parts of the in-sample test.

5.2.2.4 Random Forest

Similarly to previous algorithms, this algorithm failed to achieve a good fit as can be seen in figures 4.132, 4.134, 4.136, 4.138, and 4.140. The algorithm also failed to predict any values at all in the of the in-sample tests, and also predicted sharp inclines and declines at the end some of the out-of-sample tests.

5.2.2.5 Linear Regression

This algorithm performed very well in the in-sample tests, achieving a good fit with high accuracy scores above 95% with a low number of data losses.

5.2.2.6 Neural Network

This algorithm, although scored very well in some of the in-sample tests, was not very reliable as accuracy scores varied between each test, performing extremely well in some, however not too well in others.

5.2.2.7 Stochastic Gradient Descent

This algorithm performed fairly well in most in-sample tests, having high accuracy scores and low data losses escept for CDE as can be seen in figure 4.164.

5.3 Bayesian Statistics

5.3.0.1 Metropolis-Hastings

This algorithm failed to achieve a good fit in the in-sample tests, having a huge difference in sharpe ratios based on the original price returns and in-sample predicted price returns.

5.3.0.2 No-U-Turn-Sampler (NUTS)

The algorithm achieved a good fit in the in-sample tests, having very similar sharpe ratios based on the original price returns and in-sample predicted price returns.

5.4 Strategy

5.4.1 Classification

When tasked with predicting rises in stock prices, the algorithm did fairly well, marking a total return of 49.1%, this however did not beat the benchmark's return of 100.8%. The algorithm underperformed immensely when tasked at also predicting stock price falls, marking a negative total return of -80%. To compensate for this, a stop loss was added to sell all positions if the stock in question falls below -20%, this resulted in a profit of 67.5%.

5.4.2 Regression

The performance of this algorithm was extremely similar to that of the previous one, this goes to show that both classifiers and regressors can be used for stock price prediction. When tasked with predicting rises in stock prices, the algorithm did fairly well, marking a slightly lower total return of 43%. The algorithm also underperformed immensely when tasked at also predicting stock price falls, marking a negative total return of -80.4%. To compensate for this, a stop loss was added to sell all positions if the stock in question falls below -20%, this resulted in a profit of 83.7%.

Chapter 6

Conclusion

Three financial forecasting methods were presented in this report, two of which showed little to no potential of ever producing any statistically significant result when the correct methodology was applied. The third method, machine learning, showed some potential in the tests carried out, which is why this method was built into an automated algorithmic strategy to trade with. The algorithm proved to be successful in forecasting future prices, using both classification and regression methods. However, the backtesting proved this method to fail in forecasting price falls. Once this factor was removed from the equation, the algorithms were very successful and reported a profit by the end of the test. This is however not always ideal as stocks which could fall in price could be catastrophic to the strategy. A stop loss would be ideal in insuring that no positions are held in downward falling stocks. It was also evident that regression methods were more successful in forecasting future price movements when compared to classification methods.

If there is anything that this report shows, is that profitable stock market prediction is an extremely tough problem. Even though the strategies reported a profit by the end of the backtest, they still did not beat the market. Whether it is at all possible to use such methods to outperform the market's returns, ultimately remains an open question. These findings support the Efficient Market Hypothesis, proving that casual investors are better off investing in passive buy and hold strategies consisting of index funds and ETFs. However, there was some evidence found showing that the Random Walk Hypothesis does not hold true for all cases, as some stocks did show signs of repeating trends.

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.
- [18] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15, 2014.