Forecasting Cryptocurrency Prices (ETH) Using Various Time Series Models

Johan Chua, Nicholas Delianedis, Grace Pham

12/21/23

Contents

I Introduction **2**

II Results **3**
(a) Time Series Plot 3
(b) ARIMA 5
(c) ETS 8
(d) Holt-Winters 11
(e) NNETAR 14
(f) Prophet 16
(g) Forecast Combination 17

III Conclusions and Future Work **19**

IV References **20**

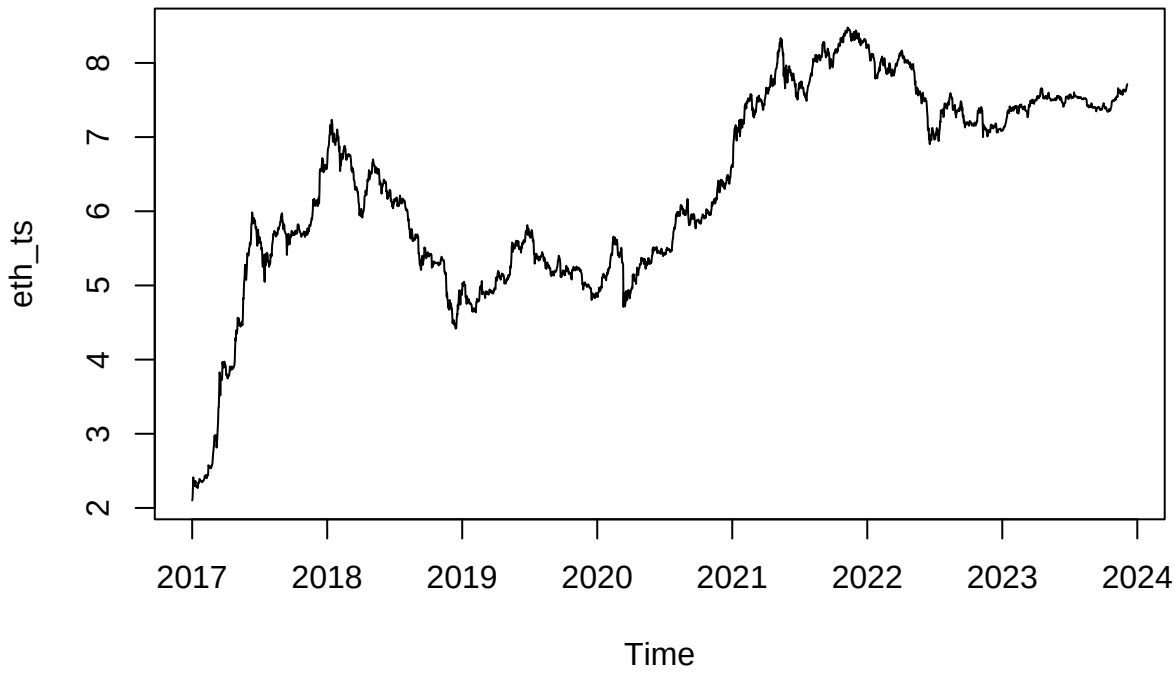
I Introduction

In this project, we use a variety of methods to forecast and model the price of the Ethereum cryptocurrency. Ethereum is a digital currency that serves as a decentralized application and smart contract platform. The dataset we used includes the daily Ethereum price from January 1, 2017 to December 4, 2023, sourced from the Federal Reserve Bank of St. Louis (FRED). We will estimate an ARIMA, ETS, Holt-Winters, NNETAR, Prophet, and forecast combination. Using these trained models, we generate forecasts for future Ethereum prices. By forecasting and combining these models, we intend to improve our understanding of Ethereum's market dynamics, offering insightful information that could help stakeholders, analysts, and investors make more informed decisions.

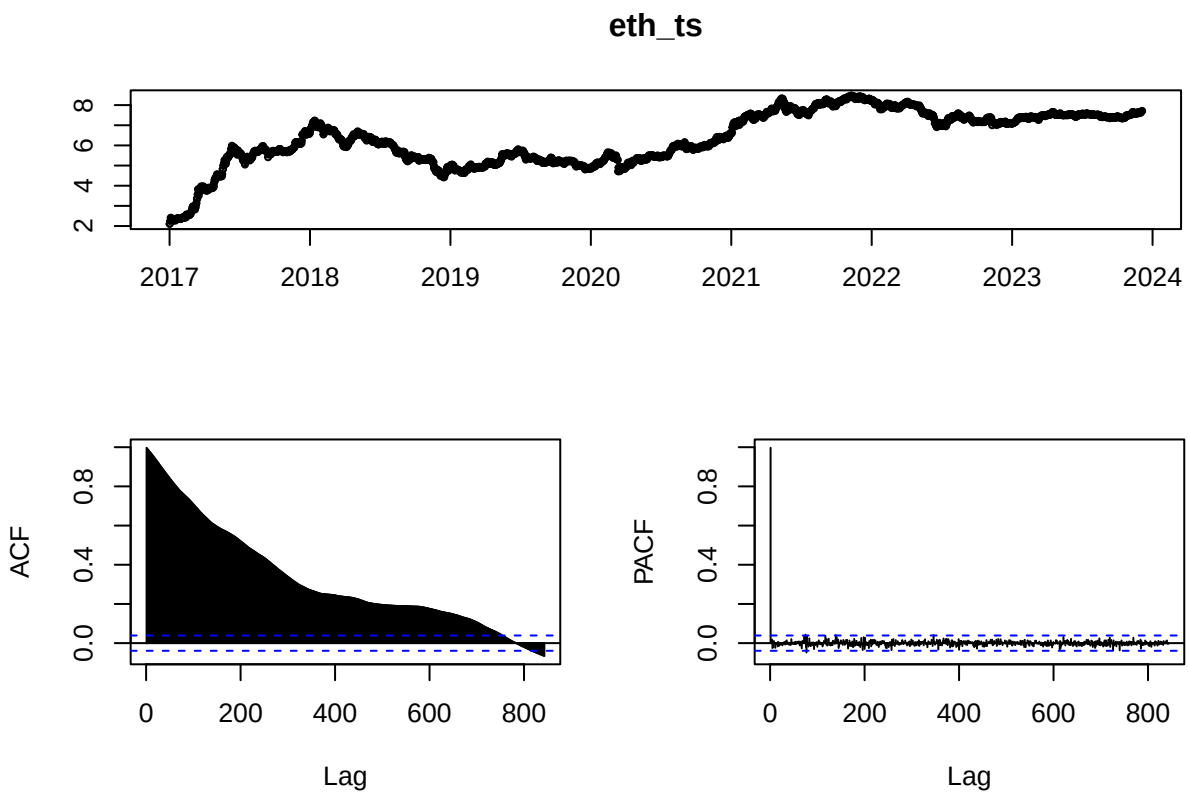
II Results

(a) Time Series Plot

```
eth <- read.csv("ETH.csv")
eth_ts <- ts(log(eth$CBETHUSD), start = c(2017,1), frequency = 365)
plot(eth_ts)
```




```
tsdisplay(eth_ts)
```

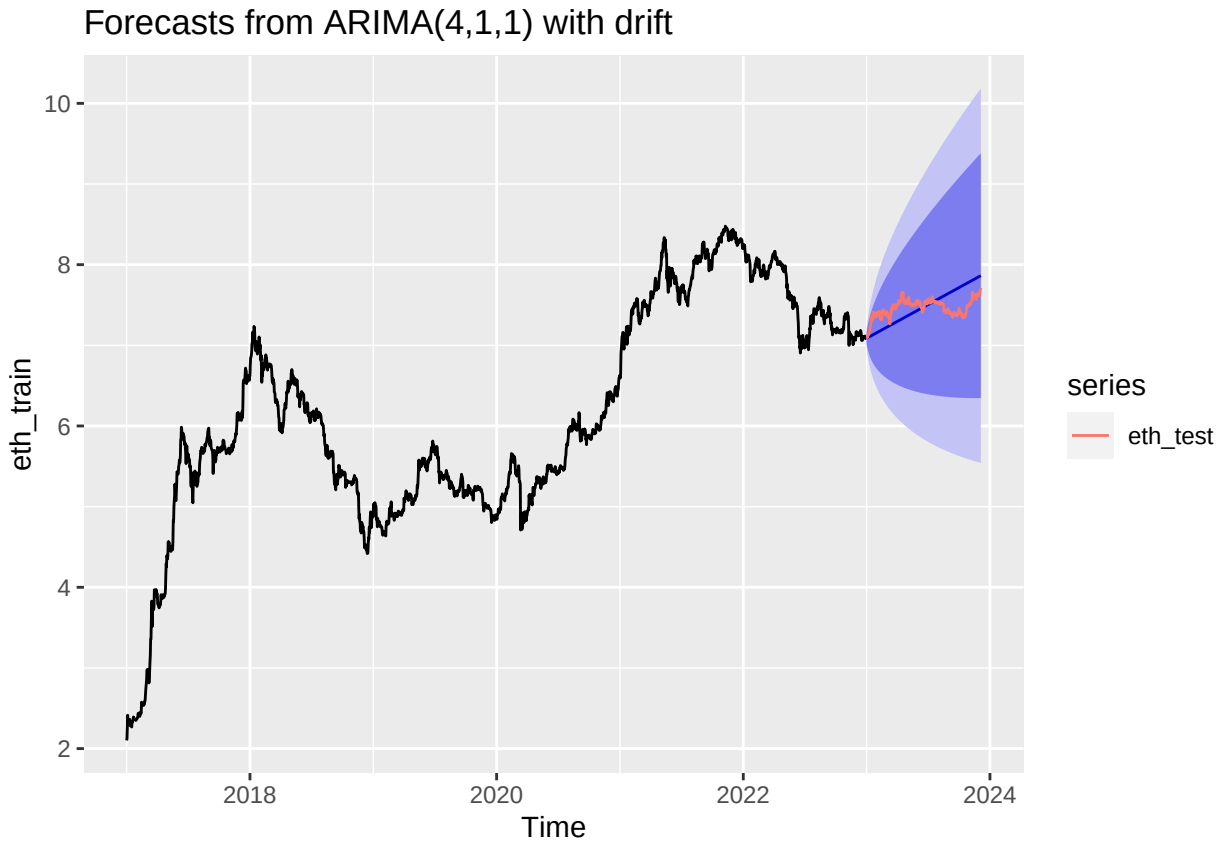



```
# train / test datasets
eth_train <- window(eth_ts, end = c(2022, 365))
eth_test <- window(eth_ts, start = c(2023, 1))
```

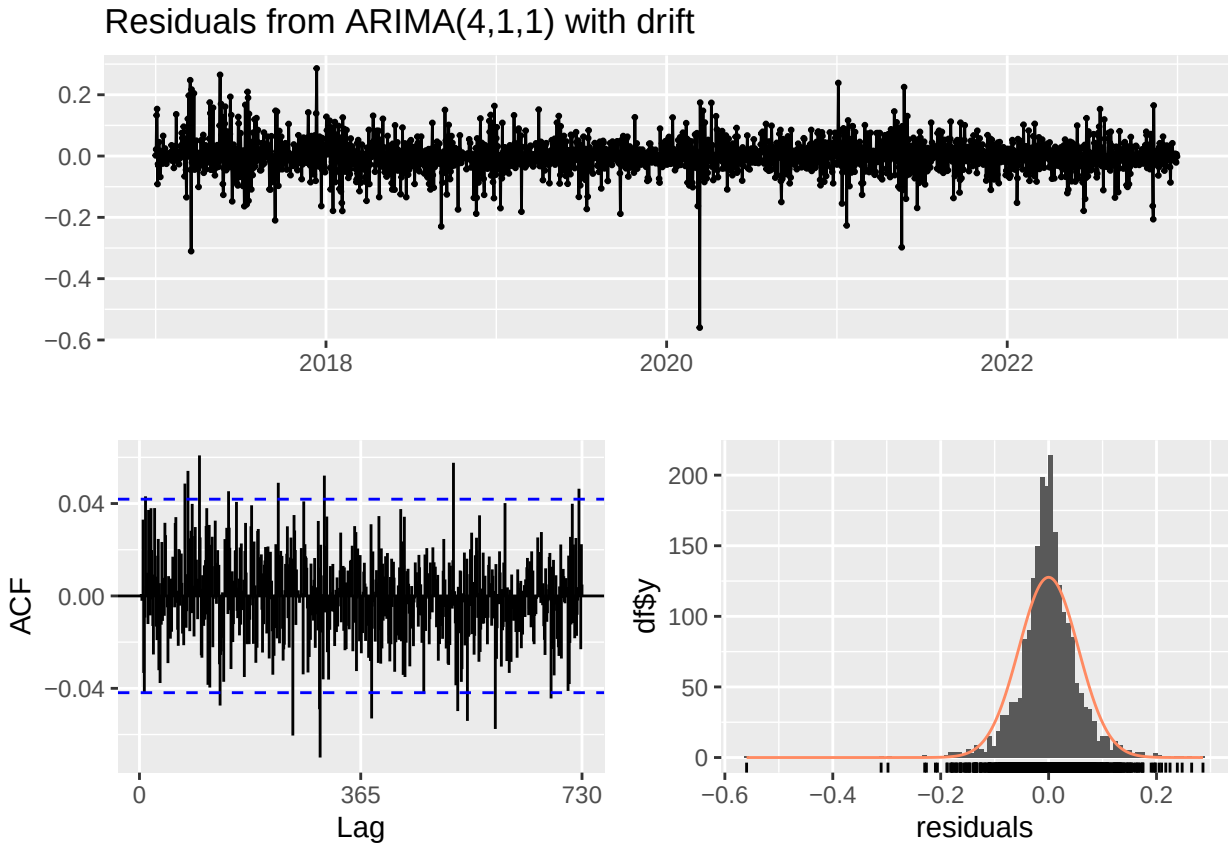
(b) ARIMA

```
# model selection
eth_arima <- auto.arima(eth_train)

# model forecast
eth_arimaforecast <- forecast(eth_arima, h = length(eth_test))
autoplot(eth_arimaforecast) + autolayer(eth_test)
```

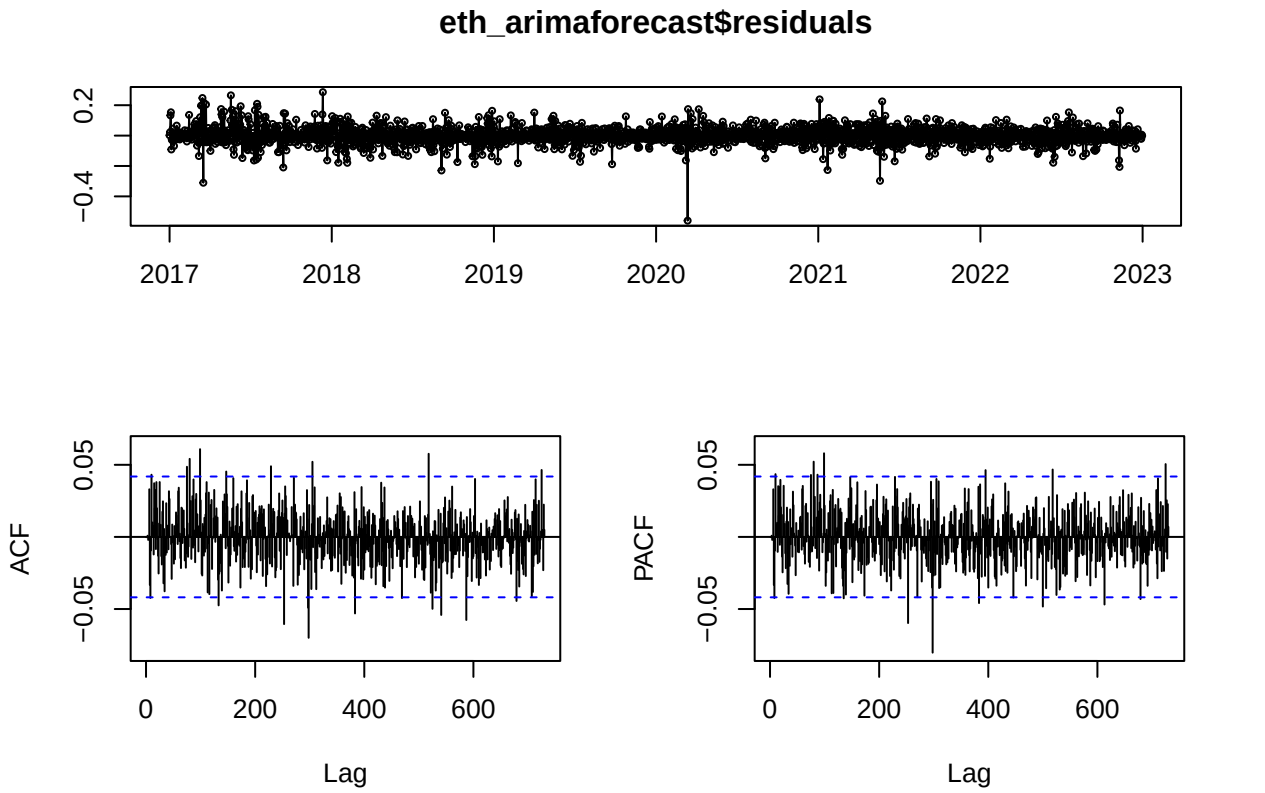



```
# model diagnostics
checkresiduals(eth_arimaforecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,1,1) with drift
## Q* = 421.54, df = 433, p-value = 0.6443
##
## Model df: 5.   Total lags used: 438
```

```
tsdisplay(eth_arimaforecast$residuals)
```




```
accuracy(eth_arimaforecast, eth_test)
```

```
##                         ME       RMSE        MAE          MPE      MAPE
## Training set -0.0000190573 0.05512113 0.03782763  0.009182906 0.6531456
## Test set     -0.0109534835 0.20338188 0.17575526 -0.148600368 2.3567033
##                    MASE          ACF1 Theil's U
## Training set 0.03128781 -0.0001119348        NA
## Test set     0.14536987  0.9922177365  8.405354
```

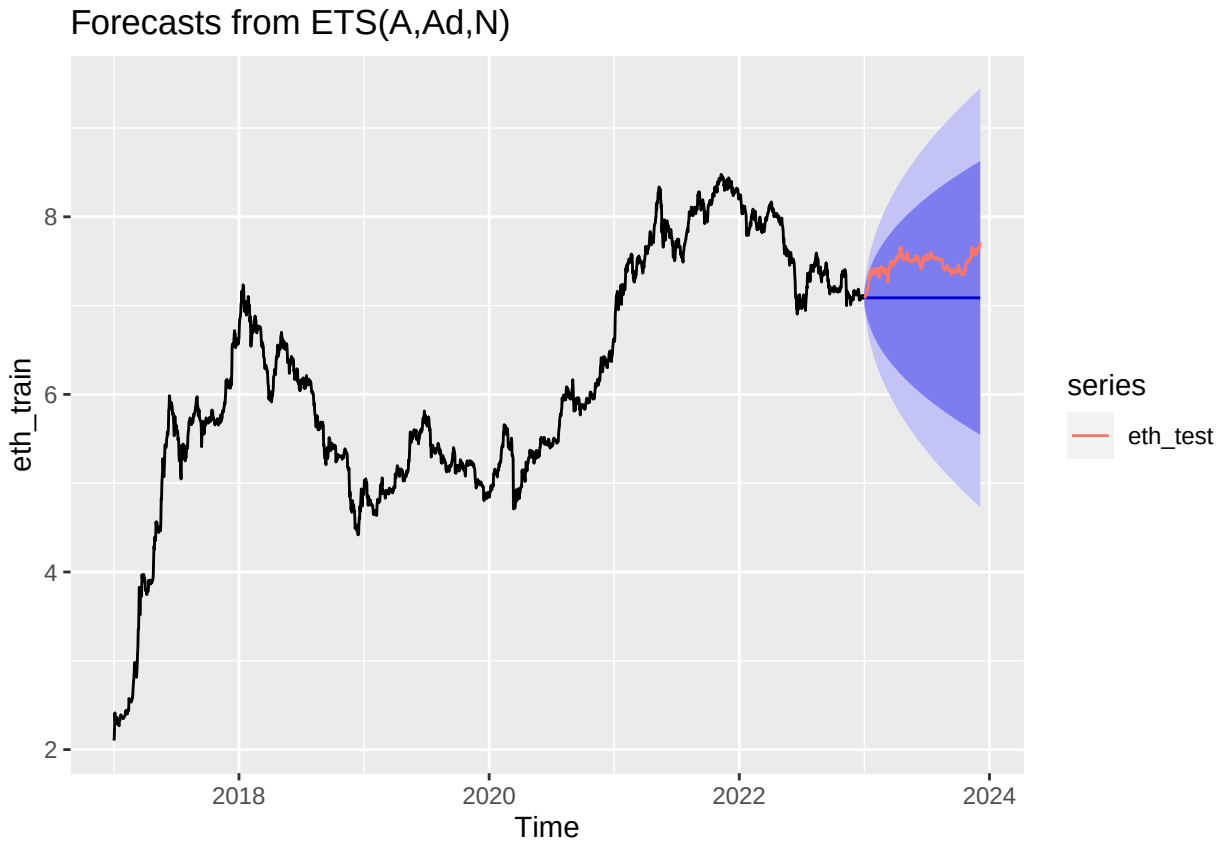
First we use the auto.arima function to estimate our ARIMA model. The model it estimated was an ARIMA(4,1,1). When looking at the residual plots, there are some significant lags, indicating serial autocorrelation. We then forecast our ARIMA model and plot. Looking at the rmse values we see that our training set performs better than the test set.

(c) ETS

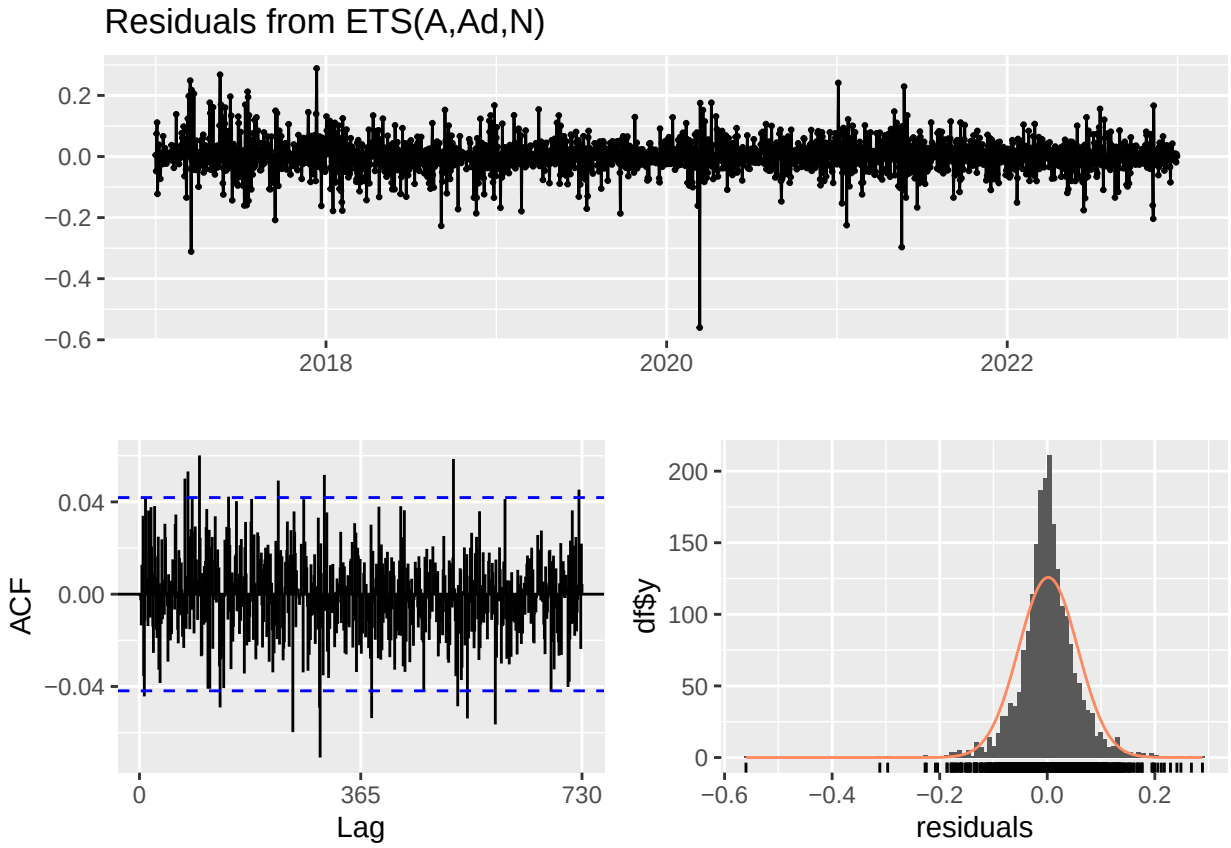
```
# model selection
eth_ets <- ets(eth_train)
```

```
## Warning in ets(eth_train): I can't handle data with frequency greater than 24.
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.
```

```
# model forecast
eth_etsforecast <- forecast(eth_ets, h = length(eth_test))
autoplot(eth_etsforecast) + autolayer(eth_test)
```

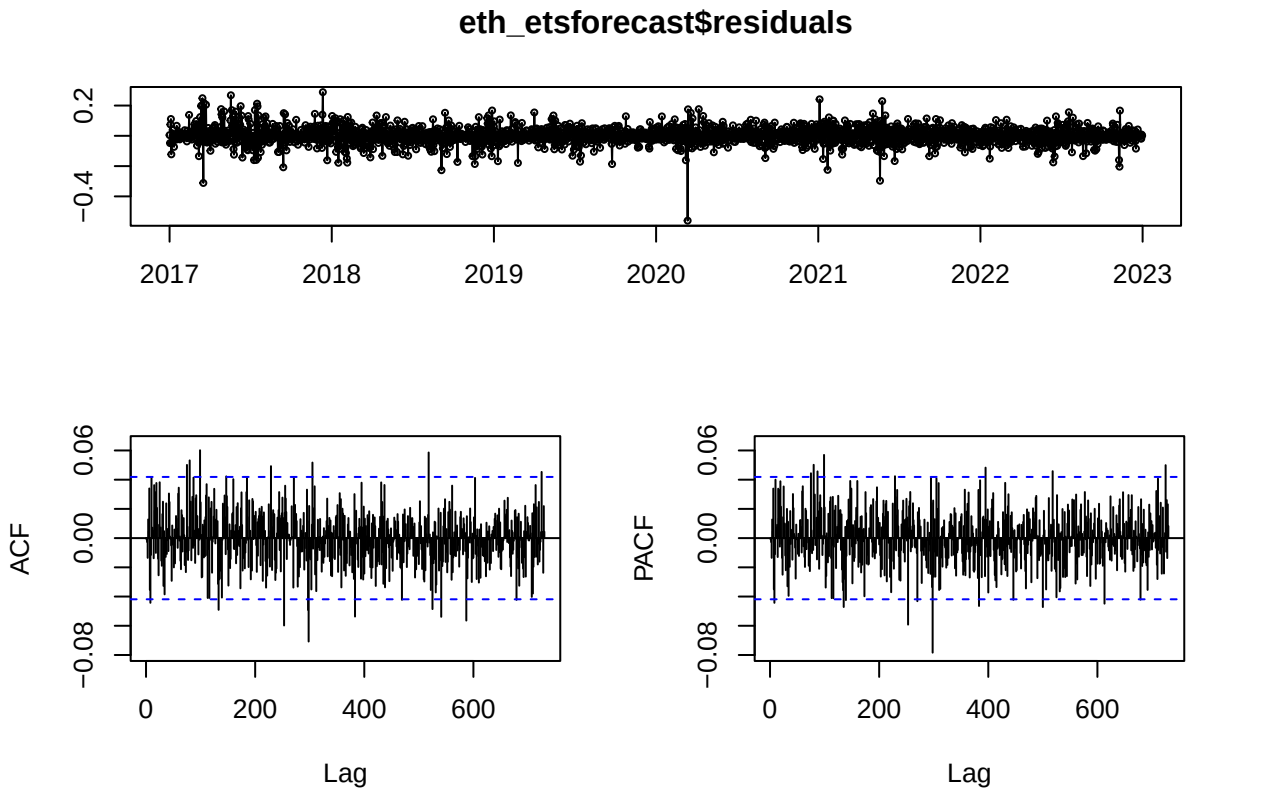



```
# model diagnostics
checkresiduals(eth_etsforecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,N)
## Q* = 424.83, df = 438, p-value = 0.6652
##
## Model df: 0.   Total lags used: 438
```

```
tsdisplay(eth_etsforecast$residuals)
```




```
accuracy(eth_etsforecast, eth_test)
```

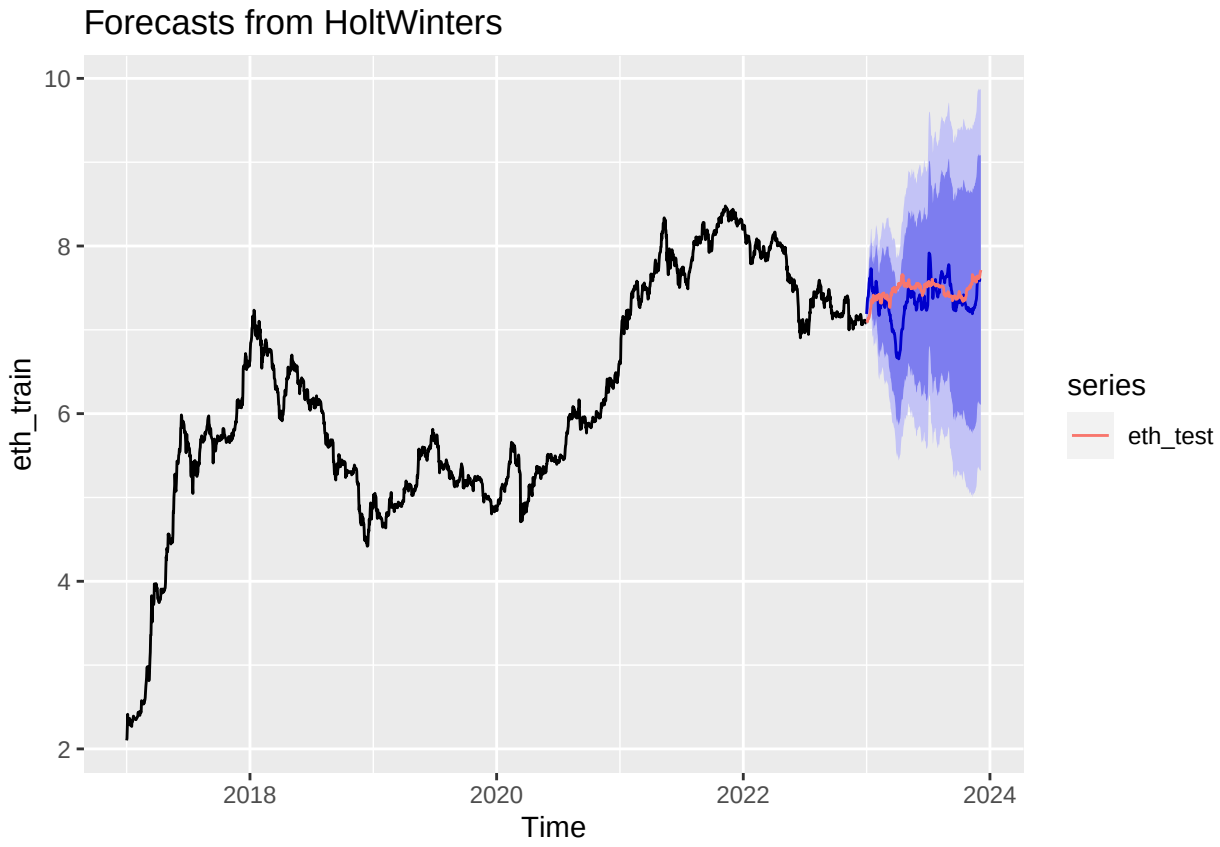
```
##                       ME       RMSE        MAE       MPE      MAPE       MASE
## Training set 0.001780243 0.05511016 0.03780385 0.0365215 0.6528693 0.03126814
## Test set     0.377584475 0.39235865 0.37760102 5.0386540 5.0388875 0.31231959
##                       ACF1 Theil's U
## Training set -0.0009369972        NA
## Test set      0.9473499766  16.07623
```

Here we fit an ETS model to the training data, which chose to use additive errors, damped additive trend, and no seasonality. The ETS forecast just used the last mean, so the forecast is not very useful for our purposes. When looking at the autocorrelation functions of the residuals, we see that the residuals look like white noise, so the model captures most of the data's dynamics.

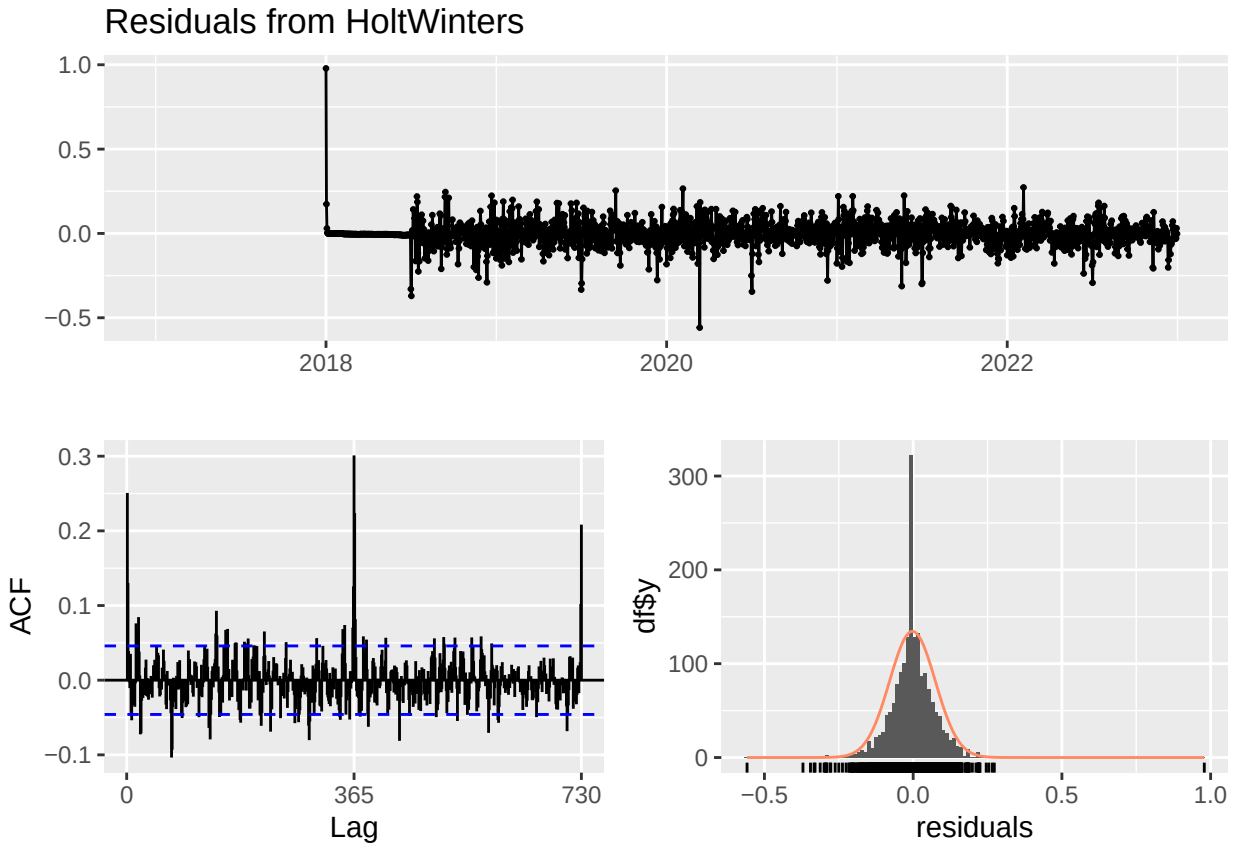
(d) Holt-Winters

```
# model selection
eth_hw <- HoltWinters(eth_train)

# model forecast
eth_hwforecast <- forecast(eth_hw, h = length(eth_test))
autoplot(eth_hwforecast) + autolayer(eth_test)
```

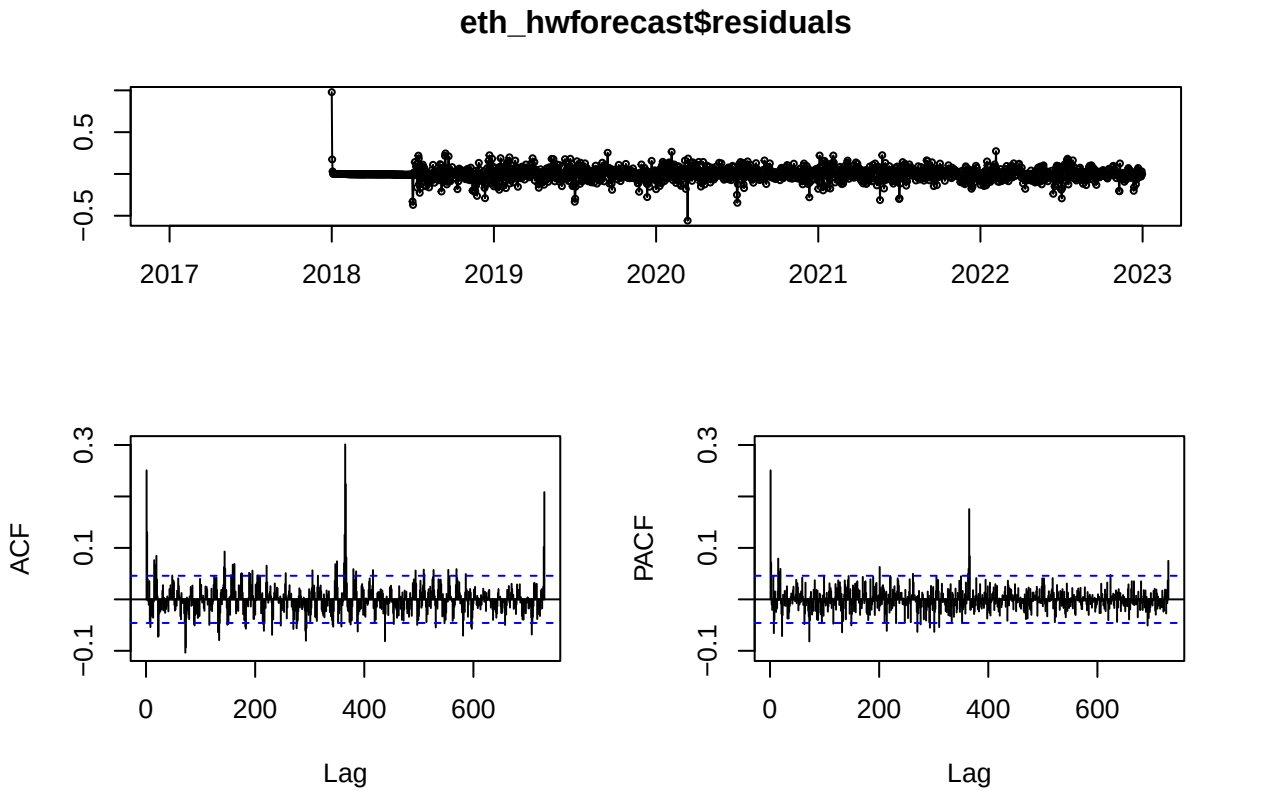



```
# model diagnostics
checkresiduals(eth_hwforecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 1397.7, df = 438, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 438
```

```
tsdisplay(eth_hwforecast$residuals)
```




```
accuracy(eth_hwforecast, eth_test)
```

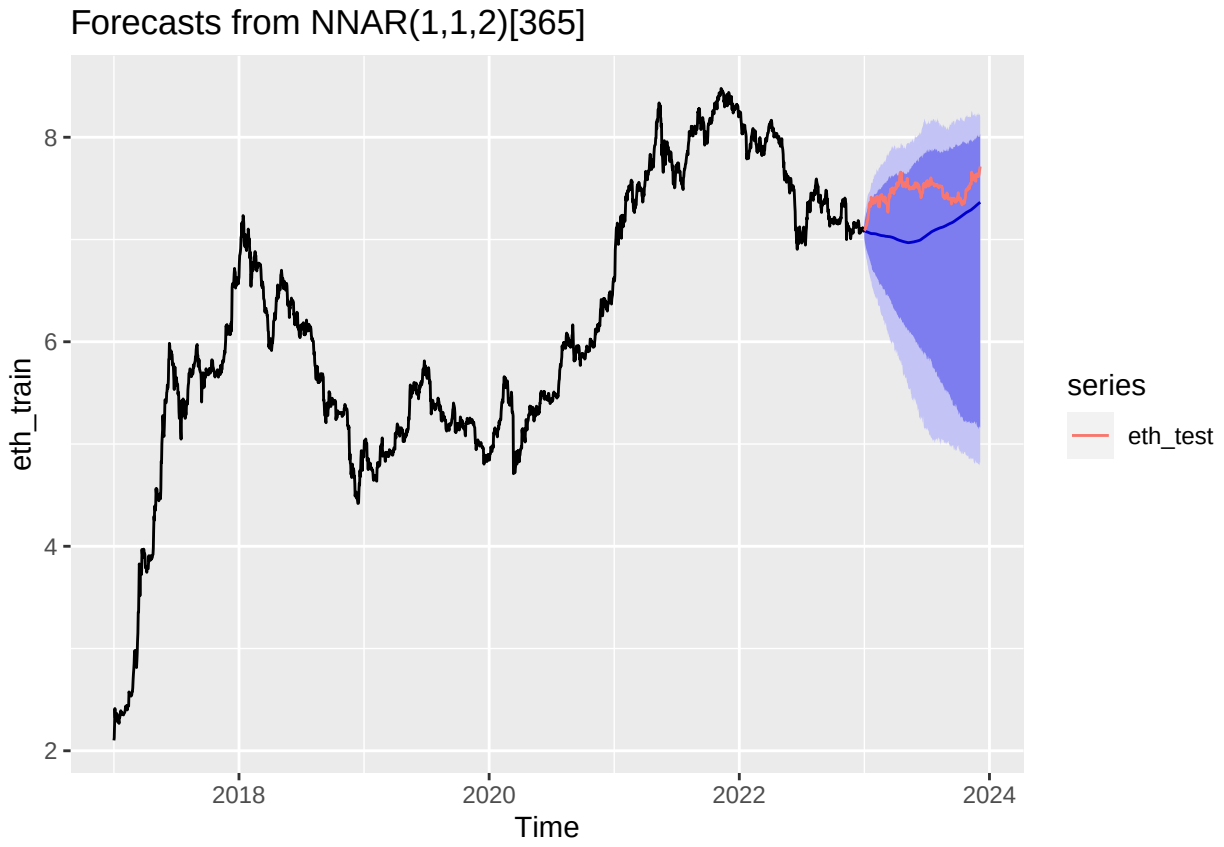
```
##                       ME       RMSE       MAE         MPE      MAPE       MASE
## Training set -0.002417328 0.07683958 0.0515127 -0.04488472 0.8414025 0.04260694
## Test set      0.111655231 0.28256428 0.2079218  1.47240433 2.7789088 0.17197530
##                    ACF1 Theil's U
## Training set 0.2507739        NA
## Test set     0.9822164  11.59936
```

The Holt-Winters forecast that we fit to our data uses trend and an additive seasonal element. We see that the Holt-Winters forecast fit the data very well, and has much lower error measurements than the other models. When looking at the autocorrelation functions of the residuals, we see that the residuals look like white noise except for spikes at lags 1, 365 (one year), 730 (two years), so a seasonal component may capture more dynamics, but the model is very good as is.

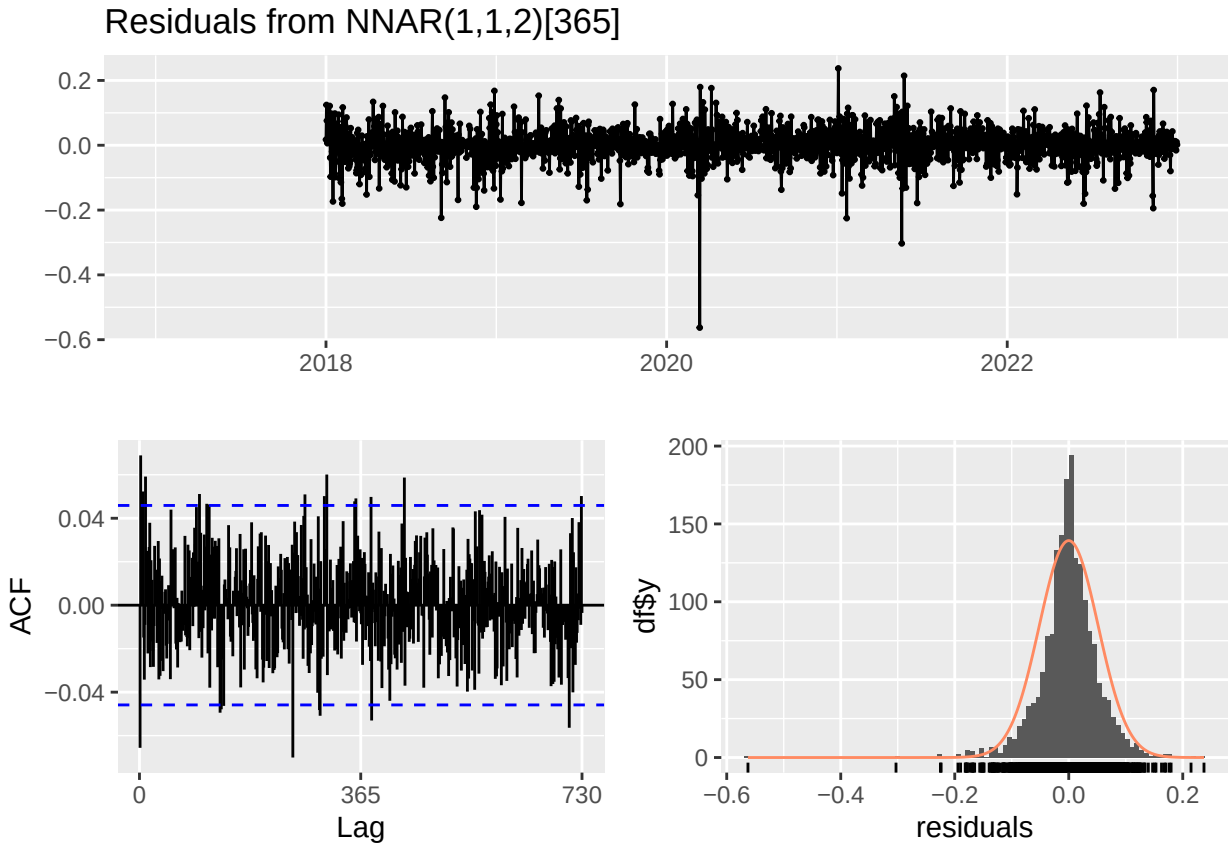
(e) NNETAR

```
# model selection
eth_nnetar <- nnetar(eth_train)

# model forecast
eth_nnetarforecast <- forecast(eth_nnetar, PI = TRUE, h = length(eth_test))
autoplot(eth_nnetarforecast) + autolayer(eth_test)
```




```
# model diagnostics
checkresiduals(eth_nnetar)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,2)[365]
## Q* = 426.02, df = 438, p-value = 0.6502
##
## Model df: 0.   Total lags used: 438
```

```
accuracy(eth_nnetarforecast$mean, eth_test)
```

```
##                 ME      RMSE       MAE      MPE     MAPE     ACF1 Theil's U
## Test set 0.3678891 0.3952222 0.3678976 4.910624 4.910744 0.976204  16.20404
```

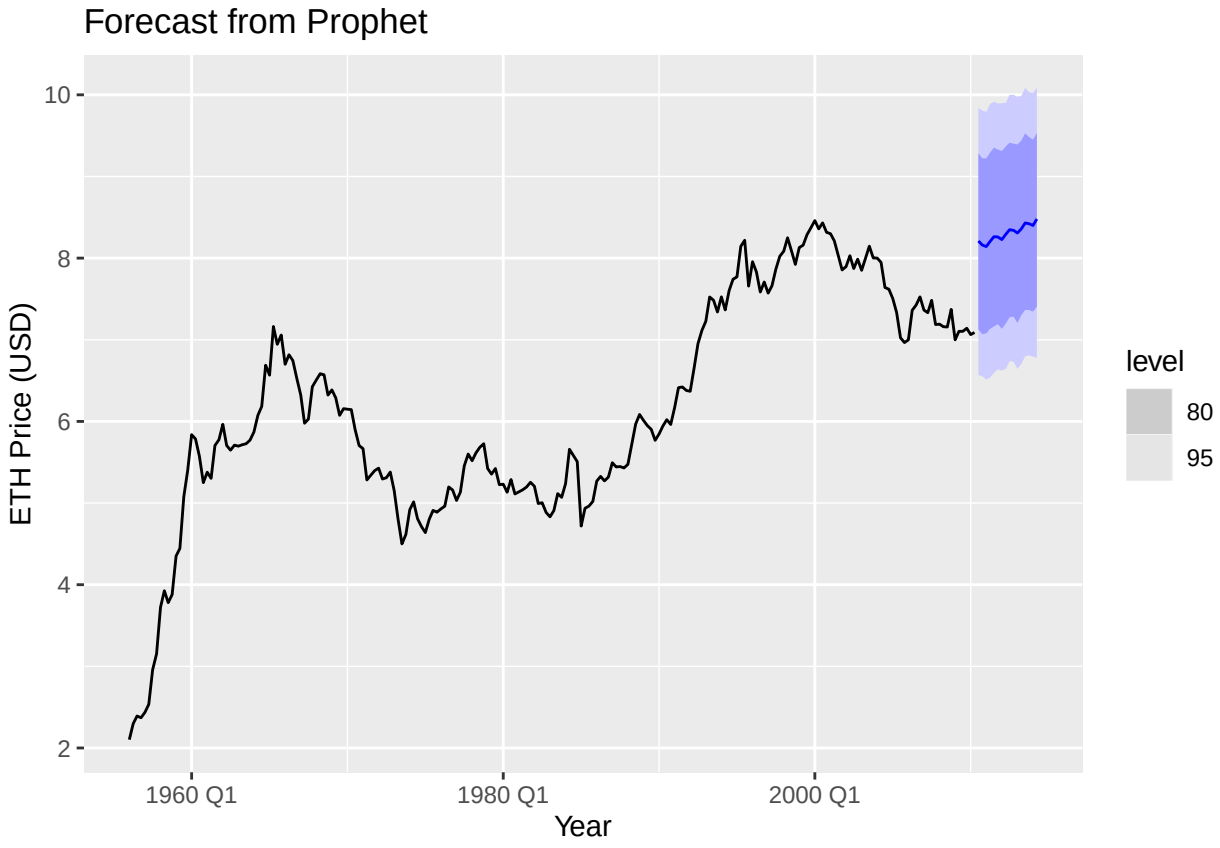
Using neural networks, we fit a model to our data and use it to forecast our testing set. We find that our forecast underestimated the actual testing data set values. Analyzing our NNETAR model residual plots, we see that the residuals do take the form of white noise, with constant variance and an overall mean centered around zero. We also see the residuals are normally distributed. A potential concern is the residual right at the start of 2020 that is much more negative that its surrounding residuals. This reflects the shock due to the COVID pandemic. Lastly, to analyzing testing error, we report a RMSE of 0.386241.

(f) Prophet

```
# model selection
eth_tsibble <- eth2 %>%
  add_column(ETH = eth_train[seq(from = 1, to = length(eth_train),
                                 along.with = eth2$Quarter)])

eth_prophet <- eth_tsibble |>
  model(prophet(ETH ~ season(period = 4, order = 2, type = "multiplicative")))

# model forecast
eth_prophetforecast <- eth_prophet |> forecast(h = "4 years")
eth_prophetforecast |> autoplot(eth_tsibble) +
  labs(x = "Year", y = "ETH Price (USD)") + ggtitle("Forecast from Prophet")
```



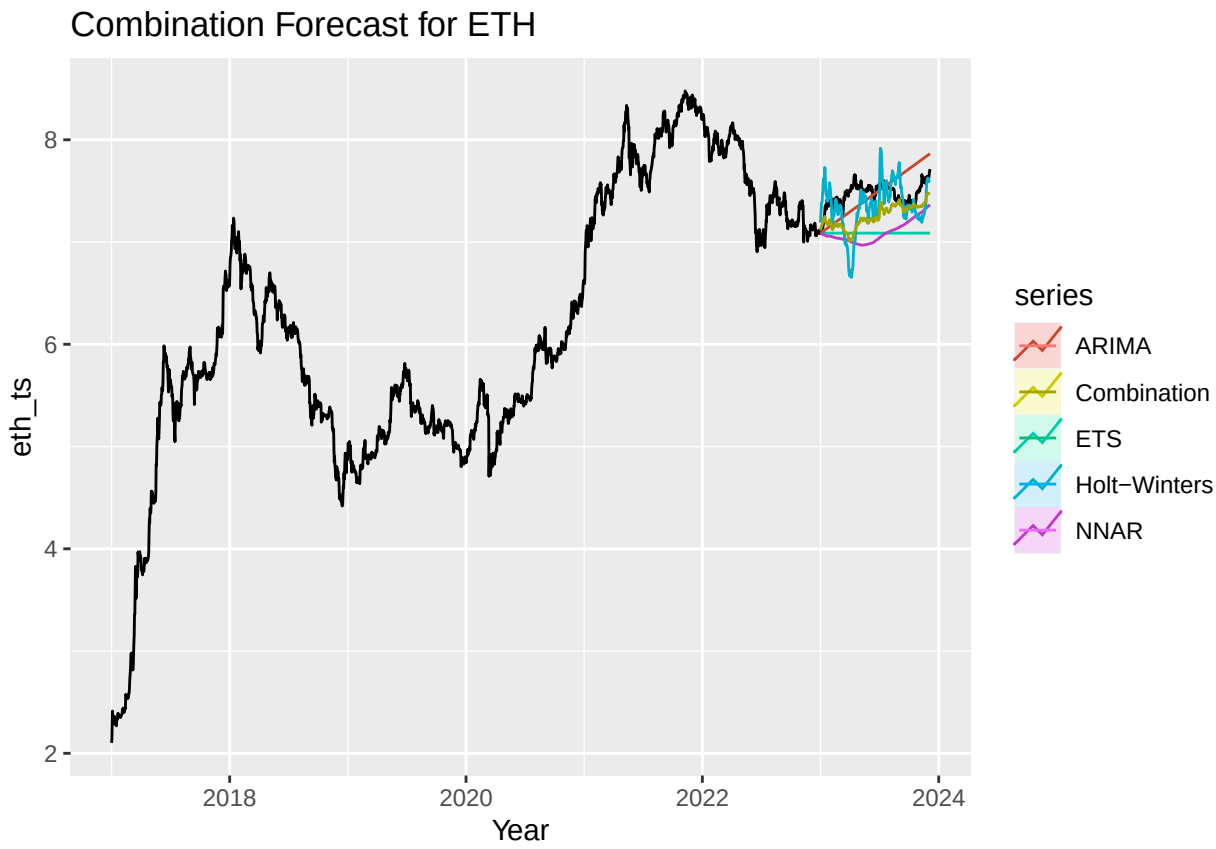

```
# model diagnostics
accuracy(eth_prophetforecast$.mean, eth_test[seq(from = 1, to = length(eth_test), length = 16)])
```

```
##                   ME      RMSE       MAE      MPE    MAPE
## Test set -0.8453735 0.8548624 0.8453735 -11.3648 11.3648
```

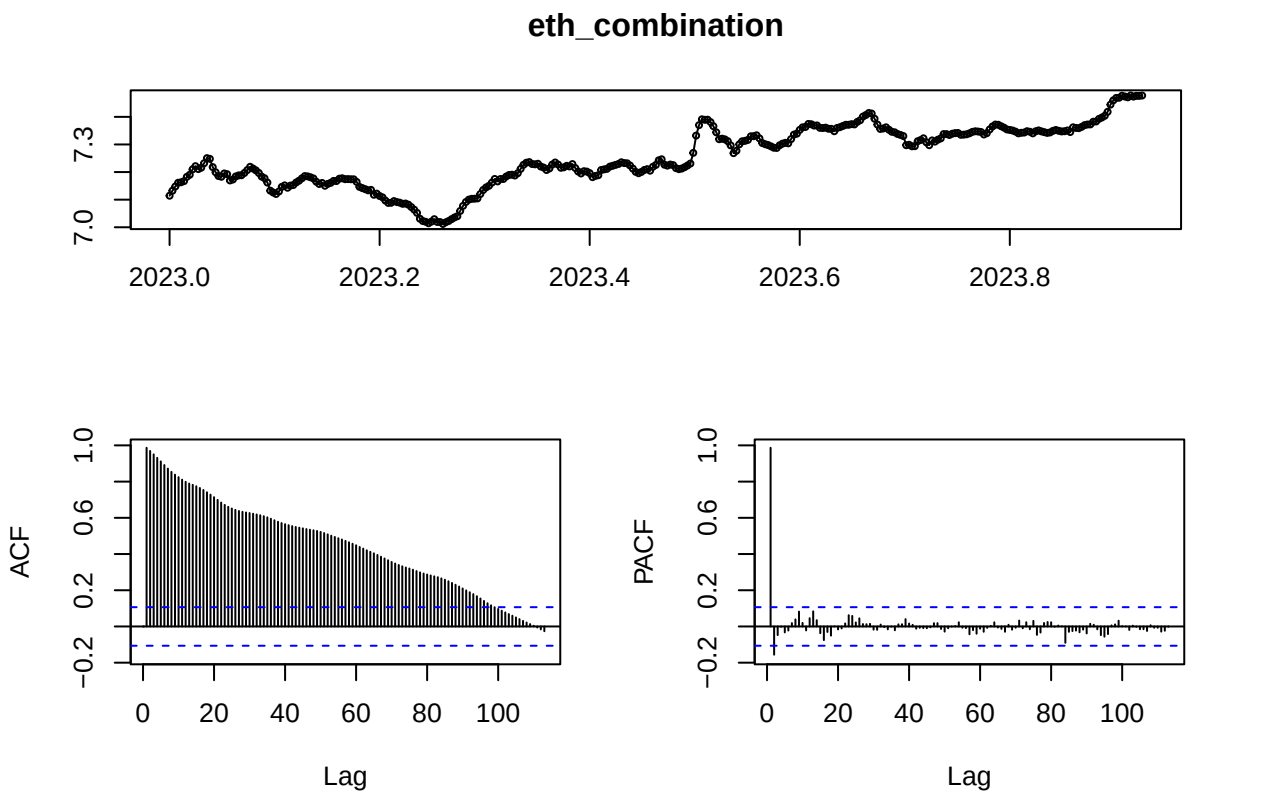
Using Facebook's Prophet model, we find that our forecast is overestimated. We see that the forecast predicts a strong upwards linear trend with minimal seasonality or cycles. Furthermore, we report a RMSE of 0.8600636 between our forecast and the testing dataset.

(g) Forecast Combination

```
# model forecast
eth_combination <- (eth_arimaforecast[["mean"]] + eth_etsforecast[["mean"]] +
                      eth_hwforecast[["mean"]] + eth_nnetarforecast[["mean"]])/4
autoplot(eth_ts) +
  autolayer(eth_etsforecast, series="ETS", PI=FALSE) +
  autolayer(eth_arimaforecast, series="ARIMA", PI=FALSE) +
  autolayer(eth_nnetarforecast, series="NNAR", PI=FALSE) +
  autolayer(eth_hwforecast, series="Holt-Winters", PI=FALSE) +
  autolayer(eth_combination, series="Combination") +
  xlab("Year") + ggtitle("Combination Forecast for ETH")
```




```
# model diagnostics
tsdisplay(eth_combination)
```




```
accuracy(eth_combination, eth_test)
```

```
##                    ME      RMSE       MAE      MPE     MAPE      ACF1 Theil's U
## Test set 0.2115438 0.2508836 0.2146502 2.818271 2.861565 0.9764815  10.27482
```

Model	RMSE
ARIMA	0.2033819
ETS	0.3923587
Holt-Winters	0.2825643
NNETAR	0.3952222
Prophet	0.8548624
Combination	0.2508836

Following the Irrelevance Proposition, which states that we should always combine forecasts (as long as they are reasonable) unless we have infinite data and a complete information set, we decided to combine all our forecasts except the Prophet forecast to create our forecast combination. We excluded the Prophet model due to its relatively high RMSE, which indicated to us that including it will most likely disadvantage our combination forecast more than it will benefit it. Even then, it appears that our forecast combination of ARIMA, ETS, Holt-Winters and NNETAR performed worse than half of the individual forecasts it included. Thus, evaluating on the criteria of Testing RMSE, our best model is the ARIMA model, followed by the Holt-Winters, Combination, NNETAR, ETS, and Prophet.

III Conclusions and Future Work

To sum, we used six models to try to capture all of the dynamics of the price of the crypto coin Ethereum between January 1, 2017 and December 4, 2023. We started by taking the log of our data so that we could deal with changes in price instead of price itself, and then splitting our data into training and testing data, with all 2023 data as our testing data.

Our first model was an ARIMA(4,1,1) model, which indicates that there was significant autocorrelation of up to four days behind, and also of one day's lag of the error. The model also included a trend component, as indicated by I(1). The model performed well on the training set, but it did significantly worse on the testing data, so it may not be the best model to use.

Second, we fitted an ETS model to our data, and the model chose to use additive errors, damped additive trend, and no seasonality. Similarly to the ARIMA model, the ETS model did much worse on the testing data than on the training data, and forecasted an almost fully immobile model, so we may not want to use this model either.

Next, we used a Holt-Winters model, which seemed to forecast the data very well, and uses trend as well as an additive seasonal model, which is interesting because most of our other models concluded that there is no seasonality in our data. The plot of the forecast on the testing data showed that it followed the general direction of the data well, and stayed relatively close to it most of the time. The error metrics for the Holt-Winter model indicate that the testing model errors are closer to the training model errors than was the case for either the ARIMA or ETS model, so the Holt-Winters forecast may be one that we will want to consider later.

After that, we used a neural network model, NNETAR, which requires a higher level of computing power to fit a model. Our NNETAR forecast worked fairly well, staying relatively close to the testing data, while lagging behind the data in terms of general trend. According to the error metrics, the NNETAR model did very well on the testing data, so we may also want to consider this model later, in addition to the Holt-Winters model.

Next, we fit a Prophet model to our data, which is an additive model that works best with strongly seasonal data, so we would not expect it to fare particularly well on our data. We see this in the plot of the forecast, which shows an extreme jump from the last data point in the training data, which does not reflect our data well at all, since our data is continuous, and does not even have a strongly positive trend.

Finally, we combined four of our previous five models, all except Prophet, since that model performed especially poorly, to get an improved model that combines all positive aspects of those four component models. The error metrics for the combined model seem very good, and by the Irrelevance Proposition, this combined forecast should be an improvement on the individual models, so we should also consider this one.

To conclude, the Holt-Winters model, the NNETAR model, and the combined forecast worked well on our data, and while any of these would be good models to use for this data, we think that ARIMA and Holt-Winters models would be the best two to use due to their relative simplicity compared to the neural network model and the combined forecast, as well as their relatively low error metrics. To improve our models, we could try other models that do not require seasonality in the data, such as the Kalman Filter, to obtain more optimal fits to our data. Although, in doing so, we would lose much of the economic interpretability In addition, we could fine tune some of the parameters on our models, such as the number of nodes on our neural network model, and we could try models on different transformations of the data to see if any other transformations work better than on the one we used.

IV References

Coinbase, Coinbase Ethereum [CBETHUSD], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/CBETHUSD, December 4, 2023.