CryptoPredictor

Thomas D. Pellegrin

December 2023

Abstract

This report presents the methods and results from the second capstone assignment of the HarvardX PH125.9 "Data Science" course available on the online learning platform edX. The assignment used an archival dataset of the daily price of ETH, the native token of the Ethereum blockchain. The purpose of this project was to test how accurately machine learning algorithms could predict the actual price of ETH given a suite of predictors.

Introduction

The desire to reliably and accurately predict the future price of trading assets has long been a goal of quantitative finance. Countless technical analysis methods and statistical models exist that attempt to identify predictable patterns in the prices of bonds, commodities, cryptocurrencies, foreign exchange pairs, securities, etc. Some hedge funds, such as Renaissance Technologies's Medallion Fund, specialize in detecting arbitrage opportunities and market inefficiencies, which are usually short-lived and disappear once they have been taken advantage of.

It is not the ambition of this project to invent some novel way to predict future ETH prices. If such a goal was within reach, this author would be retiring to his own private island rather than pursuing online certifications in data science. Instead, the intent of this assignemnt is to demonstrate enough proficiency in the setup of machine learning algorithms to meet the requirements of the second assignment of the HarvardX PH125.9 "Data Science" course.

Methods

Data preparation

An R environment was set up with the following libraries loaded. The Technical Trading Rules (*TTR*) library was selected for its convenient ability to create technical indicators from a price time series.

A time series set of historical Ethereum token (ETH) prices was downloaded from the CoinMarketCap website on December 1, 2023.

```
# Load the data
# Downloaded from https://coinmarketcap.com/currencies/ethereum/historical-data/
eth <- read_delim(
    file       = "eth.csv",
    delim      = ";",
    col_types  = c("TTTTnnnnnnT"),
    # col_select = c(1, 8:9)
  )

# Describe the data
lapply(eth, class) |> as.data.frame() |> slice(1) |> kable()
```

The data consist of 3,038 observations of ETH's daily *open*, *high*, *low*, and *close* prices in U.S. dollars for dates ranging from 2015-08-07 to 2023-11-30, trading *volume*, and *market capitalization*. Because cryptocurrencies trade continuously, the closing price is simply the last price of the day, recorded at 23:59:59 GMT. Decision was made to pick this closing price as the response variable. Furthermore, the date and trading volume are likely predictors of price, and were thus kept as well. Other columns were dropped.

Next, technical indicators were calculated and added as variables to the dataset as they might be predictors of the price too. The *TTR* library's standard functions for *simple moving average* (SMA), *exponential moving average* (EMA), *relative strength index* (RSI) were used with 7-, 14-, and 21-day moving windows, respectively. The NAs created in the first rows of the dataset from creating those moving averages were removed. Lastly, the date was converted to a continuous index variable starting at 0 for the first observation.

```
eth <- eth |>
  select(timeOpen, close, volume) |>
  rename(price = close) |>
  mutate(
    timeOpen  = as.Date(timeOpen),
    SMA7  = SMA(price, n =  7),  # Simple Moving Average
    EMA7  = EMA(price, n =  7),  # Exponential Moving Average
    RSI7  = RSI(price, n =  7),  # Relative Strength Index
    SMA14 = SMA(price, n = 14),  # Simple Moving Average
    EMA14 = EMA(price, n = 14),  # Exponential Moving Average
    RSI14 = RSI(price, n = 14),  # Relative Strength Index
    SMA21 = SMA(price, n = 21),  # Simple Moving Average
    EMA21 = EMA(price, n = 21),  # Exponential Moving Average
    RSI21 = RSI(price, n = 21)   # Relative Strength Index
  ) |>
  na.omit(eth) |> # Remove NAs
  mutate(index = as.numeric(timeOpen - min(timeOpen)), timeOpen = NULL) # Index
```

After treatment, the data were found to contain 3,017 observations of 12 variables, of which one is a response variable (*price*) and the rest are predictor variables.

Data partition

Next, the data were partitioned into training and testing data sets. The latter is to be used only for the final validation of the models. R's random number generator seed was set to a fixed value for reproducibility of the results.

```
# Set a seed for reproducibility
suppressWarnings(set.seed(1, sample.kind = "Rounding"))

# Partition the data. Test set will be 10% of data
index <- createDataPartition(
  y     = eth$index,
  times = 1,
  p     = 0.1,
  list  = FALSE
)

# Build the training and test data sets
eth_train <- eth[-index, ]
eth_test  <- eth[ index, ]
```

Model preparation

The next step was to prepare the machine learning algorithms for training. The *caret* (Classification and Regression Training) library was used. Parameters were set for a 10-fold cross-validation with a partition of 50% between training and validation data sets, all within the *eth_train* training set from earlier; the final holdout *eth_test* set remains unseen at this point.

Four different machine learning algorithms were selected based on a brief literature review of typical models used for price predictions. They are:

1. The *Generalized Boosted Regression Modeling* (gbm) method and its eponymous R library.
2. The *Generalized Linear (regression) Model* (glmnet) with regularization via penalized maximum likelihood. For this one, a combinatorics product of alpha and lambda values were set up in a search grid to allow for model tuning.
3. The *Linear (regression) Model* (lm).
4. The *Random Forest* (rf) method and its eponymous R library.

```
# Training control
trControl <- trainControl(
  method        = "cv",
  number        = 10,
  p             = .5,
  search        = "grid",
  allowParallel = TRUE,
  verboseIter   = TRUE
)

# Tuning grid for the glmnet model
tuneGrid_glmnet <- expand.grid(
  .alpha  = seq(0, 1,     length.out = 100),
  .lambda = seq(0.001, 5, length.out = 100)
)

# Tuning grid for the gbm model
tuneGrid_gbm <- expand.grid(
  interaction.depth = c(1, 3, 5),         # Max depth of trees
  n.trees           = c(50, 100, 150),    # Number of trees
  shrinkage         = c(0.01, 0.1, 0.3), # Learning rate
  n.minobsinnode    = c(10, 20)           # Minimum number of observations
)
```

Model training

Next, the four models were trained against the re-partitioned *eth_train* dataset using cross-validation as detailed above. The *caret* package takes care of tuning the parameters to minimize the residual error.

```
# Train the models
train <- list(
  gbm = train(
    price ~ .,
    data       = eth_train,
    method     = "gbm",
    trControl  = trControl,
    tuneGrid   = tuneGrid_gbm
  ),
  glmnet = train(
    price ~ .,
    data       = eth_train,
    method     = "glmnet",
    trControl  = trControl,
    tuneGrid   = tuneGrid_glmnet
  ),
  lm = train(
    price ~ .,
    data       = eth_train,
    method     = "lm",
    trControl  = trControl
    ),
  rf = train(
    price ~ .,
    data       = eth_train,
    method     = "rf",
    trControl  = trControl,
    tuneLength = 50L
  )
)
```

Model evaluation

Lastly, the performance of each trained model was evaluated by predicting prices for the previously-unseen *eth_test* data set. The *lapply* function was used to run the prediction against every model defined earlier.

Where negative prices were predicted due to the early historical prices of ETH hovering just above zero, they were replaced with a value of one cent. The motivation for this is that (i) is it nonsensical to consider a negative price for a token, and (ii) it is acceptable to constraint the output of the models as long as the rule is consistent and independent from the data in the test set.

Lastly, a fifth "virtual" model was created by taking the rowwise mean of all four models.

```
# Predict prices based on test data predictors
eth_pred <- lapply(
  X   = names(train),
  FUN = function(x) predict(train[x], newdata = eth_test)
) |>
  as.data.frame() |>
  # In case a model predicts negative prices, set the floor to one cent
  mutate_all(list(~ replace(., . < 0, 0.01))) |>
  # Rowwise mean of all models
  mutate(mean = rowMeans(across(everything()))) |>
  # Add the index for plotting
  mutate(index = eth_test$index)
```

The results from the predictions were then measured using the Root Mean Square Error (RMSE):

```
# Display the results
res <- lapply(X = eth_pred |> select(-index), FUN = function(x) postResample(obs = x, eth_test$price))

print(res)
```

```
## $gbm
##       RMSE   Rsquared        MAE
## 29.5469809  0.9991393 16.2998833
##
## $glmnet
##       RMSE   Rsquared        MAE
## 44.6187427  0.9980512 25.7539599
##
## $lm
##       RMSE   Rsquared        MAE
## 25.5774151  0.9993563 14.7706525
##
## $rf
##       RMSE   Rsquared        MAE
## 37.5808982  0.9986226 14.3665871
##
## $mean
##       RMSE   Rsquared        MAE
## 29.2273645  0.9991622 14.8036673
```

The model with the lowest RMSE was found to be the linear regression:

```
# Name of the model with the lowest RMSE
win <- res |>
  as.data.frame() |>
  t() |>
  as.data.frame() |>
  filter(RMSE == min(RMSE))

# Print name
print(rownames(win))
```

```
## [1] "lm"
```

Lastly, the results were plotted on top of the original time series data. The training data are shown here as a light grey line. The test data (10% of the training data, randomly selected with a set seed) are shown as red dots. The best model's predictions are shown as blue dots.

Note that a scaled-up, high-resolution version of this plot is visible at https://github.com/TheAviationDoctor/HarvardX-PH125.9x/tree/main/CryptoPredictor/plot.png.

```
# Plot the results
p <- ggplot() +
  geom_line( data = eth_train, aes(x = index, y = price),              color = "gray") +
  geom_point(data = eth_test,  aes(x = index, y = price),              color = "red",  alpha = .25) +
  geom_point(data = eth_pred,  aes(x = index, y = get(rownames(win))), color = "blue", alpha = .25) +
  labs(x = "Time", y = "Price in USD")
```

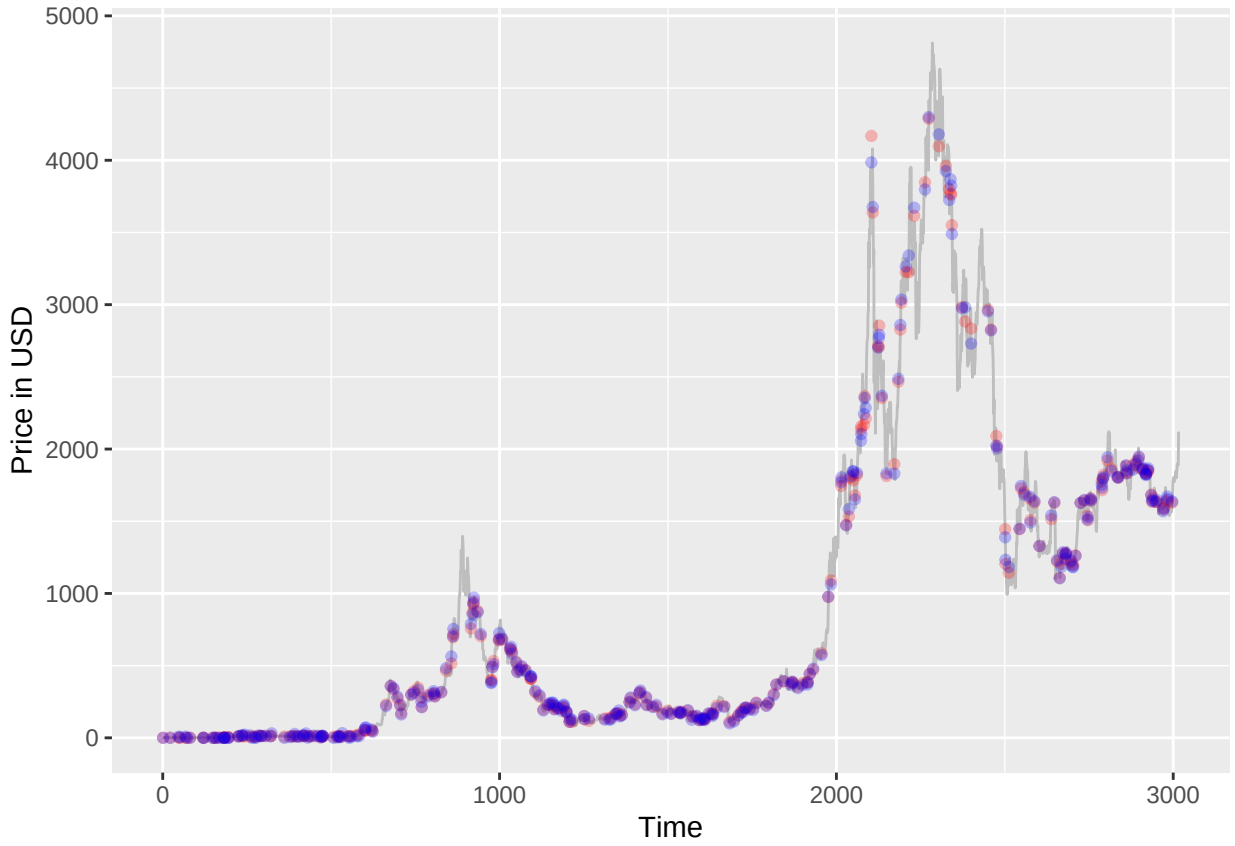

Figure 1: Model results

Results

The plot shows that the classic linear regression algorithm returns plausible and accurate price predictions based solely on a combination of *index* (i.e., position in a time series), *trading volume*, and *technical indicators* including the SMA, EMA, and RSI for three time windows. The resulting RMSE stands at 25.5774151, the R-squared at 0.9993563, and the Mean Absolute Error (MAE) at 14.7706525. The predictions, in red on the plot, follow the actual prices from the test data set (in blue) closely despite the intense longitudinal volatility experienced by a crypto asset such as ETH.

The model built above is merely a proof of concept. It is likely that additional refinements, either in the form of variants of random forest or further tweaking of the model's hyperparameters, would yield an even superior performance. Additional predictors could also supplement the model with (for example) social signals about market sentiment.

Conclusion

The purpose of this assignment was to demonstrate the feasibility of using a machine learning algorithm to obtain reasonable price predictions for a volatile cryptocurrency. Several algorithms were trained and validated, and the lm model outperformed all others by returning plausible values for the response variable compared to the testing data set. Future research can expand these results by testing alternative models, tune the hyperparameters further, and add more predictors.