Bitcoin candlestick predictions using lagged features and machine learning algorithms in R

Training machine learning algorithms to predict the next candlestick's direction of the Bitcoin chart using a set of lagged features

Sandoche Adittane

2025-04-23

Abstract

This report explores how to achieve the best accuracy in predicting the next candlestick of the Bitcoin chart using the previous ones. It compares different algorithms: Generalized Linear Model, Decision Tree, Random Forest, KNN, and Gradient Boosting, considering different numbers of lagged features. We ended up getting an accuracy of 0.5457 with a fine tuned model using 1 lagged features and the Gradient Boosting algorithm. This project is part of the 'Data Science: Capstone' module of HarvardX PH125.9x on the edx platform.

Contents

List of Figures

List of Tables

1 Overview

In this study we will try to predict the direction of the next candlestick of the Bitcoin chart. Before starting, it's important to understand: what are Bitcoin and candlesticks, and the goal of this study.

1.1 Introduction to Bitcoin

In recent years, Bitcoin (BTC) has been gaining attention not only from retail investors but also from institutional investors. In 2024 we've seen the emergence of spot Bitcoin exchange-traded funds (ETF) from institutions such as BlackRock, VanEck, Grayscale [1]. With a market capitalization of about 1.68 trillion in dollars at the time of writing, Bitcoin started as a peer-to-peer currency, a free alternative to centralized currencies controlled by central banks. Now it is often used as an investment, a store of value and even considered strategic reserve assets by some countries [2], [3].

Bitcoin owes its decentralization to its data structure, the blockchain — a chain of blocks that contains transactions, and to its consensus, the proof of work. Without going too much into details, it makes Bitcoin a currency that does not rely on a centralized server. Proof of work is a cryptographic algorithm that enables a competition between Bitcoin servers (called nodes) to decide which transactions will be part of the next block added to the blockchain. They go through a process called mining where nodes have to use their computing power to find a number called nonce. This computing power is called the hashrate. The node that succeeds at "mining" successfully gets rewarded for that [4].

Bitcoin is defined by its source code, and that's quite fascinating. Its ledger is visible and publicly available, which gives a lot of data available to analyze. Moreover, unlike stocks, BTC can be traded any time; there are no opening or closing hours, the Bitcoin market never stops and it is very easy for anyone to buy and sell Bitcoin. Those are two reasons worth studying Bitcoin's candlestick charts instead of other assets.

1.2 What are candlesticks?

Let's talk about the candlestick. The price of assets such as Bitcoin is described by a time series of candlesticks, each defined by an opening price, a close price, a high, and a low, also called OHLC. A candlestick can be "up" (often green and also called bullish) or "down" (often red and also called bearish). You can see this visually with the following figure.

The candlestick chart is defined as a time series of candles; each candle is defined at a specific time and has a specific time duration. We will explain this in more detail in the exploratory analysis section.

1.3 Candlesticks pattern

Some technical analysts study candlestick patterns to try to predict the direction of the market; this field is known as candlestick pattern analysis. It consists of knowing a set of patterns and their likely outcomes.

Traders look for specific patterns like "Doji", "Hammer", "Engulfing patterns", and many others to make trading decisions. Each pattern has a specific interpretation based on market psychology and historical tendencies [6].

If such patterns really exist, we believe that machine learning algorithms should be able to detect them.

1.4 Goal of the study

The goal of the study is to find a model able to predict the direction of a candlestick using N previous candles. This number N will also be part of the research. We will have to not only find N but also find the best features to achieve the best accuracy.

1.5 Applications

Why does the direction of a candlestick matter? Because being able to predict the direction of the next candle could enable traders to buy and sell on the spot market when the predicted candle is green. Also,

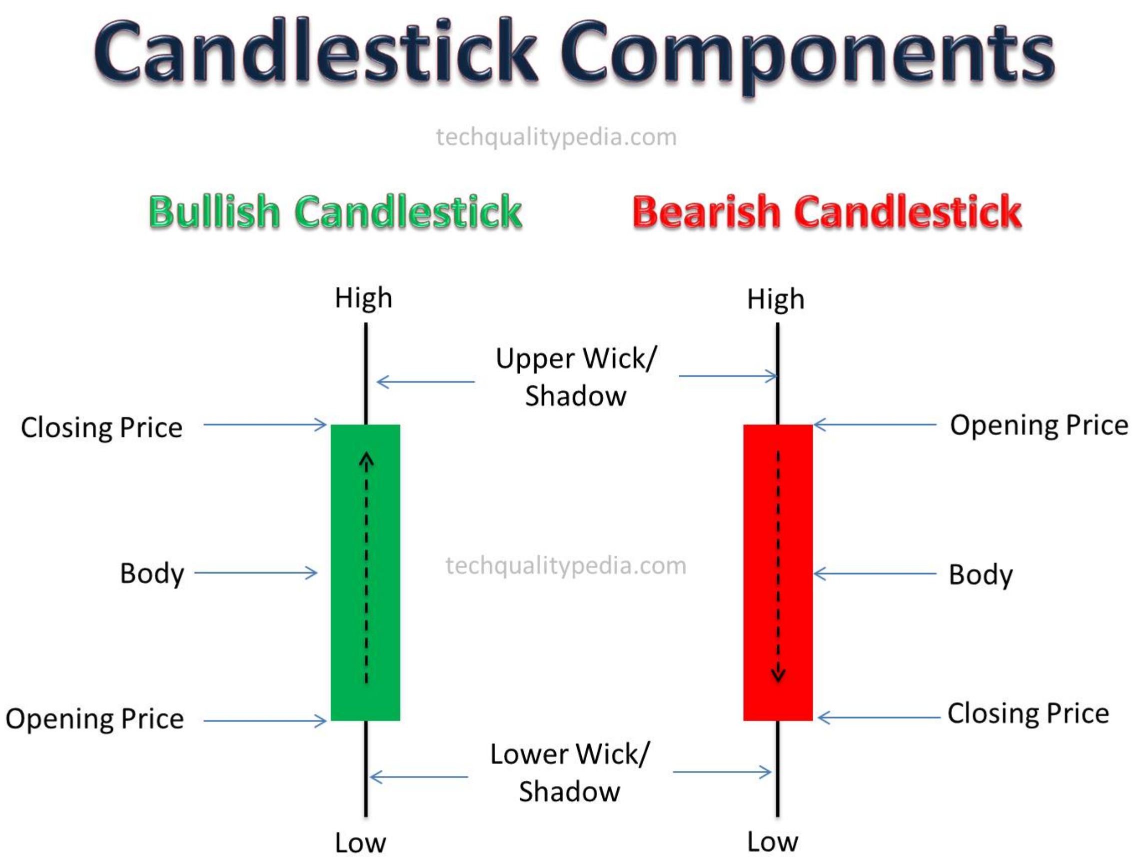


Figure 1: Candlestick components [5]

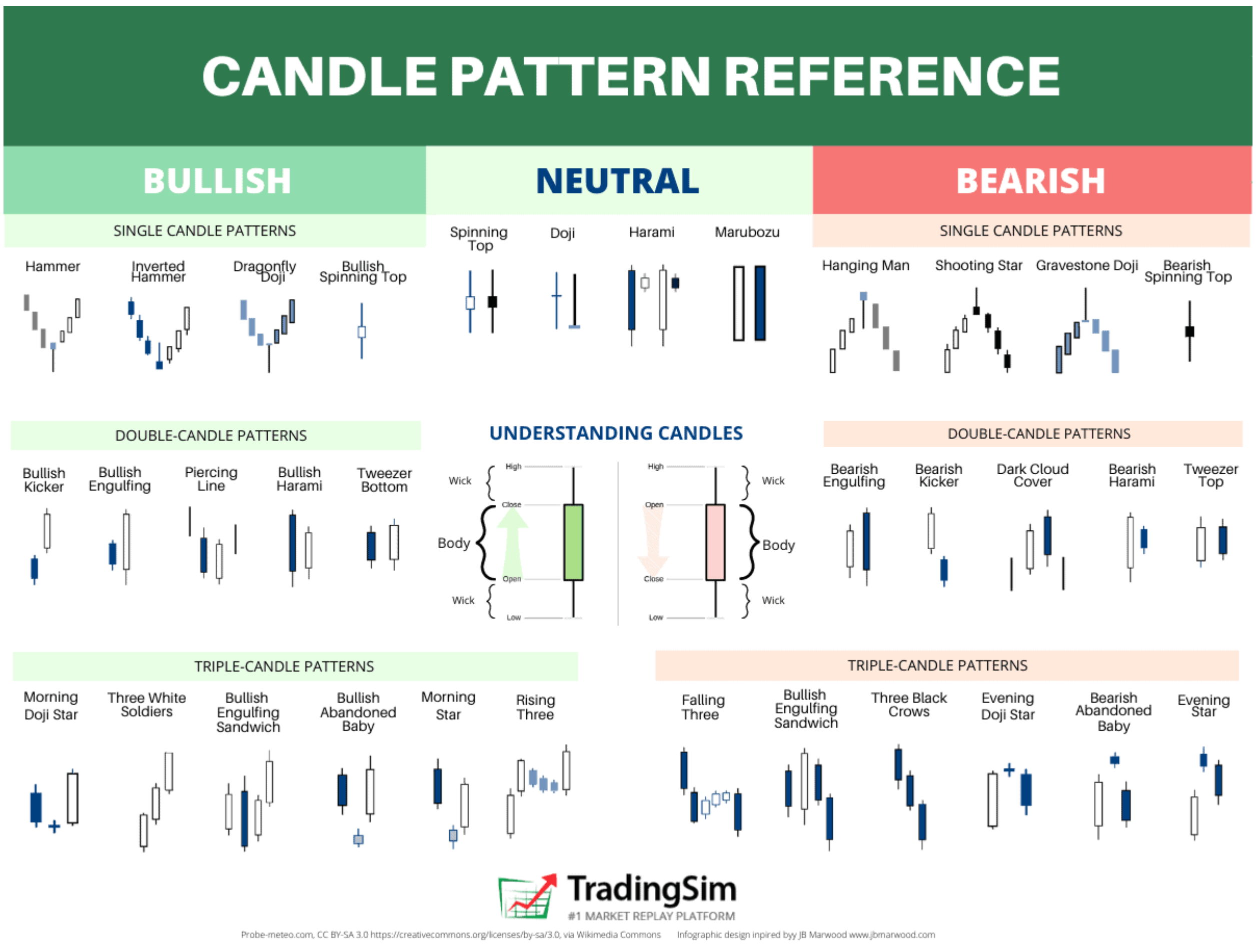


Figure 2: Common candlestick patterns guide [6]

perpetual futures traders can go both ways; they can “long” when the prediction says “up” and “short” when the predictions says “down”.

2 Exploratory data analysis

In this section, we will see what are the different datasets available. We will see what features are available to train the different models. Then we will prepare the data, verify it, and choose which machine learning algorithms we are going to train.

2.1 Data sets

In order to conduct this study, we used the historical rates for the trading pair BTC-USD using the Coinbase API as the main dataset.

We used the following global variables for the full project:

```r
trading_pair <- "BTC-USD"
start_date <- "2024-01-01"
end_date <- "2025-03-29"
candlestick_period <- 3600
set.seed(1)
```

The timeframe covers the period from January 2024 to March 2025, up until the day this study commenced. Note that since January 2024, Bitcoin ETF has officially been approved. The period `candlestick_period <- 3600` is the time of a candle; the candle closes 1h after it starts. Which means we have 24 candles per day [7].

We chose these settings to have a dataset of around 10,000 candles. Also, since the Bitcoin ETF has been approved, the market may have taken on a different dynamic than in previous years.

Let’s see how the dataset looks like.

Table 1: Overview of the BTC-USD candlestick dataset

time	low	high	open	close	volume
2024-01-01 00:00:00	42261.58	42543.64	42288.58	42452.66	379.1973
2024-01-01 01:00:00	42415.00	42749.99	42453.83	42594.68	396.2019
2024-01-01 02:00:00	42488.03	42625.68	42594.58	42571.32	227.1412
2024-01-01 03:00:00	42235.00	42581.26	42571.32	42325.11	306.0057
2024-01-01 04:00:00	42200.00	42393.48	42325.10	42389.77	296.2336
2024-01-01 05:00:00	42175.65	42396.09	42389.78	42231.47	188.1280

We have 10,873 entries in our candlestick dataset. As described in the overview, it contains the OHLC data, timestamp, and the volume of each candle.

Bitcoin is used by 3 types of users:

- Traders — they are interested by the price and make profit
- Users — using the currency to do payments or to transfer money around the world
- Miners — they mine Bitcoin to sell it; their interest is that the price of Bitcoin is higher than the cost of mining Bitcoin

Keeping this in mind, we tried to find other datasets that could represent each type of user that could eventually help in our predictions, and we picked the following:

- Fear and greed index — represents the overall mood of the market (traders)

- Hash-rate — defines the overall mining power (miners)
- Average block size — the higher it is, the more transactions are happening (users)
- Number of transactions — defines the activity of the network (users)
- Number of unspent transaction outputs (UTXO) — defines how many addresses contain Bitcoin, and reflects the network activity (users)

Table 2: Overview of the BTC fear and greed index dataset

value	value_classification	timestamp
26	Fear	2025-03-29
44	Fear	2025-03-28
40	Fear	2025-03-27
47	Neutral	2025-03-26
46	Fear	2025-03-25
45	Fear	2025-03-24

This dataset is a time series of the daily fear and greed index; it is a value between 0 and 100, 0 being the most fearful and 100 being the most greedy. The dataset contains 453 entries.

The blockchain data (hash rate, average block size, number of transactions, and UTXO count) was sourced from Blockchain.com Explorer [8], while the fear and greed index was obtained from Alternative.me [9].

Table 3: Overview of the BTC hash rate dataset

timestamp	hash_rate
2024-01-01	501122294
2024-01-02	509303882
2024-01-03	505213088
2024-01-04	520042217
2024-01-05	545098332
2024-01-06	538450791

This dataset is a time series of the daily hash rate; it is a value in TH/s. The dataset contains 454 entries.

Table 4: Overview of the BTC average block size dataset

timestamp	avg_block_size
2024-01-01	1.653640
2024-01-02	1.718455
2024-01-03	1.771466
2024-01-04	1.782402
2024-01-05	1.774551
2024-01-06	1.847959

This dataset is a time series of the daily average block size; it is a value in megabytes. The dataset contains 454 entries.

Table 5: Overview of the BTC number of transactions dataset

timestamp	n_transactions
2024-01-01	657752
2024-01-02	367319
2024-01-03	502749
2024-01-04	482557
2024-01-05	420884
2024-01-06	382140

This dataset is a time series of the daily number of transactions. The dataset contains 454 entries.

Table 6: Overview of the BTC UTXO count dataset

timestamp	utxo_count
2024-01-01	135878807
2024-01-02	136204295
2024-01-03	136536575
2024-01-04	136871780
2024-01-05	137209298
2024-01-06	137552822

This dataset is a time series of the daily number of UTXOs; it is the count of UTXOs in the network. The dataset contains 454 entries. We had to group by timestamp because some dates had more than one value.

As we can see, `fear_and_greed_index` seems to be missing one data point. We will fix this in the following sections.

We have different datasets that we will use for the predictions, but we are still missing one important piece of data used by traders: technical analysis indicators.

Based on a previous research and blog post I wrote [10], we decided to include a few indicators that are very common in trading :

- Moving average convergence divergence (MACD)
- Rate of change (ROC)
- Bolinger Bands (BB)
- Relative Strenght Index (RSI)

Before preparing the dataset, let's see what features we can extract from the OHLC candlestick data.

2.2 Features

Our candlesticks dataset from Coinbase gives us values that are based on the price of Bitcoin, but using these raw absolute values directly in a machine learning algorithm will be difficult since the price always fluctuates. Therefore, we have to think about what defines the candlesticks we have seen above?

If we isolate a candle, we can see the following features:

- Size of the body
- Size of the upper shadow
- Size of the lower shadow

- Direction of the candle (up or down)
- Closing price
- Volume

We are now ready to prepare the dataset for the study.

2.3 Preparation

The preparation of the dataset is performed by the following function:

```r
enhance_dataset <- function(candles_data, fear_and_greed_index_data, hash_rate_data,
↪  average_block_size_data, n_transactions_data, utxo_count_data) {
  candles_enhanced <- candles_data %>%
    mutate(date_only = as.Date(time)) %>%
    left_join(fear_and_greed_index_data, by = c("date_only" = "timestamp")) %>%
    left_join(hash_rate_data, by = c("date_only" = "timestamp")) %>%
    left_join(average_block_size_data, by = c("date_only" = "timestamp")) %>%
    left_join(n_transactions_data, by = c("date_only" = "timestamp")) %>%
    left_join(utxo_count_data, by = c("date_only" = "timestamp")) %>%
    mutate(
      body_size = abs(close - open),
      upper_shadow_size = high - pmax(close, open),
      lower_shadow_size = pmin(close, open) - low,
      direction = ifelse(close > open, "up", "down"),
    ) %>%
    tq_mutate(
      select = close,
      mutate_fun = ROC,
      n = 14,
      col_rename = "roc"
    ) %>%
    tq_mutate( #
    ↪  https://www.keenbase-trading.com/find-best-macd-settings/#t-1719588154943
      select = close,
      mutate_fun = MACD,
      nFast = 12,
      nSlow = 26,
      nSig = 9,
      col_rename = c("macd", "signal")
    ) %>%
    tq_mutate(
      select = close,
      mutate_fun = RSI,
      n = 14,
      col_rename = "rsi"
    ) %>%
    tq_mutate(
      select = close,
      mutate_fun = BBands,
      n = 20,
      sd = 2,
      col_rename = "bband"
    )

  candles_enhanced
```

```r
}

candles_enhanced <- enhance_dataset(candles, fear_and_greed_index, hash_rate,
↪  average_block_size, n_transactions, utxo_count)
names(candles_enhanced)
```

```
##  [1] "time"               "low"                "high"
##  [4] "open"               "close"              "volume"
##  [7] "date_only"          "value"              "value_classification"
## [10] "hash_rate"          "avg_block_size"     "n_transactions"
## [13] "utxo_count"         "body_size"          "upper_shadow_size"
## [16] "lower_shadow_size"  "direction"          "roc"
## [19] "macd"               "signal"             "rsi"
## [22] "dn"                 "mavg"               "up"
## [25] "pctB"
```

We now have a table containing all the features discussed above. Before adding the lagged features, let's explore our dataset.

First, let's check for NAs.

```r
na_indexes <- which(apply(candles_enhanced, 1, function(x) any(is.na(x))))
knitr::kable(head(candles_enhanced[na_indexes, ] %>%
    select(direction, roc, macd, signal, rsi, mavg, pctB, value)),
    format = "simple", caption = " NAs related to TA indicators and the Fear & Greed
    ↪  index - Head")
```

Table 7: NAs related to TA indicators and the Fear & Greed index - Head

direction	roc	macd	signal	rsi	mavg	pctB	value
up	NA	NA	NA	NA	NA	NA	65
up	NA	NA	NA	NA	NA	NA	65
down	NA	NA	NA	NA	NA	NA	65
down	NA	NA	NA	NA	NA	NA	65
up	NA	NA	NA	NA	NA	NA	65
down	NA	NA	NA	NA	NA	NA	65

```r
knitr::kable(tail(candles_enhanced[na_indexes, ] %>%
    select(direction, roc, macd, signal, rsi, mavg, pctB, value)),
    format = "simple", caption = " NAs related to TA indicators and the Fear & Greed
    ↪  index - Tail")
```

Table 8: NAs related to TA indicators and the Fear & Greed index - Tail

direction	roc	macd	signal	rsi	mavg	pctB	value
down	0.0012887	-0.1426441	-0.1844117	47.95418	66866.20	0.6227201	NA
up	0.0007626	-0.1086808	-0.1692655	52.68220	66893.00	0.8544387	NA
up	0.0012911	-0.0721331	-0.1498390	54.72628	66919.90	0.9343443	NA
down	0.0024273	-0.0527284	-0.1304169	51.93706	66944.51	0.7779949	NA
down	-0.0002377	-0.0424873	-0.1128310	50.40502	66960.23	0.6689238	NA
down	0.0005901	-0.0375443	-0.0977736	49.39916	66970.03	0.5904000	NA

We can see in the table above that there are 2 types of NAs:

1. Technical analysis indicators
2. Fear and greed index

The technical analysis indicators are located at the beginning of the table. This is expected, as calculating these indicators requires a certain number of previous values. Removing rows with these NAs will be sufficient.

As for the missing fear and greed index value, we can notice that we are missing the values for the day `2024-10-26`, so we will manually add the average of the values from the day before and the day after.

It's important to do this to ensure coherent lagged values.

```
date_na <- as.Date("2024-10-26")
fear_and_greed_index_date_before_na <- fear_and_greed_index %>%
    filter(timestamp == as.Date("2024-10-25"))
fear_and_greed_index_date_after_na <- fear_and_greed_index %>%
    filter(timestamp == as.Date("2024-10-27"))
fear_and_greed_value_date_na <- mean(c(fear_and_greed_index_date_before_na$value,
    fear_and_greed_index_date_after_na$value))

fear_and_greed_index_corrected <- fear_and_greed_index %>%
    bind_rows(tibble(timestamp = date_na, value = fear_and_greed_value_date_na,
        value_classification = "Greed"))

fear_and_greed_index %>%
    filter(timestamp == date_na) %>%
    nrow()
```

```
## [1] 0
```

```
fear_and_greed_index_corrected %>%
    filter(timestamp == date_na) %>%
    nrow()
```

```
## [1] 1
```

Let's check the NAs again.

```
candles_enhanced_cleaned <- enhance_dataset(candles, fear_and_greed_index_corrected,
    hash_rate, average_block_size, n_transactions, utxo_count)
na_indexes <- which(apply(candles_enhanced_cleaned, 1, function(x) any(is.na(x))))
knitr::kable(tail(candles_enhanced_cleaned[na_indexes, ] %>%
    select(direction, roc, macd, signal, rsi, mavg, pctB, value)),
    format = "simple", caption = "NAs of the dataset cleaned")
```

Table 9: NAs of the dataset cleaned

direction	roc	macd	signal	rsi	mavg	pctB	value
down	0.0625693	2.103478	NA	83.37869	43483.53	1.0029836	71
down	0.0588336	2.115233	NA	76.65897	43616.84	0.8874993	71
down	0.0549824	2.098952	NA	76.59357	43745.10	0.8455368	71
down	0.0560958	2.060150	NA	76.37608	43871.11	0.8093231	71
up	0.0604901	2.058537	NA	78.83895	44011.84	0.8389528	71
up	0.0604520	2.086971	NA	81.02236	44169.85	0.8654968	71

We can now see that the only remaining NAs are the missing TA values at the beginning, which can be removed easily.

```
candles_enhanced_cleaned_no_na <- candles_enhanced_cleaned %>%
    drop_na()
sum(is.na(candles_enhanced_cleaned_no_na))
```

```
## [1] 0
```

2.4 Visual analysis

First, let's plot the data for visual verification.

Below are the plots for the different Technical Analysis indicators.

Comparing these charts with data from TradingView confirms their correctness.

Let's now examine the distribution of "up" and "down" candles.

Table 10: Distribution of up and down candles

up	down	total	up_percentage	down_percentage
5538	5302	10840	0.5108856	0.4891144

We can notice that the distribution is not exactly 50%. With a start price of `$42261.58` and an end price of `$84311.11` within the time frame we can consider the trend of this market as bullish.

2.5 Adding lagged candles

Our study aims at predicting the direction of a candle using the previous candle's data and other features.

Therefore, we need to create a function that generates a dataset containing lagged candle data. We also created another function to directly prepare the right data.

```
add_lagged_candles <- function(enhanced_clean_dataset, n_lag) {
    dataset_with_lagged_candles <- enhanced_clean_dataset

    for (i in 1:n_lag) {
        dataset_with_lagged_candles[[paste0("body_size_lag_",
            i)]] <- lag(dataset_with_lagged_candles$body_size,
            i)
        dataset_with_lagged_candles[[paste0("upper_shadow_size_lag_",
            i)]] <- lag(dataset_with_lagged_candles$upper_shadow_size,
            i)
        dataset_with_lagged_candles[[paste0("lower_shadow_size_lag_",
            i)]] <- lag(dataset_with_lagged_candles$lower_shadow_size,
            i)
        dataset_with_lagged_candles[[paste0("direction_lag_",
            i)]] <- lag(dataset_with_lagged_candles$direction,
            i)
        dataset_with_lagged_candles[[paste0("volume_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$volume,
            i)
        dataset_with_lagged_candles[[paste0("value_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$value,
            i)
```

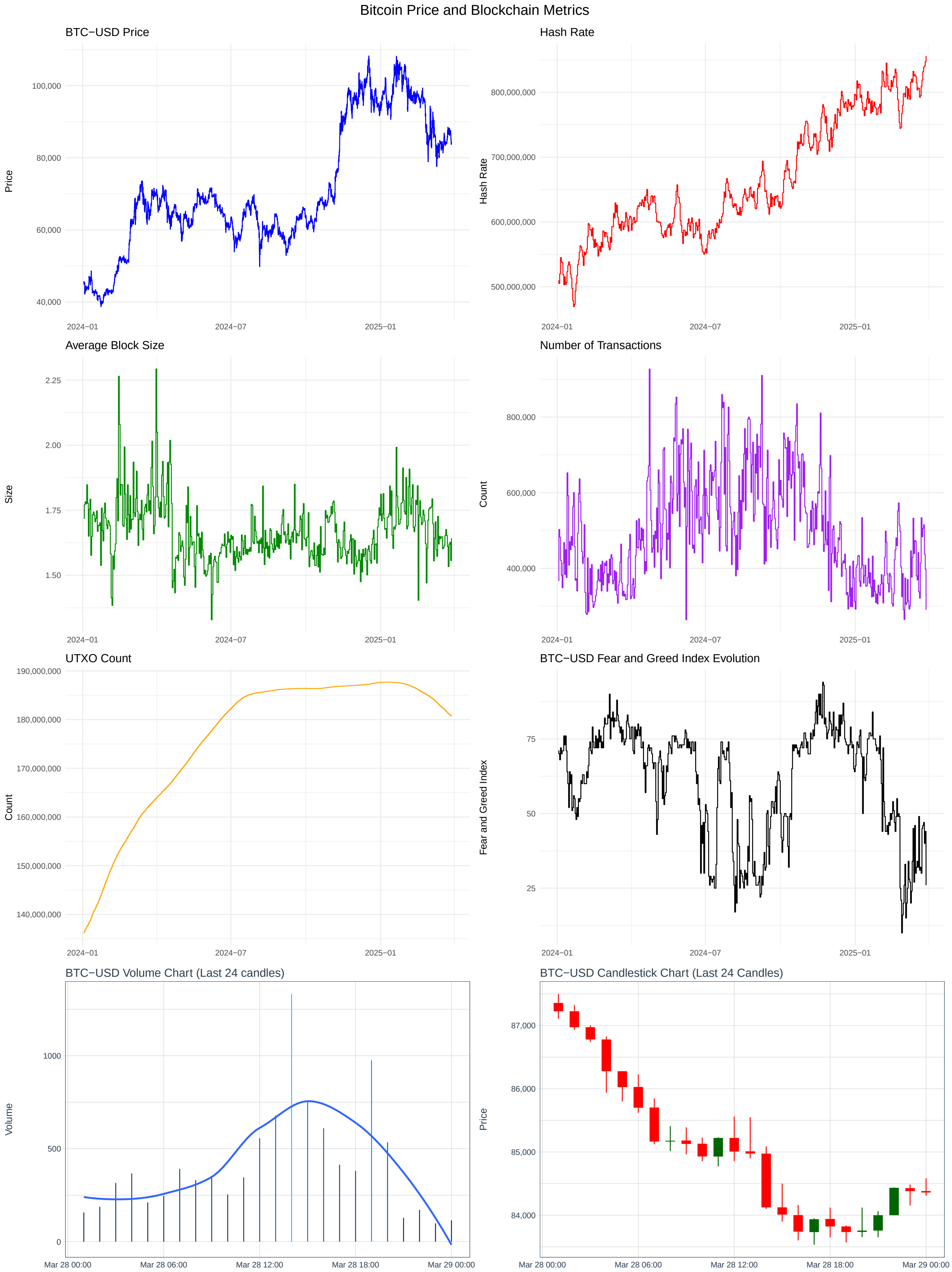


Figure 3: Visual analysis of BTC-USD data

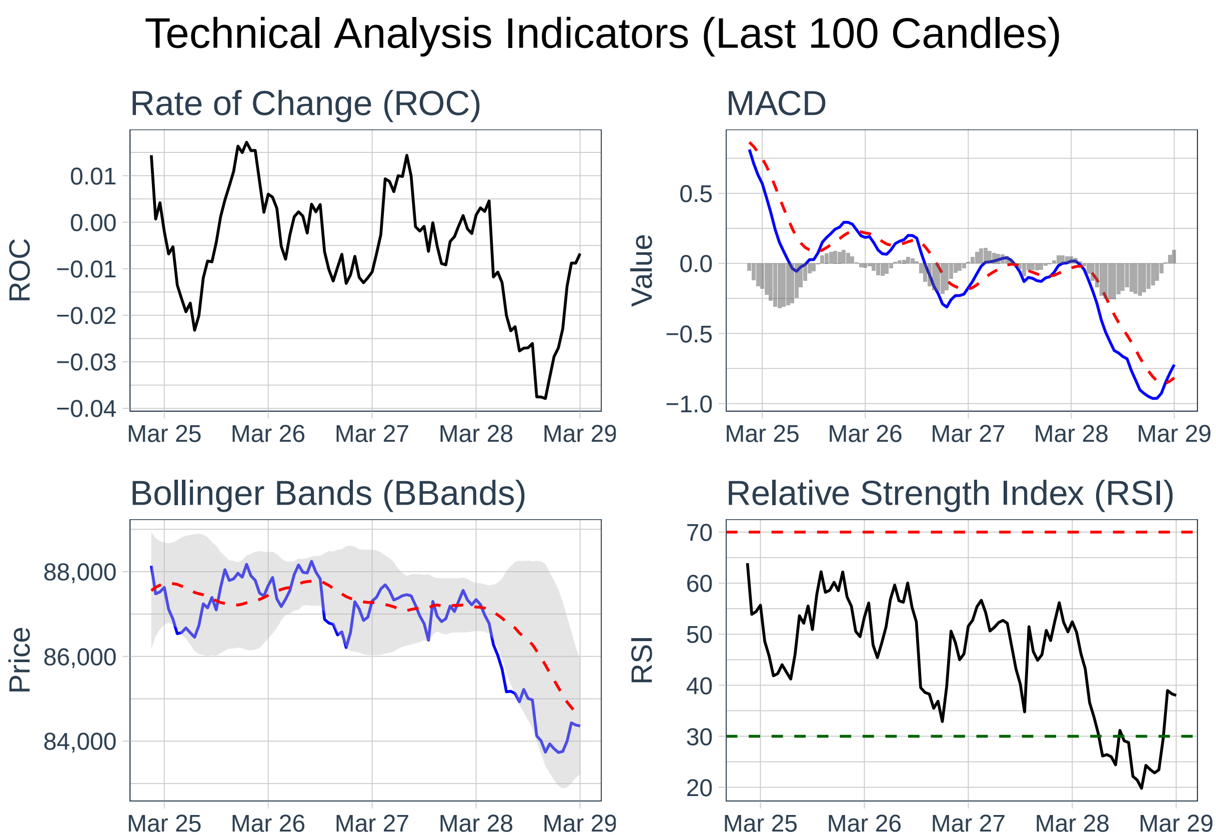


Figure 4: Technical analysis indicators of BTC-USD

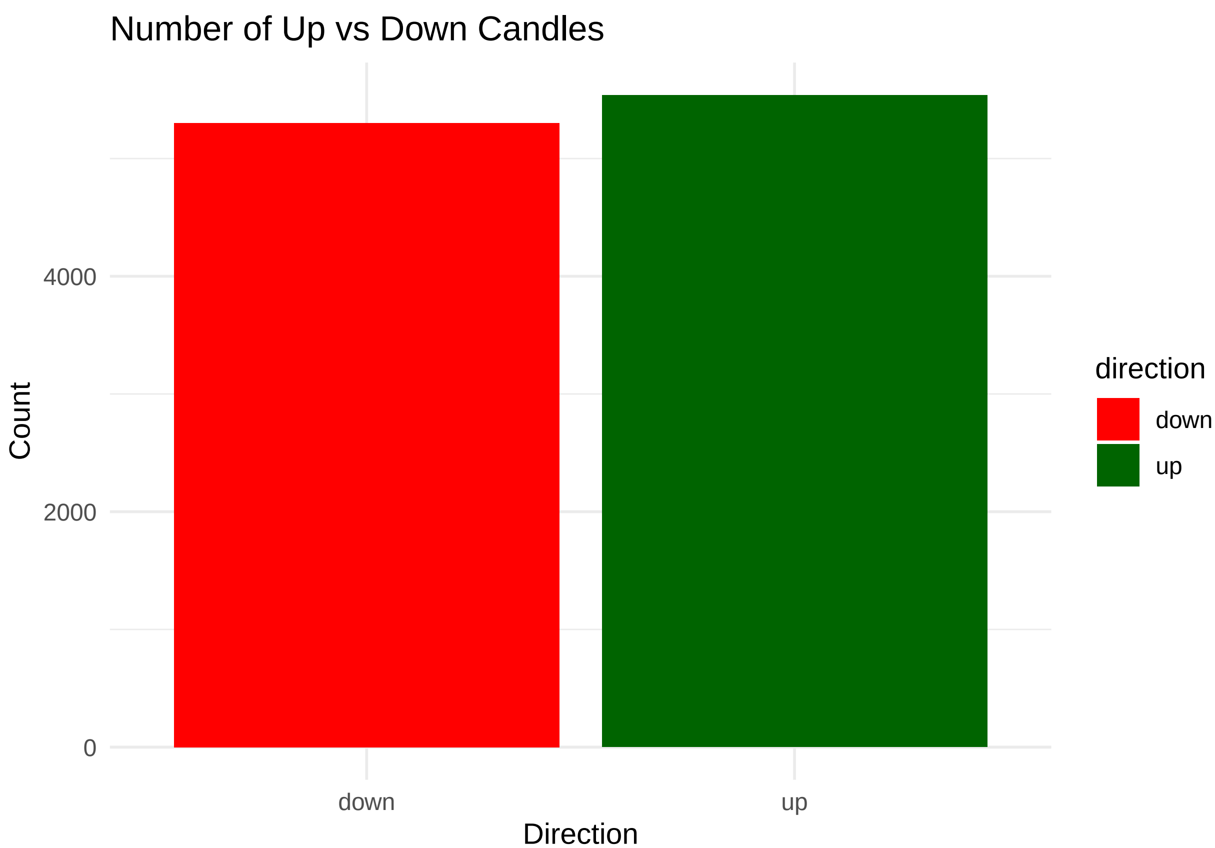


Figure 5: Distribution of up and down candles in the dataset

```
        dataset_with_lagged_candles[[paste0("close_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$close,
            i)
        dataset_with_lagged_candles[[paste0("hash_rate_lag_",
            i)]] <- lag(dataset_with_lagged_candles$hash_rate,
            i)
        dataset_with_lagged_candles[[paste0("avg_block_size_lag_",
            i)]] <- lag(dataset_with_lagged_candles$avg_block_size,
            i)
        dataset_with_lagged_candles[[paste0("n_transactions_lag_",
            i)]] <- lag(dataset_with_lagged_candles$n_transactions,
            i)
        dataset_with_lagged_candles[[paste0("utxo_count_lag_",
            i)]] <- lag(dataset_with_lagged_candles$utxo_count,
            i)
        dataset_with_lagged_candles[[paste0("open_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$open,
            i)
        dataset_with_lagged_candles[[paste0("high_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$high,
            i)
        dataset_with_lagged_candles[[paste0("low_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$low,
            i)
        dataset_with_lagged_candles[[paste0("roc_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$roc,
            i)
        dataset_with_lagged_candles[[paste0("macd_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$macd,
            i)
        dataset_with_lagged_candles[[paste0("signal_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$signal,
            i)
        dataset_with_lagged_candles[[paste0("rsi_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$rsi,
            i)
        dataset_with_lagged_candles[[paste0("up_bband_lag_",
            i)]] <- lag(dataset_with_lagged_candles$up, i)
        dataset_with_lagged_candles[[paste0("mavg_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$mavg,
            i)
        dataset_with_lagged_candles[[paste0("dn_bband_lag_",
            i)]] <- lag(dataset_with_lagged_candles$dn, i)
        dataset_with_lagged_candles[[paste0("pctB_lag_", i)]] <-
↪   lag(dataset_with_lagged_candles$pctB,
            i)
    }

    dataset_with_lagged_candles
}

prepare_dataset <- function(candles_data, fear_and_greed_index_data,
    hash_rate_data, average_block_size_data, n_transactions_data,
    utxo_count_data) {
```

```
    enhanced_clean_dataset <- enhance_dataset(candles_data, fear_and_greed_index_data,
        hash_rate_data, average_block_size_data, n_transactions_data,
        utxo_count_data)
    enhanced_clean_dataset_without_na <- enhanced_clean_dataset %>%
        drop_na()
    dataset_with_lagged_candles <- add_lagged_candles(enhanced_clean_dataset_without_na,
        15)
    dataset_with_lagged_candles_without_na <- dataset_with_lagged_candles %>%
        drop_na()
    dataset_with_lagged_candles_without_na
}
```

Using the `prepare_dataset` function allows us to directly obtain the final dataset with lagged data.

2.6 Test and training datasets

We put together the code to fix the fear_and_greed_index, prepare the datasets, and split them into train and test sets.

```
date_na <- as.Date("2024-10-26")
fear_and_greed_index_date_before_na <- fear_and_greed_index %>%
    filter(timestamp == as.Date("2024-10-25"))
fear_and_greed_index_date_after_na <- fear_and_greed_index %>%
    filter(timestamp == as.Date("2024-10-27"))
fear_and_greed_value_date_na <- mean(c(fear_and_greed_index_date_before_na$value,
    fear_and_greed_index_date_after_na$value))

fear_and_greed_index_corrected <- fear_and_greed_index %>%
    bind_rows(tibble(timestamp = date_na, value = fear_and_greed_value_date_na,
        value_classification = "Greed"))

project_dataset <- prepare_dataset(candles, fear_and_greed_index_corrected,
    hash_rate, average_block_size, n_transactions, utxo_count)

sum(is.na(project_dataset))
```

```
## [1] 0
```

```
nrow(project_dataset)
```

```
## [1] 10825
```

```
nrow(candles)
```

```
## [1] 10873
```

```
test_index <- createDataPartition(y = project_dataset$direction,
    times = 1, p = 0.2, list = FALSE)
train_set <- project_dataset[-test_index, ]
test_set <- project_dataset[test_index, ]
```

The number of rows was reduced because adding lags introduced NAs in the initial rows, which we subsequently removed. Also, we decided not to use cross-validation to reduce the training time for the different machine learning algorithms. Note that we already initiated the project using `set.seed(1)` as part of the global variables.

2.7 Machine learning algorithms

Based on preliminary research, we selected the following machine learning algorithms that seem to work well with our type of dataset [11], [12]:

- Generalized Linear Model (GLM)
- Decision Tree (DT)
- Random Forest (RF)
- K-nearest neighbor (KNN)
- Gradient boosting (GBM)

We will also compare these algorithms with a random guess baseline as a reference.

2.8 Utility functions

Since we want to compare each algorithm for a different set of features, we need a function to create the formula that will be passed to the machine learning training function.

```
create_feature_formula <- function(feature_names, n_lags) {
    features <- c()

    for (feature_name in feature_names) {
        for (i in 1:n_lags) {
            features <- c(features, paste0(feature_name, "_lag_",
                i))
        }
    }

    formula_str <- paste("direction ~", paste(features, collapse = " + "))

    as.formula(formula_str)
}


train_with_cache <- function(formula, train_set, method) {
    formula_hash <- digest::digest(formula)
    filepath <- paste0("models/", method, "_", formula_hash,
        ".rds")
    if (file.exists(filepath)) {
        model <- readRDS(filepath)
        print(paste("Model loaded from cache:", filepath))
    } else {
        start_time <- Sys.time()
        if (method == "rf") {
            model <- train(formula, data = train_set, method = "rf",
                ntree = 100)
        } else if (method == "glm") {
            model <- train(formula, data = train_set, method = "glm",
                family = "binomial")
        } else if (method == "rpart") {
            model <- train(formula, data = train_set, method = "rpart")
        } else if (method == "knn") {
            model <- train(formula, data = train_set, method = "knn",
                preProcess = c("center", "scale"), tuneGrid = data.frame(k = seq(3,
                  15, 2)))
```

```r
        } else if (method == "gbm") {
            model <- train(formula, data = train_set, method = "gbm")
        } else {
            stop("Invalid method")
        }
        end_time <- Sys.time()
        print(paste("Training time:", format(end_time - start_time,
            digits = 2)))

        saveRDS(model, filepath)
    }

    model
}

evaluate_models <- function(feature_set, test_set, lags = c(1,
    3, 5, 7, 15)) {
    # Define model types
    model_types <- c("glm", "rf", "rpart", "knn", "gbm")

    # Create a data frame to store results
    results <- data.frame(model = character(), model_type = character(),
        lag = numeric(), accuracy = numeric(), stringsAsFactors = FALSE)

    # Evaluate each model type and lag combination
    for (model_type in model_types) {
        for (lag in lags) {
            model_name <- paste0(model_type, "_model_", feature_set,
                "_lag_", lag)

            if (exists(model_name)) {
                # Get the model object
                model <- get(model_name)

                # Make predictions
                predictions <- predict(model, test_set)

                # Calculate accuracy
                accuracy <- mean(predictions == test_set$direction)

                # Add to results
                results <- rbind(results, data.frame(model = model_name,
                  model_type = model_type, lag = lag, accuracy = accuracy,
                  stringsAsFactors = FALSE))
            }
        }
    }

    # Sort by accuracy in descending order
    results <- results[order(-results$accuracy), ]

    # Add rank column
    results$rank <- 1:nrow(results)
```

```
    results
}
```

3 Training Machine Learning Algorithms

We are now going to train and compare various machine learning algorithms with different sets of selected features and different numbers of lags.

Before starting with those algorithms, we will start with very simple ones to have a reference for comparison.

3.1 Simple algorithms

3.1.1 Random guess

We will run a montecarlo simulation of 1000 random guesses of direction and compare it with the test set.

```
random_guess_simulations <- replicate(1000, {
    estimated_direction <- replicate(nrow(test_set), sample(c("up",
        "down"), 1))
    mean(estimated_direction == test_set$direction)
})

mean_accuracy <- mean(random_guess_simulations)
standard_deviation <- sd(random_guess_simulations)
print(paste("Mean accuracy:", round(mean_accuracy, 4)))
```

```
## [1] "Mean accuracy: 0.5002"
```

```
print(paste("Standard deviation:", round(standard_deviation,
    4)))
```

```
## [1] "Standard deviation: 0.0109"
```

3.1.2 Always up

We can also compare this with an always up strategy:

```
always_up <- function(test_set) {
    replicate(nrow(test_set), "up")
}
always_up_accuracy <- mean(always_up(test_set) == test_set$direction)
print(paste("Always up accuracy:", round(always_up_accuracy,
    4)))
```

```
## [1] "Always up accuracy: 0.5111"
```

3.1.3 Previous direction

Now, let's see how this compares to simply predicting the same direction as the previous candle (lag 1):

```
previous_direction <- function(test_set) {
    test_set$direction_lag_1
}
previous_direction_accuracy <- mean(previous_direction(test_set) ==
    test_set$direction)
print(paste("Previous direction accuracy:", round(previous_direction_accuracy,
```

```
    4)))
```

```
## [1] "Previous direction accuracy: 0.4658"
```

3.1.4 Opposite direction to previous one

```
opposite_direction <- function(test_set) {
    ifelse(test_set$direction_lag_1 == "up", "down", "up")
}
opposite_direction_accuracy <- mean(opposite_direction(test_set) ==
    test_set$direction)
print(paste("Opposite direction to the previous one accuracy:",
    round(opposite_direction_accuracy, 4)))
```

```
## [1] "Opposite direction to the previous one accuracy: 0.5342"
```

We can see that giving the opposite direction to the previous one has the highest accuracy so far: `0.5342`. Let's see how it compares with machine learning algorithms.

3.2 Machine learning algorithms

For each of these machine learning algorithms, we will use either 1, 3, 5, 7, or 15 lagged data. We will later fine-tune this number of lagged candles to determine the optimal value.

3.2.1 OHLC features

We will first try to use the lagged OHLC features obtained directly from the Coinbase dataset:

- open
- high
- low
- close
- volume

Table 11: Model comparison for OHLC features

model	model_type	lag	accuracy	rank
glm_model_OHLC_lag_1	glm	1	0.5429363	1
glm_model_OHLC_lag_3	glm	3	0.5397045	2
glm_model_OHLC_lag_7	glm	7	0.5337027	3
glm_model_OHLC_lag_5	glm	5	0.5323176	4
glm_model_OHLC_lag_15	glm	15	0.5212373	5
rpart_model_OHLC_lag_1	rpart	1	0.5110803	6
rpart_model_OHLC_lag_7	rpart	7	0.5110803	7
rpart_model_OHLC_lag_3	rpart	3	0.5096953	8
gbm_model_OHLC_lag_15	gbm	15	0.5069252	9
knn_model_OHLC_lag_1	knn	1	0.5064635	10
gbm_model_OHLC_lag_3	gbm	3	0.5060018	11
rf_model_OHLC_lag_15	rf	15	0.5023084	12
gbm_model_OHLC_lag_1	gbm	1	0.5004617	13
rf_model_OHLC_lag_3	rf	3	0.4972299	14
rpart_model_OHLC_lag_15	rpart	15	0.4958449	15

model	model_type	lag	accuracy	rank
rf_model_OHLC_lag_1	rf	1	0.4944598	16
knn_model_OHLC_lag_7	knn	7	0.4930748	17
knn_model_OHLC_lag_15	knn	15	0.4926131	18
rpart_model_OHLC_lag_5	rpart	5	0.4912281	19
gbm_model_OHLC_lag_5	gbm	5	0.4898430	20
gbm_model_OHLC_lag_7	gbm	7	0.4884580	21
rf_model_OHLC_lag_5	rf	5	0.4833795	22
rf_model_OHLC_lag_7	rf	7	0.4764543	23
knn_model_OHLC_lag_5	knn	5	0.4755309	24
knn_model_OHLC_lag_3	knn	3	0.4699908	25

Table 12: Summary statistics for OHLC features

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.4983	0.0088	0.5069
glm	0.5340	0.0083	0.5429
knn	0.4875	0.0147	0.5065
rf	0.4908	0.0106	0.5023
rpart	0.5038	0.0095	0.5111

3.2.2 Candle features

Now let's try to use the lagged candle features:

- body_size
- upper_shadow_size
- lower_shadow_size
- direction
- close
- volume

Table 13: Model comparison for candles features

model	model_type	lag	accuracy	rank
gbm_model_candles_lag_1	gbm	1	0.5470914	1
rpart_model_candles_lag_1	rpart	1	0.5447830	2
glm_model_candles_lag_7	glm	7	0.5433980	3
glm_model_candles_lag_5	glm	5	0.5401662	4
glm_model_candles_lag_3	glm	3	0.5397045	5
rpart_model_candles_lag_3	rpart	3	0.5341644	6
rpart_model_candles_lag_7	rpart	7	0.5341644	7
gbm_model_candles_lag_7	gbm	7	0.5313943	8
glm_model_candles_lag_1	glm	1	0.5300092	9
rpart_model_candles_lag_5	rpart	5	0.5286242	10
knn_model_candles_lag_3	knn	3	0.5272392	11
gbm_model_candles_lag_15	gbm	15	0.5258541	12
glm_model_candles_lag_15	glm	15	0.5249307	13
rf_model_candles_lag_1	rf	1	0.5244691	14

model	model_type	lag	accuracy	rank
rf_model_candles_lag_7	rf	7	0.5235457	15
knn_model_candles_lag_1	knn	1	0.5235457	16
gbm_model_candles_lag_3	gbm	3	0.5221607	17
knn_model_candles_lag_7	knn	7	0.5203139	18
gbm_model_candles_lag_5	gbm	5	0.5184672	19
knn_model_candles_lag_15	knn	15	0.5147738	20
rpart_model_candles_lag_15	rpart	15	0.5110803	21
rf_model_candles_lag_3	rf	3	0.5101570	22
rf_model_candles_lag_5	rf	5	0.5083102	23
knn_model_candles_lag_5	knn	5	0.5050785	24
rf_model_candles_lag_15	rf	15	0.5027701	25

Table 14: Summary statistics for candles features

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5290	0.0112	0.5471
glm	0.5356	0.0078	0.5434
knn	0.5182	0.0086	0.5272
rf	0.5139	0.0097	0.5245
rpart	0.5306	0.0124	0.5448

3.2.3 Candles features and fear and greed index

Now let's try to use the lagged candles features and the fear and greed index:

- body_size
- upper_shadow_size
- lower_shadow_size
- direction
- close
- value
- volume

Table 15: Model comparison for candles features and fear and greed index

model	model_type	lag	accuracy	rank
rpart_model_candles_fg_lag_1	rpart	1	0.5447830	1
glm_model_candles_fg_lag_7	glm	7	0.5438596	2
glm_model_candles_fg_lag_5	glm	5	0.5429363	3
glm_model_candles_fg_lag_3	glm	3	0.5410896	4
rpart_model_candles_fg_lag_3	rpart	3	0.5341644	5
rpart_model_candles_fg_lag_5	rpart	5	0.5341644	6
rpart_model_candles_fg_lag_7	rpart	7	0.5341644	7
rpart_model_candles_fg_lag_15	rpart	15	0.5341644	8
gbm_model_candles_fg_lag_7	gbm	7	0.5332410	9
gbm_model_candles_fg_lag_3	gbm	3	0.5318560	10

model	model_type	lag	accuracy	rank
rf_model_candles_fg_lag_3	rf	3	0.5295476	11
gbm_model_candles_fg_lag_1	gbm	1	0.5290859	12
glm_model_candles_fg_lag_15	glm	15	0.5286242	13
glm_model_candles_fg_lag_1	glm	1	0.5277008	14
gbm_model_candles_fg_lag_5	gbm	5	0.5249307	15
rf_model_candles_fg_lag_15	rf	15	0.5198523	16
rf_model_candles_fg_lag_1	rf	1	0.5184672	17
knn_model_candles_fg_lag_1	knn	1	0.5175439	18
rf_model_candles_fg_lag_5	rf	5	0.5115420	19
knn_model_candles_fg_lag_7	knn	7	0.5115420	20
knn_model_candles_fg_lag_5	knn	5	0.5073869	21
knn_model_candles_fg_lag_15	knn	15	0.5041551	22
gbm_model_candles_fg_lag_15	gbm	15	0.5041551	23
rf_model_candles_fg_lag_7	rf	7	0.4976916	24
knn_model_candles_fg_lag_3	knn	3	0.4967682	25

Table 16: Summary statistics for candles features and fear and greed index

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5247	0.0119	0.5332
glm	0.5368	0.0080	0.5439
knn	0.5075	0.0078	0.5175
rf	0.5154	0.0118	0.5295
rpart	0.5363	0.0047	0.5448

3.2.4 Candles features, fear and greed index and chain data

We will try to use the lagged candles features, the fear and greed index and the chain data:

- body_size
- upper_shadow_size
- lower_shadow_size
- direction
- close
- value
- hash_rate
- avg_block_size
- n_transactions
- utxo_count
- volume

Table 17: Model comparison for candles features, fear and greed index and chain data

model	model_type	lag	accuracy	rank
rpart_model_candles_fg_chain_lag_15	rpart	15	0.5447830	1
rpart_model_candles_fg_chain_lag_1	rpart	1	0.5341644	2
rpart_model_candles_fg_chain_lag_3	rpart	3	0.5341644	3
rpart_model_candles_fg_chain_lag_5	rpart	5	0.5341644	4
gbm_model_candles_fg_chain_lag_1	gbm	1	0.5290859	5
glm_model_candles_fg_chain_lag_3	glm	3	0.5258541	6
glm_model_candles_fg_chain_lag_7	glm	7	0.5235457	7
gbm_model_candles_fg_chain_lag_15	gbm	15	0.5221607	8
rf_model_candles_fg_chain_lag_7	rf	7	0.5212373	9
glm_model_candles_fg_chain_lag_15	glm	15	0.5203139	10
glm_model_candles_fg_chain_lag_5	glm	5	0.5198523	11
rf_model_candles_fg_chain_lag_3	rf	3	0.5184672	12
gbm_model_candles_fg_chain_lag_5	gbm	5	0.5184672	13
rpart_model_candles_fg_chain_lag_7	rpart	7	0.5110803	14
gbm_model_candles_fg_chain_lag_3	gbm	3	0.5101570	15
knn_model_candles_fg_chain_lag_15	knn	15	0.5092336	16
glm_model_candles_fg_chain_lag_1	glm	1	0.5069252	17
knn_model_candles_fg_chain_lag_5	knn	5	0.5060018	18
knn_model_candles_fg_chain_lag_7	knn	7	0.5060018	19
rf_model_candles_fg_chain_lag_1	rf	1	0.5032318	20
knn_model_candles_fg_chain_lag_3	knn	3	0.5023084	21
gbm_model_candles_fg_chain_lag_7	gbm	7	0.5023084	22
rf_model_candles_fg_chain_lag_5	rf	5	0.5000000	23
knn_model_candles_fg_chain_lag_1	knn	1	0.4990766	24
rf_model_candles_fg_chain_lag_15	rf	15	0.4981533	25

Table 18: Summary statistics for candles features, fear and greed index and chain data

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5164	0.0104	0.5291
glm	0.5193	0.0073	0.5259
knn	0.5045	0.0039	0.5092
rf	0.5082	0.0108	0.5212
rpart	0.5317	0.0124	0.5448

3.2.5 Candles features, fear and greed index, chain data and technical analysis indicators

Finally let's add the technical analysis indicators to the model, so we will use the following lagged features:

- body_size
- upper_shadow_size
- lower_shadow_size
- direction
- close
- value

- hash_rate
- avg_block_size
- n_transactions
- utxo_count
- volume
- roc
- macd
- signal
- rsi
- up_bband
- mavg
- dn_bband
- pctB

Table 19: Model comparison for candles features, fear and greed index, chain data and technical analysis indicators

model	model_type	lag	accuracy	rank
gbm_model_candles_fg_chain_ta_lag_1	gbm	1	0.5498615	1
gbm_model_candles_fg_chain_ta_lag_7	gbm	7	0.5424746	2
rpart_model_candles_fg_chain_ta_lag_1	rpart	1	0.5383195	3
glm_model_candles_fg_chain_ta_lag_3	glm	3	0.5364728	4
glm_model_candles_fg_chain_ta_lag_5	glm	5	0.5327793	5
gbm_model_candles_fg_chain_ta_lag_3	gbm	3	0.5304709	6
glm_model_candles_fg_chain_ta_lag_15	glm	15	0.5295476	7
gbm_model_candles_fg_chain_ta_lag_5	gbm	5	0.5295476	8
rpart_model_candles_fg_chain_ta_lag_3	rpart	3	0.5272392	9
rpart_model_candles_fg_chain_ta_lag_5	rpart	5	0.5272392	10
rpart_model_candles_fg_chain_ta_lag_7	rpart	7	0.5272392	11
rpart_model_candles_fg_chain_ta_lag_15	rpart	15	0.5272392	12
glm_model_candles_fg_chain_ta_lag_7	glm	7	0.5253924	13
rf_model_candles_fg_chain_ta_lag_7	rf	7	0.5249307	14
glm_model_candles_fg_chain_ta_lag_1	glm	1	0.5235457	15
gbm_model_candles_fg_chain_ta_lag_15	gbm	15	0.5203139	16
rf_model_candles_fg_chain_ta_lag_3	rf	3	0.5193906	17
rf_model_candles_fg_chain_ta_lag_15	rf	15	0.5143121	18
knn_model_candles_fg_chain_ta_lag_5	knn	5	0.5101570	19
knn_model_candles_fg_chain_ta_lag_3	knn	3	0.5078486	20
rf_model_candles_fg_chain_ta_lag_1	rf	1	0.5041551	21
knn_model_candles_fg_chain_ta_lag_1	knn	1	0.5036934	22
rf_model_candles_fg_chain_ta_lag_5	rf	5	0.5018467	23
knn_model_candles_fg_chain_ta_lag_7	knn	7	0.4921514	24
knn_model_candles_fg_chain_ta_lag_15	knn	15	0.4847645	25

Table 20: Summary statistics for candles features, fear and greed index, chain data and technical analysis indicators

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5345	0.0116	0.5499
glm	0.5295	0.0053	0.5365
knn	0.4997	0.0109	0.5102
rf	0.5129	0.0098	0.5249
rpart	0.5295	0.0050	0.5383

3.3 Models comparison

Table 21: Top models across all feature sets

model	model_type	lag	accuracy	rank
gbm_model_candles_fg_chain_ta_lag_1	gbm	1	0.5498615	1
gbm_model_candles_lag_1	gbm	1	0.5470914	1
rpart_model_candles_lag_1	rpart	1	0.5447830	2
rpart_model_candles_fg_lag_1	rpart	1	0.5447830	1
rpart_model_candles_fg_chain_lag_15	rpart	15	0.5447830	1
glm_model_candles_fg_lag_7	glm	7	0.5438596	2
glm_model_candles_lag_7	glm	7	0.5433980	3
glm_model_OHLC_lag_1	glm	1	0.5429363	1
glm_model_candles_fg_lag_5	glm	5	0.5429363	3
gbm_model_candles_fg_chain_ta_lag_7	gbm	7	0.5424746	2

Table 22: Summary of feature sets

feature_set	avg_accuracy	sd_accuracy
candles	0.5255402	0.0122263
candles_fg	0.5244137	0.0143247
candles_fg_chain_ta	0.5214589	0.0153129
candles_fg_chain	0.5161034	0.0127251
OHLC	0.5030102	0.0194071

As we can see, the models using `Gradient boosting` generally perform best, and the `candles` feature set appears to yield good results overall.

Also, as expected, the OHLC feature set did not perform well, likely because it uses raw price data which can be challenging for machine learning algorithms to utilize effectively without further transformation.

But in order to proceed to the fine tuning, we will use `gbm_model_candles_fg_chain_ta_lag_1` since it is outperforming the other models with an accuracy of `0.5498615`.

4 Fine tuning

Before proceeding to the fine tuning, it's worth checking if the GBM model performs better with the same features and 2 lags instead of 1.

```
## [1] "Model loaded from cache: models/gbm_42d3a7ed8ef45c8b47ea08a3f199fd00.rds"
```

```
## [1] 0.5369344
```

The results confirm that the GBM model using the `candles_fg_chain_ta` feature set with 1 lag (`gbm_model_candles_fg_chain_ta_lag_1`) still performs better.

Let's use its tuning values and fine-tune it.

Table 23: Best tuning values for the GBM model

n.trees	interaction.depth	shrinkage	n.minobsinnode
50	1	0.1	10

We will use a grid of values around these initial best parameters to fine-tune the algorithm. Also, unlike the previous training runs, we will use cross-validation for this step to promote a more robust prediction algorithm that should perform more reliably on unseen data compared to a model tuned only on the specific `test_set`.

```r
# Define the tuning grid with the best values
gbm_grid <- expand.grid(n.trees = c(45, 46, 47, 48, 49, 50, 51,
    52, 53, 54, 55), interaction.depth = c(1, 2), shrinkage = c(0.05,
    0.1, 0.15), n.minobsinnode = c(8, 9, 10, 11, 12))

# Set up cross-validation
train_control <- trainControl(method = "cv", number = 5, verboseIter = TRUE,
    classProbs = TRUE, summaryFunction = twoClassSummary)

# Train the fine-tuned model with cross-validation
formula_candles_fg_chain_ta_lag_1 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value", "hash_rate", "avg_block_size", "n_transactions",
    "utxo_count", "roc", "macd", "signal", "rsi", "up_bband",
    "mavg", "dn_bband", "pctB", "volume"), 1)


if (!file.exists("models/gbm_model_candles_fg_chain_ta_lag_1_tuned.rds")) {
    gbm_model_candles_fg_chain_ta_lag_1_tuned <- train(formula_candles_fg_chain_ta_lag_1,
        data = train_set, method = "gbm", trControl = train_control,
        tuneGrid = gbm_grid, metric = "ROC")
    saveRDS(gbm_model_candles_fg_chain_ta_lag_1_tuned,
     ↪  "models/gbm_model_candles_fg_chain_ta_lag_1_tuned.rds")
} else {
    gbm_model_candles_fg_chain_ta_lag_1_tuned <-
↪   readRDS("models/gbm_model_candles_fg_chain_ta_lag_1_tuned.rds")
}

# Evaluate the fine-tuned model on the test set
accuracy_gbm_model_candles_fg_chain_ta_lag_1_tuned <-
↪   mean(predict(gbm_model_candles_fg_chain_ta_lag_1_tuned,
    test_set) == test_set$direction)
print(paste("Fine-tuned model accuracy:",
↪   accuracy_gbm_model_candles_fg_chain_ta_lag_1_tuned))
```

```
## [1] "Fine-tuned model accuracy: 0.545706371191136"
```

As the result shows, the fine-tuned model performs slightly worse on the test set than the original non-tuned model, but its accuracy is still higher than the third-best model identified earlier.

We can see below the values of the different parameters:

Table 24: Best tuning values for the GBM model after fine tuning

n.trees	interaction.depth	shrinkage	n.minobsinnode
46	2	0.05	11

Now let's compare the results and analyze what we have found.

5 Results

We will compare the best model for each feature set.

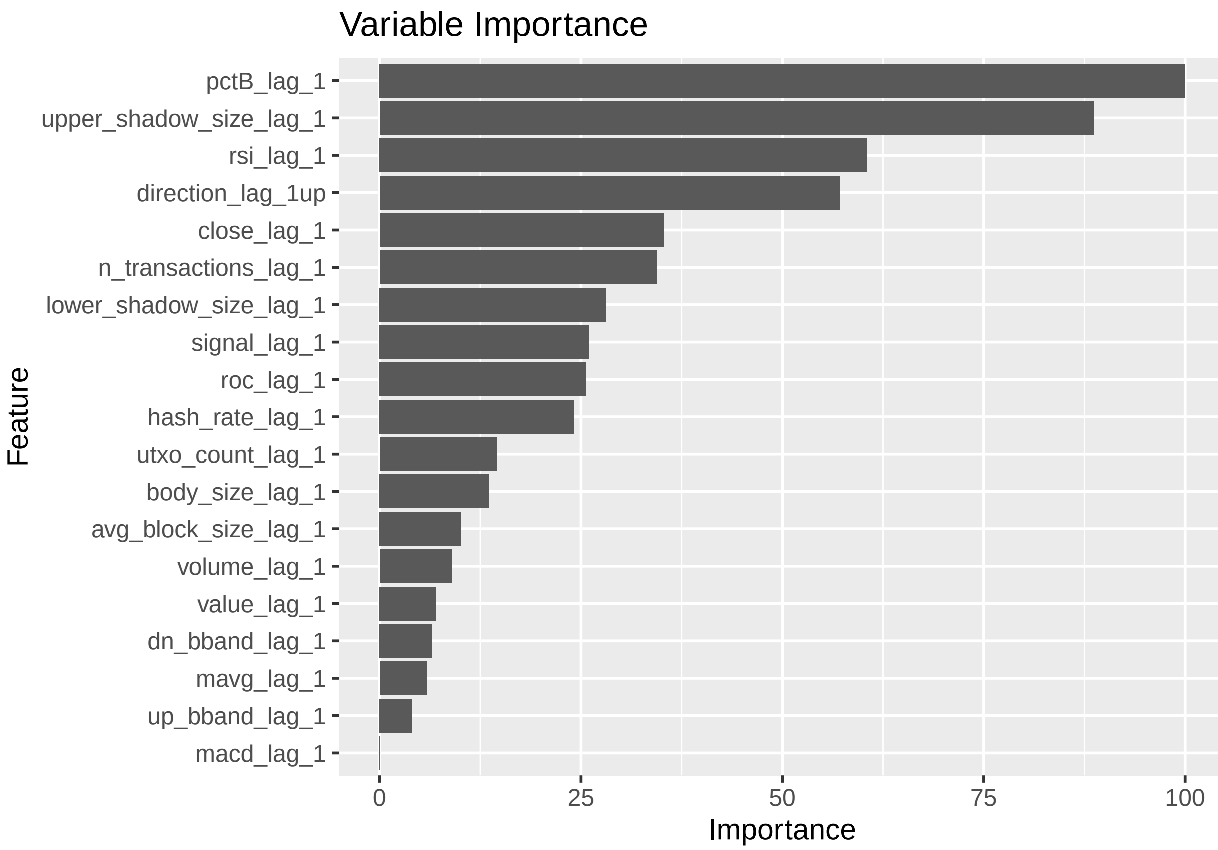
Table 25: Comparison of best models from each feature set and baseline methods

	Model	Features	Model Type	Lag	Accuracy
211	gbm_model_candles_fg_chain_ta_lag_1	Candles, F&G, Chain, TA	gbm	1	0.5499
21	gbm_model_candles_lag_1	Candles	gbm	1	0.5471
12	gbm_model_candles_fg_chain_ta_lag_1_tuned	Candles, F&G, Chain, TA	gbm	1	0.5457
11	rpart_model_candles_fg_lag_1	Candles, F&G	rpart	1	0.5448
15	rpart_model_candles_fg_chain_lag_15	Candles, F&G, Chain	rpart	15	0.5448
5	glm_model_OHLC_lag_1	OHLC	glm	1	0.5429
4	opposite_direction	simple	simple	1	0.5342
2	always_up	simple	simple	NA	0.5111
1	random_guess	simple	simple	NA	0.5002
3	previous_direction	simple	simple	1	0.4658

While the non-tuned model achieved the highest accuracy (`0.5499`) on the test set, we consider the fine-tuned model (accuracy `0.5457`) potentially more robust due to the use of cross-validation during tuning.

5.1 Most relevant features

Now, let's examine the most important features identified by the fine-tuned algorithm.



Interestingly, the features with an importance score higher than 25 include indicators from all three categories: Technical Analysis (TA), Chain Data, and Candle Data.

5.2 Confusion matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  up down
##       up   816  693
##       down 291  366
##
##                Accuracy : 0.5457
##                  95% CI : (0.5245, 0.5668)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 0.0006755
##
##                   Kappa : 0.0834
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7371
##             Specificity : 0.3456
##          Pos Pred Value : 0.5408
##          Neg Pred Value : 0.5571
##              Prevalence : 0.5111
##          Detection Rate : 0.3767
##    Detection Prevalence : 0.6967
```

```
##        Balanced Accuracy : 0.5414
##
##         'Positive' Class : up
##
```

We observe that the model is slightly better at predicting `down` candles than `up` candles, at least on this particular `test_set`. This suggests that in future trading it would potentially get a better success rate at shorting rather than longing.

6 Conclusion

This study demonstrates that predictions using a trained model can outperform random chance.

With an accuracy of `0.5457`, it would be important for a trader to use the predictions along with well-defined targets for take-profit and stop-loss, where the potential profit outweighs the potential loss.

Let's discuss the limitations of this study and what could be done next.

6.1 Limitations

As mentioned in the report, most of the training has been done without cross-validation in order to save computation time; therefore, some other algorithms or configurations might have performed better than the current one.

6.2 Potential improvements

A deeper review of existing research could provide a stronger foundation for improving this algorithm. Furthermore, other algorithms might perform better than GBM and warrant investigation.

Also, some other Technical Analysis indicators could be used to potentially achieve better predictions, perhaps by coupling our hourly candles with data from smaller time frames.

Last but not least, using different algorithms depending on the market conditions could also be a good solution. By comparing how algorithms perform in different market types (bearish, bullish, or sideways) and switching to the appropriate model depending on the type of market could also improve the accuracy.

6.3 Trading application

In order to be closer to the trading reality and test the ability of the model to make profit, we would recommend starting with backtesting to see how it performs when setting appropriate stop-loss / take-profit targets. Then, after tuning the trading algorithm, it would be worthwhile to conduct paper trading before deploying the model for actual real-money trading.

References

[1] T. Dong, "New spot bitcoin ETFs to buy." Accessed: Jan. 03, 2025. [Online]. Available: https://money.usnews.com/investing/articles/new-spot-bitcoin-etfs-to-buy

[2] Reuters, "Trump signs order to establish strategic bitcoin reserve, white house crypto czar." Accessed: Mar. 07, 2025. [Online]. Available: https://www.reuters.com/technology/trump-signs-order-establish-strategic-bitcoin-reserve-white-house-crypto-czar-2025-03-07/

[3] Reuters, "El salvador announces more bitcoin purchases, gives IMF assurances." Accessed: Mar. 05, 2025. [Online]. Available: https://www.reuters.com/technology/el-salvador-announces-more-bitcoin-purchases-gives-imf-assurances-2025-03-05/

[4] S. Adittane, "How cryptocurrencies work (technical guide)." Medium, May 2018. Accessed: May 08, 2018. [Online]. Available: https://medium.com/learning-lab/how-cryptocurrencies-work-technical-guide-95950c002b8f

[5] TechQualityPedia, "Candlestick patterns: bullish." Accessed: Sep. 07, 2024. [Online]. Available: https://techqualitypedia.com/candlestick-patterns-bullish/

[6] A. Hill, "Candlestick patterns explained." Accessed: Jun. 04, 2021. [Online]. Available: https://www.tradingsim.com/blog/candlestick-patterns-explained

[7] Coinbase, "Get product candles - coinbase exchange REST API reference." [Online]. Available: https://docs.cdp.coinbase.com/exchange/reference/exchangerestapi_getproductcandles

[8] Blockchain.com, "Blockchain explorer - bitcoin charts." [Online]. Available: https://www.blockchain.com/explorer/charts/total-bitcoins

[9] Alternative.me, "Crypto fear & greed index." [Online]. Available: https://alternative.me/crypto/fear-and-greed-index/

[10] S. Adittane, "Become a better crypto trader with technical and chart analysis." Medium, 2018. Accessed: Jan. 20, 2018. [Online]. Available: https://medium.com/learning-lab/become-a-better-crypto-trader-with-technical-and-chart-analysis-1496b2fc6b85

[11] "An optimized machine learning model for candlestick chart analysis to predict stock market trends," *NeuroQuantology*, 2022, Available: https://www.neuroquantology.com/open-access/An+Optimized+Machine+Learning+Model+for+Candlestick+Chart+Analysis+to+Predict+Stock+Market+Trends_9861/

[12] M. Nakano, A. Takahashi, and S. Takahashi, "Stock price prediction by deep learning with text and price information," *arXiv*, 2016, Accessed: Jun. 06, 2016. [Online]. Available: https://arxiv.org/pdf/1606.00930