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

Training various machine learning algorithms to predict the next candlestick of the bitcoin price using various lagged features

Sandoche Adittane

2025-04-18

Abstract

This report explores how to get the best accuracy on 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 and different number of lagged features. This project is part of the 'Data Science: Capstone' module of HarvardX PH125.9x from 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, candlesticks and what is the goal of this study.

1.1 Introduction to Bitcoin

This last years Bitcoin (BTC) has been gaining attention not only by retail investors but also by institutional investor. In 2025 we've seen the emergence of spot Bitcoin exchange-traded funds (ETF) from institutions such as BlackRock, VanEck, Grayscale. With a market capitalization of about 1.68 billion in dollars at the time of writing, Bitcoin started as a peer-to-peer currency, a free alernative to centralized currencies controlled by central banks. It is now used more as an investment, a store of value and even considered as a strategic reserve assets by some countries.

TODO: Add examples with sources.

Bitcoin ows is decentralization and to it's data structure, the blockchain, a chain of block that contains transaction, and to its consensus, the proof of work. Without going too much into details, it makes a Bitcoin a currency that does not rely on a centralized server. Proof of work is a cryptographic competition where the Bitcoin servers called nodes compete to decide which one is the next block to be 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 who succeed at "mining" successfully gets rewarded for that.

TODO: reference to my article

The fact that Bitcoin is defined by its codebase is quite facinating, also having all its ledger visible and publically available gives a lot of data available to analyze. Moreover unlike stocks BTC can be traded any time, there is 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 asset.

1.2 What are candlesticks?

Let's talk about the candlestick. The price of assets such as bitcoin is described by a serie of candle stick defined by, an opening price, a close price a high and a low also called OHLC. A candlestick can be "up" / "bullish" if closing price is higher than opening price, or "down" / "bearish" otherwise. You can see this visually with the following figure. " "

https://i0.wp.com/techqualitypedia.com/wp-content/uploads/2024/09/candlestick-components.jpg?w=1491&ssl=1 Source: https://techqualitypedia.com/candlestick-patterns-bullish/

The candle stick chart is defined as a time serie of candles, each candle is defined at a defined time and have a time duration. We will explain more in detail in the exploratory analysis.

1.3 Candlesticks pattern

TODO Talk about chartists and common patterns

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 be also part of the research. We will have to not only find N but also find what are the best features to achieve the best accuracy.

1.5 Applications

Why is the direction of a candlestick matter? Because being able to predict the direction of the next candle could enable trader to buy and sell on spot market when the predicted candle is green. Also perpetual futures trader can go both way, they can long when the prediction says "up" and "short" when the predictions says "down".

TODO: Give some resource to learn about spot vs future.

2 Exploratory data analysis

In this section we will see what are the are the different dataset available, see what features are available to train the different models, prepare the data, verify it, and choose different machine learning algorithms we will use and compare.

2.1 Data sets

In order to conduct this study we used as a the main data set the historic rates for the trading pair BTC-USD using Coinbase API. TODO: add reference https://docs.cdp.coinbase.com/exchange/reference/exchangerestapi_getproductcandles

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 is the entire year 2024 and the start of the year 2025 until the day we started the study. 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.

I choose this settings to have a dataset of around 10000 candles but also since Bitcoin ETF has been approved the market may have taken a different dynamic than the 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 candle stick dataset. As described in the overview it contains the OCLH data, timestamp and the volume of each candles.

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, I tried to find other dataset that could represent each of the type of users that could eventually help in our predictions and I 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 contains bitcoin, and reflects the network activity (users)

https://www.blockchain.com/explorer/charts/total-bitcoins https://alternative.me/crypto/fear-and-greed-index/

```
fear_and_greed_index <- read_csv(paste0("data/", trading_pair,
    "_fear_and_greed_index_", start_date, "_", end_date, ".csv"))
```

```
## Rows: 453 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): value_classification
## dbl  (1): value
## dttm (1): timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fear_and_greed_index <- fear_and_greed_index %>%
    mutate(value = as.numeric(value))
knitr::kable(head(fear_and_greed_index), format = "simple", caption = "Overview of the BTC fear and gre
```

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 serie 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 data set contains 453 entries.

```r
hash_rate <- jsonlite::fromJSON("data/hash-rate.json")$`hash-rate` %>%
    rename(timestamp = x, hash_rate = y) %>%
    mutate(timestamp = as.POSIXct(timestamp/1000, origin = "1970-01-01",
        tz = "UTC")) %>%
    filter(timestamp >= as.POSIXct(start_date, origin = "1970-01-01",
        tz = "UTC") & timestamp <= as.POSIXct(end_date, origin = "1970-01-01",
        tz = "UTC"))
knitr::kable(head(hash_rate), format = "simple", caption = "Overview of the BTC hash rate dataset")
```

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 serie of the daily hash rate, it is a value in TH/s. The data set contains 454 entries.

```r
average_block_size <- jsonlite::fromJSON("data/avg-block-size.json")$`avg-block-size` %>%
    rename(timestamp = x, avg_block_size = y) %>%
    mutate(timestamp = as.POSIXct(timestamp/1000, origin = "1970-01-01",
        tz = "UTC")) %>%
    filter(timestamp >= as.POSIXct(start_date, origin = "1970-01-01",
        tz = "UTC") & timestamp <= as.POSIXct(end_date, origin = "1970-01-01",
        tz = "UTC"))
knitr::kable(head(average_block_size), format = "simple", caption = "Overview of the BTC average block
```

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 serie of the daily average block size, it is a value in bytes. The data set contains 454 entries.

```r
n_transactions <- jsonlite::fromJSON("data/n-transactions.json")$`n-transactions` %>%
    rename(timestamp = x, n_transactions = y) %>%
    mutate(timestamp = as.POSIXct(timestamp/1000, origin = "1970-01-01",
        tz = "UTC")) %>%
    filter(timestamp >= as.POSIXct(start_date, origin = "1970-01-01",
        tz = "UTC") & timestamp <= as.POSIXct(end_date, origin = "1970-01-01",
        tz = "UTC"))
knitr::kable(head(n_transactions), format = "simple", caption = "Overview of the BTC number of transacti
```

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 serie of the daily number of transactions, it is a value in transactions. The data set contains 454 entries.

```r
utxo_count <- jsonlite::fromJSON("data/utxo-count.json")$`utxo-count` %>%
    rename(timestamp = x, utxo_count = y) %>%
    mutate(timestamp = as.POSIXct(timestamp/1000, origin = "1970-01-01",
        tz = "UTC"), timestamp = as.Date(timestamp)) %>%
    filter(timestamp >= as.Date(start_date) & timestamp <= as.Date(end_date)) %>%
    group_by(timestamp) %>%
    summarise(utxo_count = mean(utxo_count))  # Take average for each date
knitr::kable(head(utxo_count), format = "simple", caption = "Overview of the BTC UTXO count dataset")
```

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 serie of the daily number of UTXO, it is the count of UTXO in the network. The data set contains 454 entries. We had to group by timestamp since some dates had more than 1 value.

As we can see `fear_and_greed_index` seems to miss one data point. We will see fix that in the next sections.

We have different dataset that we will use for the predictions but we are still missing one important data used by traders, technical analysis indicator.

Based on a previous research and blog post I wrote, I 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)

TODO: add https://medium.com/learning-lab/become-a-better-crypto-trader-with-technical-and-chart-analysis-1496b2fc6b85

Before preparing the dataset let's see what would be the 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 used itself for a machine learning algorithm it will be hard to use those raw absolute values since the price always fluctuate, so 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 / wicks
- Size of the lower shadow / wicks
- Direction of the candle (up or down)
- Closing price

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

2.3 Preparation

```
enhance_dataset <- function(candles_data, fear_and_greed_index_data, hash_rate_data, average_block_size_
  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,
```

```r
      n = 14,
      col_rename = "rsi"
    ) %>%
    tq_mutate(
      select = close,
      mutate_fun = BBands,
      n = 20,
      sd = 2,
      col_rename = "bband"
    )

  candles_enhanced
}

candles_enhanced <- enhance_dataset(candles, fear_and_greed_index, hash_rate, average_block_size, n_tran
```

```
## Warning in coerce_to_tibble(ret, date_col_name, time_zone, col_rename): Could not rename columns. The
##   Is the length of ‘col_rename‘ the same as the number of columns returned from the ‘mutate_fun‘?
```

```r
knitr::kable(head(candles_enhanced), format = "simple", caption = "Overview of the candlestick dataset e
```

time	low	high	open	close	volume	date_only	value	value_classification	h
2024-01-01 00:00:00	42261.58	42543.64	42288.58	42452.66	379.1973	2024-01-01	65	Greed	5
2024-01-01 01:00:00	42415.00	42749.99	42453.83	42594.68	396.2019	2024-01-01	65	Greed	5
2024-01-01 02:00:00	42488.03	42625.68	42594.58	42571.32	227.1412	2024-01-01	65	Greed	5
2024-01-01 03:00:00	42235.00	42581.26	42571.32	42325.11	306.0057	2024-01-01	65	Greed	5
2024-01-01 04:00:00	42200.00	42393.48	42325.10	42389.77	296.2336	2024-01-01	65	Greed	5
2024-01-01 05:00:00	42175.65	42396.09	42389.78	42231.47	188.1280	2024-01-01	65	Greed	5

We have now a table with all the features discussed above. Before adding the lagged features let's explore our set.

First let's see if we have NAs.

```r
na_indexes <- which(apply(candles_enhanced, 1, function(x) any(is.na(x))))
knitr::kable(candles_enhanced[na_indexes, ], format = "simple",
    caption = "NAs of the dataset")
```

time	low	high	open	close	volume	date_only	value	value_classification
2024-01-01 00:00:00	42261.58	42543.64	42288.58	42452.66	379.197253	2024-01-01	65	Greed
2024-01-01 01:00:00	42415.00	42749.99	42453.83	42594.68	396.201924	2024-01-01	65	Greed
2024-01-01 02:00:00	42488.03	42625.68	42594.58	42571.32	227.141166	2024-01-01	65	Greed
2024-01-01 03:00:00	42235.00	42581.26	42571.32	42325.11	306.005694	2024-01-01	65	Greed
2024-01-01 04:00:00	42200.00	42393.48	42325.10	42389.77	296.233644	2024-01-01	65	Greed
2024-01-01 05:00:00	42175.65	42396.09	42389.78	42231.47	188.128033	2024-01-01	65	Greed
2024-01-01 06:00:00	42199.63	42463.83	42231.47	42400.90	327.010976	2024-01-01	65	Greed

time	low	high	open	close	volume	date_only	value	value_classification
2024-01-01 07:00:00	42396.80	42534.49	42400.90	42496.49	407.835097	2024-01-01	65	Greed
2024-01-01 08:00:00	42451.00	42560.26	42496.58	42552.70	134.066714	2024-01-01	65	Greed
2024-01-01 09:00:00	42533.77	42692.84	42552.70	42650.97	128.157349	2024-01-01	65	Greed
2024-01-01 10:00:00	42625.75	42750.00	42650.97	42688.50	118.457396	2024-01-01	65	Greed
2024-01-01 11:00:00	42598.94	42767.60	42686.73	42690.00	135.177672	2024-01-01	65	Greed
2024-01-01 12:00:00	42610.77	42778.74	42690.00	42647.83	143.378020	2024-01-01	65	Greed
2024-01-01 13:00:00	42608.98	42750.00	42647.85	42715.88	101.798625	2024-01-01	65	Greed
2024-01-01 14:00:00	42581.54	42723.28	42717.53	42635.19	254.331002	2024-01-01	65	Greed
2024-01-01 15:00:00	42601.88	42868.74	42633.57	42797.33	323.614895	2024-01-01	65	Greed
2024-01-01 16:00:00	42680.01	42880.97	42799.37	42742.35	330.024110	2024-01-01	65	Greed
2024-01-01 17:00:00	42720.76	42846.42	42738.88	42833.66	254.872245	2024-01-01	65	Greed
2024-01-01 18:00:00	42835.63	43228.37	42835.63	43120.92	625.461696	2024-01-01	65	Greed
2024-01-01 19:00:00	43106.97	43567.46	43123.82	43547.61	506.451809	2024-01-01	65	Greed
2024-01-01 20:00:00	43537.04	43849.90	43537.04	43701.58	559.329005	2024-01-01	65	Greed
2024-01-01 21:00:00	43467.97	43800.00	43703.06	43632.21	313.991915	2024-01-01	65	Greed
2024-01-01 22:00:00	43389.00	43677.06	43631.79	43546.06	247.539449	2024-01-01	65	Greed
2024-01-01 23:00:00	43545.99	44240.80	43545.99	44220.78	1273.322823	2024-01-01	65	Greed
2024-01-02 00:00:00	44195.13	45250.00	44220.78	45093.17	3023.793096	2024-01-02	71	Greed
2024-01-02 01:00:00	44714.89	45417.45	45093.14	44894.58	1983.082789	2024-01-02	71	Greed
2024-01-02 02:00:00	44891.40	45500.00	44891.41	45485.31	1913.577949	2024-01-02	71	Greed
2024-01-02 03:00:00	45178.34	45601.00	45485.32	45473.97	1512.833935	2024-01-02	71	Greed
2024-01-02 04:00:00	45204.47	45544.10	45476.27	45218.83	681.159462	2024-01-02	71	Greed
2024-01-02 05:00:00	45131.00	45370.54	45218.84	45216.32	484.732085	2024-01-02	71	Greed
2024-01-02 06:00:00	45166.39	45361.61	45214.41	45208.54	495.637602	2024-01-02	71	Greed
2024-01-02 07:00:00	45209.48	45707.69	45209.48	45504.64	976.921273	2024-01-02	71	Greed
2024-01-02 08:00:00	45382.34	45899.96	45504.40	45808.07	759.882169	2024-01-02	71	Greed
2024-10-26 00:00:00	66413.18	66754.02	66564.51	66635.55	359.487900	2024-10-26	NA	NA
2024-10-26 01:00:00	66430.80	66711.88	66637.60	66597.10	226.587448	2024-10-26	NA	NA
2024-10-26 02:00:00	66331.95	66930.14	66594.88	66728.09	162.061446	2024-10-26	NA	NA
2024-10-26 03:00:00	66580.85	66890.00	66730.12	66816.54	122.871792	2024-10-26	NA	NA
2024-10-26 04:00:00	66687.79	66903.91	66814.44	66855.95	148.712344	2024-10-26	NA	NA
2024-10-26 05:00:00	66851.79	67156.74	66855.94	67049.34	163.124225	2024-10-26	NA	NA
2024-10-26 06:00:00	66959.24	67159.97	67049.33	67086.89	108.339046	2024-10-26	NA	NA
2024-10-26 07:00:00	66913.98	67108.03	67086.89	66926.56	105.386323	2024-10-26	NA	NA
2024-10-26 08:00:00	66920.30	67098.13	66926.56	67058.44	94.345883	2024-10-26	NA	NA
2024-10-26 09:00:00	66973.03	67188.55	67058.44	66973.03	92.454048	2024-10-26	NA	NA
2024-10-26 10:00:00	66920.07	67108.34	66968.89	66977.74	69.774037	2024-10-26	NA	NA
2024-10-26 11:00:00	66876.82	67083.85	66977.74	67055.68	93.974564	2024-10-26	NA	NA
2024-10-26 12:00:00	66906.75	67101.59	67056.84	66946.40	99.399923	2024-10-26	NA	NA
2024-10-26 13:00:00	66784.25	67031.28	66946.40	66808.06	92.616172	2024-10-26	NA	NA
2024-10-26 14:00:00	66644.83	66874.66	66803.39	66713.12	126.183413	2024-10-26	NA	NA
2024-10-26 15:00:00	66675.24	66920.88	66712.82	66795.54	87.307429	2024-10-26	NA	NA
2024-10-26 16:00:00	66781.74	66870.57	66800.48	66818.88	2.195708	2024-10-26	NA	NA
2024-10-26 17:00:00	66388.20	67055.10	66864.73	66974.50	49.094828	2024-10-26	NA	NA
2024-10-26 18:00:00	66926.63	67069.99	66974.49	66942.16	99.453480	2024-10-26	NA	NA
2024-10-26 19:00:00	66936.07	67103.18	66942.16	67100.49	223.657084	2024-10-26	NA	NA
2024-10-26 20:00:00	67050.49	67365.18	67100.50	67173.56	144.763009	2024-10-26	NA	NA
2024-10-26 21:00:00	66999.83	67186.23	67173.56	67089.21	121.258639	2024-10-26	NA	NA
2024-10-26 22:00:00	67015.71	67163.50	67088.99	67042.50	55.228574	2024-10-26	NA	NA
2024-10-26 23:00:00	66993.44	67069.68	67039.92	67012.56	100.942655	2024-10-26	NA	NA

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

1. Technical analisis indicators
2. Fear and greed index

The technical analysis indicators are located at the start of the table which is normal since to calculate these indicators you need to have a certain number of previous values. Clearing those NAs will be enough.

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

It's important to do so to have coherant 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)
```

```
## Warning in coerce_to_tibble(ret, date_col_name, time_zone, col_rename): Could not rename columns. The
##   Is the length of `col_rename` the same as the number of columns returned from the `mutate_fun`?
```

```
na_indexes <- which(apply(candles_enhanced_cleaned, 1, function(x) any(is.na(x))))
knitr::kable(candles_enhanced_cleaned[na_indexes, ], format = "simple",
    caption = "NAs of the dataset cleaned")
```

time	low	high	open	close	volume	date_only	value	value_classification
2024-01-01 00:00:00	42261.58	42543.64	42288.58	42452.66	379.1973	2024-01-01	65	Greed
2024-01-01 01:00:00	42415.00	42749.99	42453.83	42594.68	396.2019	2024-01-01	65	Greed
2024-01-01 02:00:00	42488.03	42625.68	42594.58	42571.32	227.1412	2024-01-01	65	Greed
2024-01-01 03:00:00	42235.00	42581.26	42571.32	42325.11	306.0057	2024-01-01	65	Greed
2024-01-01 04:00:00	42200.00	42393.48	42325.10	42389.77	296.2336	2024-01-01	65	Greed
2024-01-01 05:00:00	42175.65	42396.09	42389.78	42231.47	188.1280	2024-01-01	65	Greed
2024-01-01 06:00:00	42199.63	42463.83	42231.47	42400.90	327.0110	2024-01-01	65	Greed
2024-01-01 07:00:00	42396.80	42534.49	42400.90	42496.49	407.8351	2024-01-01	65	Greed
2024-01-01 08:00:00	42451.00	42560.26	42496.58	42552.70	134.0667	2024-01-01	65	Greed
2024-01-01 09:00:00	42533.77	42692.84	42552.70	42650.97	128.1573	2024-01-01	65	Greed
2024-01-01 10:00:00	42625.75	42750.00	42650.97	42688.50	118.4574	2024-01-01	65	Greed
2024-01-01 11:00:00	42598.94	42767.60	42686.73	42690.00	135.1777	2024-01-01	65	Greed
2024-01-01 12:00:00	42610.77	42778.74	42690.00	42647.83	143.3780	2024-01-01	65	Greed
2024-01-01 13:00:00	42608.98	42750.00	42647.85	42715.88	101.7986	2024-01-01	65	Greed
2024-01-01 14:00:00	42581.54	42723.28	42717.53	42635.19	254.3310	2024-01-01	65	Greed
2024-01-01 15:00:00	42601.88	42868.74	42633.57	42797.33	323.6149	2024-01-01	65	Greed
2024-01-01 16:00:00	42680.01	42880.97	42799.37	42742.35	330.0241	2024-01-01	65	Greed
2024-01-01 17:00:00	42720.76	42846.42	42738.88	42833.66	254.8722	2024-01-01	65	Greed
2024-01-01 18:00:00	42835.63	43228.37	42835.63	43120.92	625.4617	2024-01-01	65	Greed
2024-01-01 19:00:00	43106.97	43567.46	43123.82	43547.61	506.4518	2024-01-01	65	Greed
2024-01-01 20:00:00	43537.04	43849.90	43537.04	43701.58	559.3290	2024-01-01	65	Greed
2024-01-01 21:00:00	43467.97	43800.00	43703.06	43632.21	313.9919	2024-01-01	65	Greed
2024-01-01 22:00:00	43389.00	43677.06	43631.79	43546.06	247.5394	2024-01-01	65	Greed
2024-01-01 23:00:00	43545.99	44240.80	43545.99	44220.78	1273.3228	2024-01-01	65	Greed
2024-01-02 00:00:00	44195.13	45250.00	44220.78	45093.17	3023.7931	2024-01-02	71	Greed
2024-01-02 01:00:00	44714.89	45417.45	45093.14	44894.58	1983.0828	2024-01-02	71	Greed
2024-01-02 02:00:00	44891.40	45500.00	44891.41	45485.31	1913.5779	2024-01-02	71	Greed
2024-01-02 03:00:00	45178.34	45601.00	45485.32	45473.97	1512.8339	2024-01-02	71	Greed
2024-01-02 04:00:00	45204.47	45544.10	45476.27	45218.83	681.1595	2024-01-02	71	Greed
2024-01-02 05:00:00	45131.00	45370.54	45218.84	45216.32	484.7321	2024-01-02	71	Greed
2024-01-02 06:00:00	45166.39	45361.61	45214.41	45208.54	495.6376	2024-01-02	71	Greed
2024-01-02 07:00:00	45209.48	45707.69	45209.48	45504.64	976.9213	2024-01-02	71	Greed
2024-01-02 08:00:00	45382.34	45899.96	45504.40	45808.07	759.8822	2024-01-02	71	Greed

We can see now the only NAs are the missing TA values that we can clean with a simple line of code.

```
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 of all let's plot the data to visually verify the data.

TODO Fix rendering of this data (it was fixed previously, could be just a cache issue)

```r
p1 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = close)) +
  geom_line(color = "blue") +
  theme_minimal() +
  labs(title = "BTC-USD Price", y = "Price") +
  scale_y_continuous(labels = scales::comma)

p2 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = hash_rate)) +
  geom_line(color = "red") +
  theme_minimal() +
  labs(title = "Hash Rate", y = "Hash Rate") +
  scale_y_continuous(labels = scales::comma)

p3 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = avg_block_size)) +
  geom_line(color = "green4") +
  theme_minimal() +
  labs(title = "Average Block Size", y = "Size") +
  scale_y_continuous(labels = scales::comma)

p4 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = n_transactions)) +
  geom_line(color = "purple") +
  theme_minimal() +
  labs(title = "Number of Transactions", y = "Count") +
  scale_y_continuous(labels = scales::comma)

p5 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = utxo_count)) +
  geom_line(color = "orange") +
  theme_minimal() +
  labs(title = "UTXO Count", y = "Count") +
  scale_y_continuous(labels = scales::comma)

p6 <- candles_enhanced_cleaned_no_na %>%
  ggplot(aes(x = time, y = value)) +
  geom_line() +
  theme_minimal() +
  labs(
    title = "BTC-USD Fear and Greed Index Evolution",
    x = "Time",
    y = "Fear and Greed Index"
  ) +
  scale_y_continuous(labels = scales::comma)

# For more readability we are only plotting the last 100 candles
p7 <- candles_enhanced_cleaned_no_na %>%
  tail(24) %>%
  ggplot(aes(x = time, y = volume)) +
  geom_segment(aes(xend = time, yend = 0, color = volume)) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "BTC-USD Volume Chart (Last 24 candles)", y = "Volume", x = "") +
```

```r
  theme_tq() +
  theme(legend.position = "none")

combined_plot <- (p1 / p2 / p3 / p4 / p5 / p6 / p7) +
  plot_layout(ncol = 2, heights = c(1, 1, 1, 1)) +
  plot_annotation(
    title = "Bitcoin Price and Blockchain Metrics",
    theme = theme(plot.title = element_text(hjust = 0.5, size = 16))
  ) &
  theme(axis.title.x = element_blank())

combined_plot
```

Find below the candletick chart of BTC-USD.

```r
# For more readability we are only plotting the last 24 candles
p7 <- candles_enhanced_cleaned_no_na %>%
  tail(24) %>%
  mutate(direction = ifelse(close >= open, "up", "down")) %>%
  ggplot(aes(x = time, y = close)) +
  # The shadows (wicks)
  geom_segment(aes(xend = time, y = low, yend = high, color = direction), size = 0.5) +
  # The body
  geom_segment(aes(xend = time, y = open, yend = close, color = direction), size = 5) +
  scale_color_manual(values = c("up" = "darkgreen", "down" = "red")) +
  theme_tq() +
  theme(legend.position = "none") +
  labs(
    title = "BTC-USD Candlestick Chart (Last 24 Candles)",
    x = "Time",
    y = "Price"
  ) +
  scale_y_continuous(labels = scales::comma)
```

And the plot of the different TA.

TODO fix the following for rendering on pdf

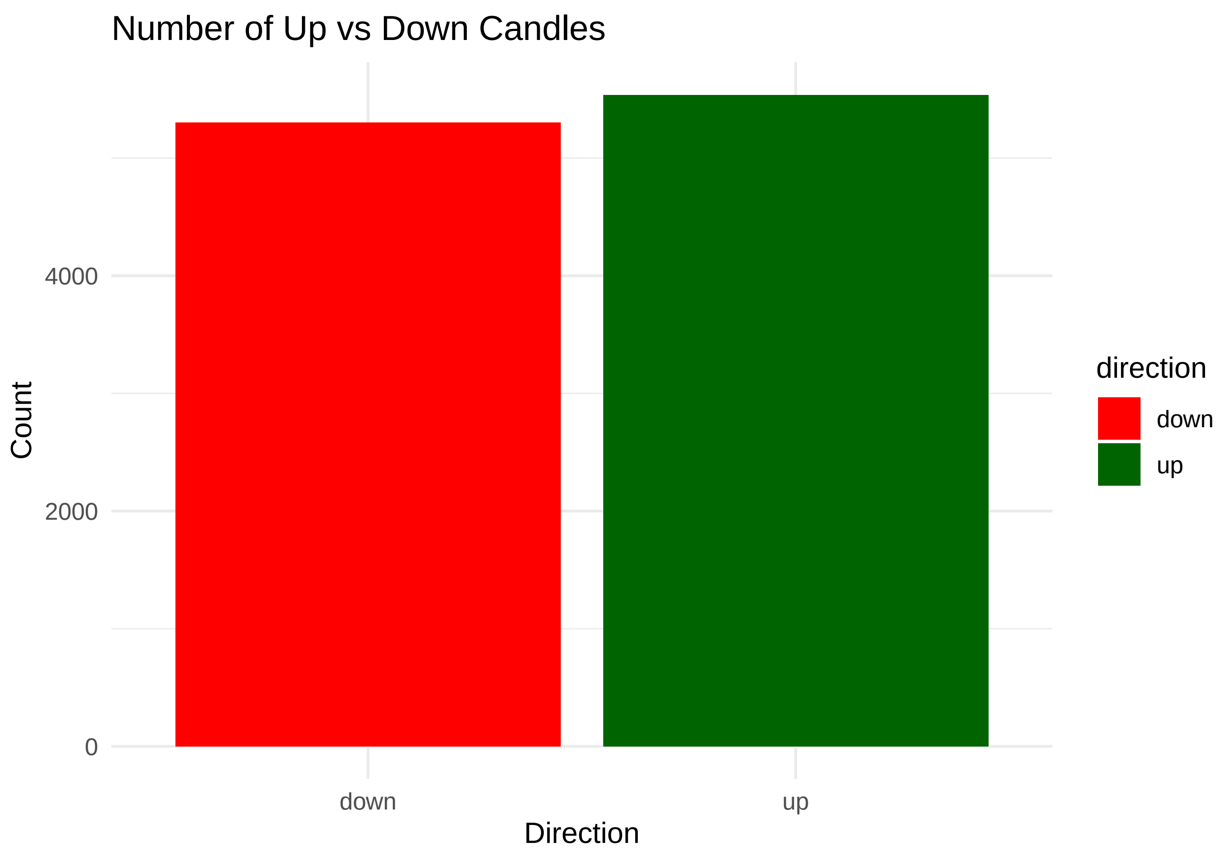
```r
plot_data_ta <- candles_enhanced_cleaned_no_na %>% tail(100)

# ROC Plot
p_roc <- plot_data_ta %>%
  ggplot(aes(x = time, y = roc)) +
  geom_line() +
  labs(title = "Rate of Change (ROC)", y = "ROC") +
  theme_tq() +
  theme(axis.title.x = element_blank())

# Bollinger Bands Plot
p_bbands <- plot_data_ta %>%
  ggplot(aes(x = time, y = close)) +
  geom_line(aes(y = close), color = "blue") + # Close price
  geom_ribbon(aes(ymin = dn, ymax = up), fill = "grey", alpha = 0.4) + # Bollinger Bands area
```

```r
  geom_line(aes(y = mavg), color = "red", linetype = "dashed") + # Moving Average
  labs(title = "Bollinger Bands (BBands)", y = "Price") +
  theme_tq() +
  theme(axis.title.x = element_blank()) +
  scale_y_continuous(labels = scales::comma)

# MACD Plot
p_macd <- plot_data_ta %>%
  ggplot(aes(x = time)) +
  geom_line(aes(y = macd), color = "blue") +  # MACD line
  geom_line(aes(y = signal), color = "red", linetype = "dashed") + # Signal line
  geom_col(aes(y = macd - signal), alpha = 0.5) + # Histogram of MACD - Signal
  labs(title = "MACD", y = "Value") +
  theme_tq() +
  theme(axis.title.x = element_blank())

# RSI Plot
p_rsi <- plot_data_ta %>%
  ggplot(aes(x = time, y = rsi)) +
  geom_line() +
  geom_hline(yintercept = 70, linetype = "dashed", color = "red") +  # Overbought level
  geom_hline(yintercept = 30, linetype = "dashed", color = "darkgreen") + # Oversold level
  labs(title = "Relative Strength Index (RSI)", y = "RSI") +
  theme_tq() +
  theme(axis.title.x = element_blank())

# Combine TA plots
combined_ta_plot <- (p_roc / p_bbands) | (p_macd / p_rsi)

combined_ta_plot + plot_annotation(
  title = "Technical Analysis Indicators (Last 100 Candles)",
  theme = theme(plot.title = element_text(hjust = 0.5, size = 16))
)
```

Comparing with the data from TradingView it seems that all the charts are correct.

Let’s now see how is the distribution of “up” and “down” candles.

```r
candles_enhanced_cleaned_no_na %>%
    mutate(direction = ifelse(close >= open, "up", "down")) %>%
    summarise(up = sum(direction == "up"), down = sum(direction ==
        "down")) %>%
    pivot_longer(cols = everything(), names_to = "direction",
        values_to = "count") %>%
    ggplot(aes(x = direction, y = count, fill = direction)) +
    geom_bar(stat = "identity") + scale_fill_manual(values = c(up = "darkgreen",
    down = "red")) + theme_minimal() + labs(title = "Number of Up vs Down Candles",
    x = "Direction", y = "Count")
```




```
distribution_data <- candles_enhanced_cleaned_no_na %>%
    mutate(direction = ifelse(close >= open, "up", "down")) %>%
    summarise(up = sum(direction == "up"), down = sum(direction ==
        "down")) %>%
    mutate(total = up + down, up_percentage = up/total, down_percentage = down/total)

knitr::kable(distribution_data, format = "simple", caption = "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 exacly 50%.

2.5 Adding lagged candles

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

So we need to create a function to create a dataset containing lagged candles. We also created another function to directly prepare the right data.

```r
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$volu
            i)
        dataset_with_lagged_candles[[paste0("value_lag_", i)]] <- lag(dataset_with_lagged_candles$value
            i)
        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$signa
            i)
        dataset_with_lagged_candles[[paste0("rsi_lag_", i)]] <- lag(dataset_with_lagged_candles$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 function `prepare_dataset` and the we can have directly the final dataset with lagged data.

2.6 Test and training datasets

We put together the code to fix the fear_and_greed_index and to prepare the datasets and split them in 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)
```

```
## Warning in coerce_to_tibble(ret, date_col_name, time_zone, col_rename): Could not rename columns. The
##   Is the length of ‘col_rename‘ the same as the number of columns returned from the ‘mutate_fun‘?
```

```
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 reduced since adding lags introduced a lot of NAs in the first rows, NAs that we removed. Also we decided not to use cross validation to reduce the time of training for the different machine learnings algorithms. Note that we initiated already the project using `set.seed(1)` part of the global variables.

2.7 Machine learning algorithms

Based on some shallow research on the press, we selected the following machine learnings algorithms that seems to work better with our type of dataset:

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

TODO add links reference https://www.neuroquantology.com/open-access/An+Optimized+Machine+Learning+Model+for+Candlestick+Chart+Analysis+to+Predict+Stock+Market+Trends_9861/?download=true https://arxiv.org/pdf/1606.00930

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

2.8 Utility functions

Since we want to compare each algorithms for a different set of features we need a function to create the formula that we will pass to the machine learning 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))
        }
```

```r
    }

    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)))
        } 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)
```

```r
    # 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 learnings algorithms

We are now going to train and compare various machine learning algorithms with different set of selected features and different number of lag.

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

3.1 Simple algorithms

3.1.1 Random guess

We will run a montecarlo simulation of `10000` random guesses of direction and compare it with the test set.

```r
random_guess_simulations <- replicate(10000, {
    estimated_direction <- replicate(nrow(test_set), sample(c("up",
        "down"), 1))
    mean(estimated_direction == test_set$direction)
})
```

```
mean_accuracy <- mean(random_guess_simulations)
print(paste("Random guess simulation results (10000 runs):"))
```

```
## [1] "Random guess simulation results (10000 runs):"
```

```
print(paste("Mean accuracy:", round(mean_accuracy, 4)))
```

```
## [1] "Mean accuracy: 0.4999"
```

3.1.2 Always up

We can also compare this with an always up strategy:

```
# Return always 'up'
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 it compares with returning the same direction as the previous (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 12 lagged data. We will later fine tune that number of lagged candles to see what works the best.

3.2.1 OHLC features

We will first try to use the OHLC features got directly from the coinbase dataset.

```
formula_OHLC_lag_1 <- create_feature_formula(c("open", "high",
    "low", "close"), 1)
formula_OHLC_lag_3 <- create_feature_formula(c("open", "high",
    "low", "close"), 3)
formula_OHLC_lag_5 <- create_feature_formula(c("open", "high",
    "low", "close"), 5)
formula_OHLC_lag_7 <- create_feature_formula(c("open", "high",
    "low", "close"), 7)
formula_OHLC_lag_15 <- create_feature_formula(c("open", "high",
    "low", "close"), 15)

glm_model_OHLC_lag_1 <- train_with_cache(formula_OHLC_lag_1,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_7701c6864aad58f4e602a2a9dfde4116.rds"
```

```
glm_model_OHLC_lag_3 <- train_with_cache(formula_OHLC_lag_3,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_50a0d006040b362ea773e2cbd5516f3e.rds"
```

```
glm_model_OHLC_lag_5 <- train_with_cache(formula_OHLC_lag_5,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_91770134b66103ae408cf84d2d61b721.rds"
```

```
glm_model_OHLC_lag_7 <- train_with_cache(formula_OHLC_lag_7,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_f3455de4f7b30ea76cf766f5c4a43307.rds"
```

```
glm_model_OHLC_lag_15 <- train_with_cache(formula_OHLC_lag_15,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_9cd031b1f2a0ee6214a25e214d453f4c.rds"
```

```
rpart_model_OHLC_lag_1 <- train_with_cache(formula_OHLC_lag_1,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_7701c6864aad58f4e602a2a9dfde4116.rds"
```

```r
rpart_model_OHLC_lag_3 <- train_with_cache(formula_OHLC_lag_3,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_50a0d006040b362ea773e2cbd5516f3e.rds"
```

```r
rpart_model_OHLC_lag_5 <- train_with_cache(formula_OHLC_lag_5,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_91770134b66103ae408cf84d2d61b721.rds"
```

```r
rpart_model_OHLC_lag_7 <- train_with_cache(formula_OHLC_lag_7,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_f3455de4f7b30ea76cf766f5c4a43307.rds"
```

```r
rpart_model_OHLC_lag_15 <- train_with_cache(formula_OHLC_lag_15,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_9cd031b1f2a0ee6214a25e214d453f4c.rds"
```

```r
rf_model_OHLC_lag_1 <- train_with_cache(formula_OHLC_lag_1, train_set,
    "rf")
```

```
## [1] "Model loaded from cache: models/rf_7701c6864aad58f4e602a2a9dfde4116.rds"
```

```r
rf_model_OHLC_lag_3 <- train_with_cache(formula_OHLC_lag_3, train_set,
    "rf")
```

```
## [1] "Model loaded from cache: models/rf_50a0d006040b362ea773e2cbd5516f3e.rds"
```

```r
rf_model_OHLC_lag_5 <- train_with_cache(formula_OHLC_lag_5, train_set,
    "rf")
```

```
## [1] "Model loaded from cache: models/rf_91770134b66103ae408cf84d2d61b721.rds"
```

```r
rf_model_OHLC_lag_7 <- train_with_cache(formula_OHLC_lag_7, train_set,
    "rf")
```

```
## [1] "Model loaded from cache: models/rf_f3455de4f7b30ea76cf766f5c4a43307.rds"
```

```r
rf_model_OHLC_lag_15 <- train_with_cache(formula_OHLC_lag_15,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_9cd031b1f2a0ee6214a25e214d453f4c.rds"
```

```r
knn_model_OHLC_lag_1 <- train_with_cache(formula_OHLC_lag_1,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_7701c6864aad58f4e602a2a9dfde4116.rds"
```

```r
knn_model_OHLC_lag_3 <- train_with_cache(formula_OHLC_lag_3,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_50a0d006040b362ea773e2cbd5516f3e.rds"
```

```r
knn_model_OHLC_lag_5 <- train_with_cache(formula_OHLC_lag_5,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_91770134b66103ae408cf84d2d61b721.rds"
```

```r
knn_model_OHLC_lag_7 <- train_with_cache(formula_OHLC_lag_7,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_f3455de4f7b30ea76cf766f5c4a43307.rds"
```

```r
knn_model_OHLC_lag_15 <- train_with_cache(formula_OHLC_lag_15,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_9cd031b1f2a0ee6214a25e214d453f4c.rds"
```

```r
gbm_model_OHLC_lag_1 <- train_with_cache(formula_OHLC_lag_1,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_7701c6864aad58f4e602a2a9dfde4116.rds"
```

```r
gbm_model_OHLC_lag_3 <- train_with_cache(formula_OHLC_lag_3,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_50a0d006040b362ea773e2cbd5516f3e.rds"
```

```r
gbm_model_OHLC_lag_5 <- train_with_cache(formula_OHLC_lag_5,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_91770134b66103ae408cf84d2d61b721.rds"
```

```r
gbm_model_OHLC_lag_7 <- train_with_cache(formula_OHLC_lag_7,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_f3455de4f7b30ea76cf766f5c4a43307.rds"
```

```r
gbm_model_OHLC_lag_15 <- train_with_cache(formula_OHLC_lag_15,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_9cd031b1f2a0ee6214a25e214d453f4c.rds"
```

```r
results_OHLC <- evaluate_models("OHLC", test_set)
knitr::kable(results_OHLC, format = "simple", caption = "Model comparison for OHLC features")
```

Table 11: Model comparison for OHLC features

	model	model_type	lag	accuracy	rank
1	glm_model_OHLC_lag_1	glm	1	0.5410896	1
3	glm_model_OHLC_lag_5	glm	5	0.5392428	2
2	glm_model_OHLC_lag_3	glm	3	0.5364728	3
4	glm_model_OHLC_lag_7	glm	7	0.5360111	4
5	glm_model_OHLC_lag_15	glm	15	0.5203139	5
21	gbm_model_OHLC_lag_1	gbm	1	0.5170822	6
11	rpart_model_OHLC_lag_1	rpart	1	0.5152355	7
25	gbm_model_OHLC_lag_15	gbm	15	0.5152355	8
22	gbm_model_OHLC_lag_3	gbm	3	0.5147738	9
14	rpart_model_OHLC_lag_7	rpart	7	0.5120037	10
24	gbm_model_OHLC_lag_7	gbm	7	0.5120037	11
23	gbm_model_OHLC_lag_5	gbm	5	0.5115420	12
12	rpart_model_OHLC_lag_3	rpart	3	0.5110803	13
13	rpart_model_OHLC_lag_5	rpart	5	0.5110803	14
15	rpart_model_OHLC_lag_15	rpart	15	0.5110803	15
9	rf_model_OHLC_lag_7	rf	7	0.5036934	16
7	rf_model_OHLC_lag_3	rf	3	0.5032318	17
6	rf_model_OHLC_lag_1	rf	1	0.4986150	18
19	knn_model_OHLC_lag_7	knn	7	0.4958449	19
8	rf_model_OHLC_lag_5	rf	5	0.4939982	20
17	knn_model_OHLC_lag_3	knn	3	0.4930748	21
18	knn_model_OHLC_lag_5	knn	5	0.4903047	22
16	knn_model_OHLC_lag_1	knn	1	0.4884580	23
20	knn_model_OHLC_lag_15	knn	15	0.4704524	24
10	rf_model_OHLC_lag_15	rf	15	0.4699908	25

```r
summary_stats_OHLC <- aggregate(accuracy ~ model_type, data = results_OHLC,
    FUN = function(x) c(mean = mean(x), sd = sd(x), max = max(x)))

summary_stats_OHLC <- data.frame(model_type = summary_stats_OHLC$model_type,
    mean_accuracy = summary_stats_OHLC$accuracy[, "mean"], sd_accuracy = summary_stats_OHLC$accuracy[,
        "sd"], max_accuracy = summary_stats_OHLC$accuracy[, "max"])

knitr::kable(summary_stats_OHLC, format = "simple", caption = "Summary statistics for OHLC features",
    digits = 4, col.names = c("Model Type", "Mean Accuracy",
        "SD Accuracy", "Max Accuracy"))
```

Table 12: Summary statistics for OHLC features

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5141	0.0023	0.5171
glm	0.5346	0.0083	0.5411
knn	0.4876	0.0100	0.4958
rf	0.4939	0.0139	0.5037
rpart	0.5121	0.0018	0.5152

3.2.2 Candle features

```
formula_candles_lag_1 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close"),
    1)
formula_candles_lag_3 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close"),
    3)
formula_candles_lag_5 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close"),
    5)
formula_candles_lag_7 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close"),
    7)
formula_candles_lag_15 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close"),
    15)

glm_model_candles_lag_1 <- train_with_cache(formula_candles_lag_1,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_9b534c2cad659d2d11ccbdf479cba62d.rds"
```

```
glm_model_candles_lag_3 <- train_with_cache(formula_candles_lag_3,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_8262167bc0017623ec10cac265090b04.rds"
```

```
glm_model_candles_lag_5 <- train_with_cache(formula_candles_lag_5,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_958600554c2ba97ab4bffdf7267111a1.rds"
```

```
glm_model_candles_lag_7 <- train_with_cache(formula_candles_lag_7,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_38a020cd45bedd311f240c4fb85a8261.rds"
```

```r
glm_model_candles_lag_15 <- train_with_cache(formula_candles_lag_15,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_1871e23c6166b1243ab9ee3b6f97b8e9.rds"
```

```r
rpart_model_candles_lag_1 <- train_with_cache(formula_candles_lag_1,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_9b534c2cad659d2d11ccbdf479cba62d.rds"
```

```r
rpart_model_candles_lag_3 <- train_with_cache(formula_candles_lag_3,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_8262167bc0017623ec10cac265090b04.rds"
```

```r
rpart_model_candles_lag_5 <- train_with_cache(formula_candles_lag_5,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_958600554c2ba97ab4bffdf7267111a1.rds"
```

```r
rpart_model_candles_lag_7 <- train_with_cache(formula_candles_lag_7,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_38a020cd45bedd311f240c4fb85a8261.rds"
```

```r
rpart_model_candles_lag_15 <- train_with_cache(formula_candles_lag_15,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_1871e23c6166b1243ab9ee3b6f97b8e9.rds"
```

```r
rf_model_candles_lag_1 <- train_with_cache(formula_candles_lag_1,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_9b534c2cad659d2d11ccbdf479cba62d.rds"
```

```r
rf_model_candles_lag_3 <- train_with_cache(formula_candles_lag_3,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_8262167bc0017623ec10cac265090b04.rds"
```

```r
rf_model_candles_lag_5 <- train_with_cache(formula_candles_lag_5,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_958600554c2ba97ab4bffdf7267111a1.rds"
```

```r
rf_model_candles_lag_7 <- train_with_cache(formula_candles_lag_7,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_38a020cd45bedd311f240c4fb85a8261.rds"
```

```r
rf_model_candles_lag_15 <- train_with_cache(formula_candles_lag_15,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_1871e23c6166b1243ab9ee3b6f97b8e9.rds"
```

```r
knn_model_candles_lag_1 <- train_with_cache(formula_candles_lag_1,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_9b534c2cad659d2d11ccbdf479cba62d.rds"
```

```r
knn_model_candles_lag_3 <- train_with_cache(formula_candles_lag_3,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_8262167bc0017623ec10cac265090b04.rds"
```

```r
knn_model_candles_lag_5 <- train_with_cache(formula_candles_lag_5,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_958600554c2ba97ab4bffdf7267111a1.rds"
```

```r
knn_model_candles_lag_7 <- train_with_cache(formula_candles_lag_7,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_38a020cd45bedd311f240c4fb85a8261.rds"
```

```r
knn_model_candles_lag_15 <- train_with_cache(formula_candles_lag_15,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_1871e23c6166b1243ab9ee3b6f97b8e9.rds"
```

```r
gbm_model_candles_lag_1 <- train_with_cache(formula_candles_lag_1,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_9b534c2cad659d2d11ccbdf479cba62d.rds"
```

```r
gbm_model_candles_lag_3 <- train_with_cache(formula_candles_lag_3,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_8262167bc0017623ec10cac265090b04.rds"
```

```r
gbm_model_candles_lag_5 <- train_with_cache(formula_candles_lag_5,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_958600554c2ba97ab4bffdf7267111a1.rds"
```

```r
gbm_model_candles_lag_7 <- train_with_cache(formula_candles_lag_7,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_38a020cd45bedd311f240c4fb85a8261.rds"
```

```r
gbm_model_candles_lag_15 <- train_with_cache(formula_candles_lag_15,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_1871e23c6166b1243ab9ee3b6f97b8e9.rds"
```

```r
results_candles <- evaluate_models("candles", test_set)
results_candles
```

```
##                          model model_type lag  accuracy rank
## 4      glm_model_candles_lag_7        glm   7 0.5433980    1
## 13   rpart_model_candles_lag_5      rpart   5 0.5401662    2
## 2      glm_model_candles_lag_3        glm   3 0.5397045    3
## 24     gbm_model_candles_lag_7        gbm   7 0.5378578    4
## 3      glm_model_candles_lag_5        glm   5 0.5364728    5
## 11   rpart_model_candles_lag_1      rpart   1 0.5341644    6
## 12   rpart_model_candles_lag_3      rpart   3 0.5341644    7
## 14   rpart_model_candles_lag_7      rpart   7 0.5341644    8
## 9       rf_model_candles_lag_7         rf   7 0.5332410    9
## 20    knn_model_candles_lag_15        knn  15 0.5327793   10
## 17     knn_model_candles_lag_3        knn   3 0.5318560   11
## 15 rpart_model_candles_lag_15      rpart  15 0.5295476   12
## 10     rf_model_candles_lag_15         rf  15 0.5290859   13
## 18     knn_model_candles_lag_5        knn   5 0.5286242   14
## 21     gbm_model_candles_lag_1        gbm   1 0.5272392   15
## 5     glm_model_candles_lag_15        glm  15 0.5258541   16
## 1      glm_model_candles_lag_1        glm   1 0.5240074   17
## 22     gbm_model_candles_lag_3        gbm   3 0.5216990   18
## 16     knn_model_candles_lag_1        knn   1 0.5212373   19
## 23     gbm_model_candles_lag_5        gbm   5 0.5193906   20
## 19     knn_model_candles_lag_7        knn   7 0.5184672   21
## 7       rf_model_candles_lag_3         rf   3 0.5152355   22
## 25    gbm_model_candles_lag_15        gbm  15 0.5115420   23
## 6       rf_model_candles_lag_1         rf   1 0.5092336   24
## 8       rf_model_candles_lag_5         rf   5 0.5083102   25
```

```r
summary_stats_candles <- aggregate(accuracy ~ model_type, data = results_candles,
    FUN = function(x) c(mean = mean(x), sd = sd(x), max = max(x)))

summary_stats_candles <- data.frame(model_type = summary_stats_candles$model_type,
    mean_accuracy = summary_stats_candles$accuracy[, "mean"],
```

```r
    sd_accuracy = summary_stats_candles$accuracy[, "sd"], max_accuracy = summary_stats_candles$accuracy
        "max"])

knitr::kable(summary_stats_candles, format = "simple", caption = "Summary statistics for candles feature
    digits = 4, col.names = c("Model Type", "Mean Accuracy",
        "SD Accuracy", "Max Accuracy"))
```

Table 13: Summary statistics for candles features

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5235	0.0098	0.5379
glm	0.5339	0.0086	0.5434
knn	0.5266	0.0064	0.5328
rf	0.5190	0.0115	0.5332
rpart	0.5344	0.0038	0.5402

3.2.3 Candles features and fear and greed index

```r
formula_candles_fg_lag_1 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value"), 1)
formula_candles_fg_lag_3 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value"), 3)
formula_candles_fg_lag_5 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value"), 5)
formula_candles_fg_lag_7 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value"), 7)
formula_candles_fg_lag_15 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value"), 15)

glm_model_candles_fg_lag_1 <- train_with_cache(formula_candles_fg_lag_1,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_cf057a7ef6d4f5e81691ceac20ea2fd2.rds"
```

```r
glm_model_candles_fg_lag_3 <- train_with_cache(formula_candles_fg_lag_3,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_71866ce0fcf7c892fd54b244cd1bb814.rds"
```

```r
glm_model_candles_fg_lag_5 <- train_with_cache(formula_candles_fg_lag_5,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_11d9f81d550f80b7c5f969323ad0d2e6.rds"
```

```r
glm_model_candles_fg_lag_7 <- train_with_cache(formula_candles_fg_lag_7,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_59f2b72bece81ca2f8fb72c51aa4df4e.rds"
```

```r
glm_model_candles_fg_lag_15 <- train_with_cache(formula_candles_fg_lag_15,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_3d5c2979b4479dcbbe4b94db911cf867.rds"
```

```r
rpart_model_candles_fg_lag_1 <- train_with_cache(formula_candles_fg_lag_1,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_cf057a7ef6d4f5e81691ceac20ea2fd2.rds"
```

```r
rpart_model_candles_fg_lag_3 <- train_with_cache(formula_candles_fg_lag_3,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_71866ce0fcf7c892fd54b244cd1bb814.rds"
```

```r
rpart_model_candles_fg_lag_5 <- train_with_cache(formula_candles_fg_lag_5,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_11d9f81d550f80b7c5f969323ad0d2e6.rds"
```

```r
rpart_model_candles_fg_lag_7 <- train_with_cache(formula_candles_fg_lag_7,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_59f2b72bece81ca2f8fb72c51aa4df4e.rds"
```

```r
rpart_model_candles_fg_lag_15 <- train_with_cache(formula_candles_fg_lag_15,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_3d5c2979b4479dcbbe4b94db911cf867.rds"
```

```r
rf_model_candles_fg_lag_1 <- train_with_cache(formula_candles_fg_lag_1,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_cf057a7ef6d4f5e81691ceac20ea2fd2.rds"
```

```r
rf_model_candles_fg_lag_3 <- train_with_cache(formula_candles_fg_lag_3,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_71866ce0fcf7c892fd54b244cd1bb814.rds"
```

```r
rf_model_candles_fg_lag_5 <- train_with_cache(formula_candles_fg_lag_5,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_11d9f81d550f80b7c5f969323ad0d2e6.rds"
```

```r
rf_model_candles_fg_lag_7 <- train_with_cache(formula_candles_fg_lag_7,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_59f2b72bece81ca2f8fb72c51aa4df4e.rds"
```

```r
rf_model_candles_fg_lag_15 <- train_with_cache(formula_candles_fg_lag_15,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_3d5c2979b4479dcbbe4b94db911cf867.rds"
```

```r
knn_model_candles_fg_lag_1 <- train_with_cache(formula_candles_fg_lag_1,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_cf057a7ef6d4f5e81691ceac20ea2fd2.rds"
```

```r
knn_model_candles_fg_lag_3 <- train_with_cache(formula_candles_fg_lag_3,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_71866ce0fcf7c892fd54b244cd1bb814.rds"
```

```r
knn_model_candles_fg_lag_5 <- train_with_cache(formula_candles_fg_lag_5,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_11d9f81d550f80b7c5f969323ad0d2e6.rds"
```

```r
knn_model_candles_fg_lag_7 <- train_with_cache(formula_candles_fg_lag_7,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_59f2b72bece81ca2f8fb72c51aa4df4e.rds"
```

```r
knn_model_candles_fg_lag_15 <- train_with_cache(formula_candles_fg_lag_15,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_3d5c2979b4479dcbbe4b94db911cf867.rds"
```

```r
gbm_model_candles_fg_lag_1 <- train_with_cache(formula_candles_fg_lag_1,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_cf057a7ef6d4f5e81691ceac20ea2fd2.rds"
```

```r
gbm_model_candles_fg_lag_3 <- train_with_cache(formula_candles_fg_lag_3,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_71866ce0fcf7c892fd54b244cd1bb814.rds"
```

```r
gbm_model_candles_fg_lag_5 <- train_with_cache(formula_candles_fg_lag_5,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_11d9f81d550f80b7c5f969323ad0d2e6.rds"
```

```r
gbm_model_candles_fg_lag_7 <- train_with_cache(formula_candles_fg_lag_7,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_59f2b72bece81ca2f8fb72c51aa4df4e.rds"
```

```r
gbm_model_candles_fg_lag_15 <- train_with_cache(formula_candles_fg_lag_15,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_3d5c2979b4479dcbbe4b94db911cf867.rds"
```

```r
results_candles_fg <- evaluate_models("candles_fg", test_set)
results_candles_fg
```

```
##                          model model_type lag  accuracy rank
## 24    gbm_model_candles_fg_lag_7        gbm   7 0.5415512    1
## 3     glm_model_candles_fg_lag_5        glm   5 0.5410896    2
## 2     glm_model_candles_fg_lag_3        glm   3 0.5401662    3
## 4     glm_model_candles_fg_lag_7        glm   7 0.5383195    4
## 11  rpart_model_candles_fg_lag_1      rpart   1 0.5341644    5
## 12  rpart_model_candles_fg_lag_3      rpart   3 0.5341644    6
## 13  rpart_model_candles_fg_lag_5      rpart   5 0.5341644    7
## 14  rpart_model_candles_fg_lag_7      rpart   7 0.5341644    8
## 9      rf_model_candles_fg_lag_7         rf   7 0.5327793    9
## 21    gbm_model_candles_fg_lag_1        gbm   1 0.5304709   10
## 25   gbm_model_candles_fg_lag_15        gbm  15 0.5263158   11
## 1     glm_model_candles_fg_lag_1        glm   1 0.5240074   12
## 19    knn_model_candles_fg_lag_7        knn   7 0.5235457   13
## 22    gbm_model_candles_fg_lag_3        gbm   3 0.5221607   14
## 23    gbm_model_candles_fg_lag_5        gbm   5 0.5221607   15
## 5    glm_model_candles_fg_lag_15        glm  15 0.5207756   16
## 6      rf_model_candles_fg_lag_1         rf   1 0.5193906   17
## 20   knn_model_candles_fg_lag_15        knn  15 0.5175439   18
## 17    knn_model_candles_fg_lag_3        knn   3 0.5133887   19
## 8      rf_model_candles_fg_lag_5         rf   5 0.5124654   20
## 15 rpart_model_candles_fg_lag_15      rpart  15 0.5110803   21
## 7      rf_model_candles_fg_lag_3         rf   3 0.5096953   22
## 16    knn_model_candles_fg_lag_1        knn   1 0.5092336   23
## 18    knn_model_candles_fg_lag_5        knn   5 0.5041551   24
## 10    rf_model_candles_fg_lag_15         rf  15 0.5032318   25
```

```r
summary_stats_candles_fg <- aggregate(accuracy ~ model_type,
    data = results_candles_fg, FUN = function(x) c(mean = mean(x),
        sd = sd(x), max = max(x)))

summary_stats_candles_fg <- data.frame(model_type = summary_stats_candles_fg$model_type,
    mean_accuracy = summary_stats_candles_fg$accuracy[, "mean"],
    sd_accuracy = summary_stats_candles_fg$accuracy[, "sd"],
    max_accuracy = summary_stats_candles_fg$accuracy[, "max"])

knitr::kable(summary_stats_candles_fg, format = "simple", caption = "Summary statistics for candles fea
    digits = 4, col.names = c("Model Type", "Mean Accuracy",
        "SD Accuracy", "Max Accuracy"))
```

Table 14: Summary statistics for candles features and fear and greed index

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5285	0.0081	0.5416
glm	0.5329	0.0097	0.5411
knn	0.5136	0.0075	0.5235
rf	0.5155	0.0113	0.5328
rpart	0.5295	0.0103	0.5342

3.2.4 Candles features, fear and greed index and chain data

```r
formula_candles_fg_chain_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"), 1)
formula_candles_fg_chain_lag_3 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value", "hash_rate", "avg_block_size", "n_transactions",
    "utxo_count"), 3)
formula_candles_fg_chain_lag_5 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value", "hash_rate", "avg_block_size", "n_transactions",
    "utxo_count"), 5)
formula_candles_fg_chain_lag_7 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value", "hash_rate", "avg_block_size", "n_transactions",
    "utxo_count"), 7)
formula_candles_fg_chain_lag_15 <- create_feature_formula(c("body_size",
    "upper_shadow_size", "lower_shadow_size", "direction", "close",
    "value", "hash_rate", "avg_block_size", "n_transactions",
    "utxo_count"), 15)

glm_model_candles_fg_chain_lag_1 <- train_with_cache(formula_candles_fg_chain_lag_1,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_0a6cd5020cb3d8af39c3f49be89b5c7e.rds"
```

```r
glm_model_candles_fg_chain_lag_3 <- train_with_cache(formula_candles_fg_chain_lag_3,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_2d6da1c4f54dc3ea3a5868848f610e23.rds"
```

```r
glm_model_candles_fg_chain_lag_5 <- train_with_cache(formula_candles_fg_chain_lag_5,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_01c4332d1952c29269f5e20f212cbf84.rds"
```

```r
glm_model_candles_fg_chain_lag_7 <- train_with_cache(formula_candles_fg_chain_lag_7,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_f9fd1e419d7a930adb8bbfcc8573e064.rds"
```

```r
glm_model_candles_fg_chain_lag_15 <- train_with_cache(formula_candles_fg_chain_lag_15,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_533d1e015be654bef82f08c63f29a4c3.rds"
```

```r
rpart_model_candles_fg_chain_lag_1 <- train_with_cache(formula_candles_fg_chain_lag_1,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_0a6cd5020cb3d8af39c3f49be89b5c7e.rds"
```

```r
rpart_model_candles_fg_chain_lag_3 <- train_with_cache(formula_candles_fg_chain_lag_3,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_2d6da1c4f54dc3ea3a5868848f610e23.rds"
```

```r
rpart_model_candles_fg_chain_lag_5 <- train_with_cache(formula_candles_fg_chain_lag_5,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_01c4332d1952c29269f5e20f212cbf84.rds"
```

```r
rpart_model_candles_fg_chain_lag_7 <- train_with_cache(formula_candles_fg_chain_lag_7,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_f9fd1e419d7a930adb8bbfcc8573e064.rds"
```

```r
rpart_model_candles_fg_chain_lag_15 <- train_with_cache(formula_candles_fg_chain_lag_15,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_533d1e015be654bef82f08c63f29a4c3.rds"
```

```r
rf_model_candles_fg_chain_lag_1 <- train_with_cache(formula_candles_fg_chain_lag_1,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_0a6cd5020cb3d8af39c3f49be89b5c7e.rds"
```

```r
rf_model_candles_fg_chain_lag_3 <- train_with_cache(formula_candles_fg_chain_lag_3,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_2d6da1c4f54dc3ea3a5868848f610e23.rds"
```

```r
rf_model_candles_fg_chain_lag_5 <- train_with_cache(formula_candles_fg_chain_lag_5,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_01c4332d1952c29269f5e20f212cbf84.rds"
```

```r
rf_model_candles_fg_chain_lag_7 <- train_with_cache(formula_candles_fg_chain_lag_7,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_f9fd1e419d7a930adb8bbfcc8573e064.rds"
```

```r
rf_model_candles_fg_chain_lag_15 <- train_with_cache(formula_candles_fg_chain_lag_15,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_533d1e015be654bef82f08c63f29a4c3.rds"
```

```r
knn_model_candles_fg_chain_lag_1 <- train_with_cache(formula_candles_fg_chain_lag_1,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_0a6cd5020cb3d8af39c3f49be89b5c7e.rds"
```

```r
knn_model_candles_fg_chain_lag_3 <- train_with_cache(formula_candles_fg_chain_lag_3,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_2d6da1c4f54dc3ea3a5868848f610e23.rds"
```

```r
knn_model_candles_fg_chain_lag_5 <- train_with_cache(formula_candles_fg_chain_lag_5,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_01c4332d1952c29269f5e20f212cbf84.rds"
```

```r
knn_model_candles_fg_chain_lag_7 <- train_with_cache(formula_candles_fg_chain_lag_7,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_f9fd1e419d7a930adb8bbfcc8573e064.rds"
```

```r
knn_model_candles_fg_chain_lag_15 <- train_with_cache(formula_candles_fg_chain_lag_15,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_533d1e015be654bef82f08c63f29a4c3.rds"
```

```r
gbm_model_candles_fg_chain_lag_1 <- train_with_cache(formula_candles_fg_chain_lag_1,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_0a6cd5020cb3d8af39c3f49be89b5c7e.rds"
```

```r
gbm_model_candles_fg_chain_lag_3 <- train_with_cache(formula_candles_fg_chain_lag_3,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_2d6da1c4f54dc3ea3a5868848f610e23.rds"
```

```r
gbm_model_candles_fg_chain_lag_5 <- train_with_cache(formula_candles_fg_chain_lag_5,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_01c4332d1952c29269f5e20f212cbf84.rds"
```

```r
gbm_model_candles_fg_chain_lag_7 <- train_with_cache(formula_candles_fg_chain_lag_7,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_f9fd1e419d7a930adb8bbfcc8573e064.rds"
```

```r
gbm_model_candles_fg_chain_lag_15 <- train_with_cache(formula_candles_fg_chain_lag_15,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_533d1e015be654bef82f08c63f29a4c3.rds"
```

```r
results_candles_fg_chain <- evaluate_models("candles_fg_chain",
    test_set)
results_candles_fg_chain
```

```
##                                    model model_type lag  accuracy rank
## 11  rpart_model_candles_fg_chain_lag_1      rpart   1 0.5341644    1
## 12  rpart_model_candles_fg_chain_lag_3      rpart   3 0.5341644    2
## 13  rpart_model_candles_fg_chain_lag_5      rpart   5 0.5341644    3
## 14  rpart_model_candles_fg_chain_lag_7      rpart   7 0.5341644    4
## 15 rpart_model_candles_fg_chain_lag_15      rpart  15 0.5341644    5
## 21    gbm_model_candles_fg_chain_lag_1        gbm   1 0.5327793    6
## 24    gbm_model_candles_fg_chain_lag_7        gbm   7 0.5290859    7
## 22    gbm_model_candles_fg_chain_lag_3        gbm   3 0.5263158    8
## 5    glm_model_candles_fg_chain_lag_15        glm  15 0.5258541    9
## 2     glm_model_candles_fg_chain_lag_3        glm   3 0.5253924   10
## 25   gbm_model_candles_fg_chain_lag_15        gbm  15 0.5221607   11
## 4     glm_model_candles_fg_chain_lag_7        glm   7 0.5212373   12
## 3     glm_model_candles_fg_chain_lag_5        glm   5 0.5207756   13
```

```
## 23    gbm_model_candles_fg_chain_lag_5        gbm   5 0.5189289   14
## 19    knn_model_candles_fg_chain_lag_7        knn   7 0.5152355   15
## 9      rf_model_candles_fg_chain_lag_7         rf   7 0.5120037   16
## 1     glm_model_candles_fg_chain_lag_1        glm   1 0.5106187   17
## 7      rf_model_candles_fg_chain_lag_3         rf   3 0.5106187   18
## 18    knn_model_candles_fg_chain_lag_5        knn   5 0.5064635   19
## 16    knn_model_candles_fg_chain_lag_1        knn   1 0.5055402   20
## 6      rf_model_candles_fg_chain_lag_1         rf   1 0.5027701   21
## 17    knn_model_candles_fg_chain_lag_3        knn   3 0.4995383   22
## 10    rf_model_candles_fg_chain_lag_15         rf  15 0.4986150   23
## 8      rf_model_candles_fg_chain_lag_5         rf   5 0.4981533   24
## 20   knn_model_candles_fg_chain_lag_15        knn  15 0.4879963   25
```

```
summary_stats_candles_fg_chain <- aggregate(accuracy ~ model_type,
    data = results_candles_fg_chain, FUN = function(x) c(mean = mean(x),
        sd = sd(x), max = max(x)))

summary_stats_candles_fg_chain <- data.frame(model_type = summary_stats_candles_fg_chain$model_type,
    mean_accuracy = summary_stats_candles_fg_chain$accuracy[,
        "mean"], sd_accuracy = summary_stats_candles_fg_chain$accuracy[,
        "sd"], max_accuracy = summary_stats_candles_fg_chain$accuracy[,
        "max"])

knitr::kable(summary_stats_candles_fg_chain, format = "simple",
    caption = "Summary statistics for candles features, fear and greed index and chain data",
    digits = 4, col.names = c("Model Type", "Mean Accuracy",
        "SD Accuracy", "Max Accuracy"))
```

Table 15: Summary statistics for candles features, fear and greed index and chain data

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5285	0.0081	0.5416
glm	0.5329	0.0097	0.5411
knn	0.5134	0.0071	0.5231
rf	0.5168	0.0111	0.5351
rpart	0.5295	0.0103	0.5342

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

```
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"), 1)
formula_candles_fg_chain_ta_lag_3 <- 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"), 3)
```

```r
formula_candles_fg_chain_ta_lag_5 <- 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"), 5)
formula_candles_fg_chain_ta_lag_7 <- 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"), 7)
formula_candles_fg_chain_ta_lag_15 <- 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"), 15)

glm_model_candles_fg_chain_ta_lag_1 <- train_with_cache(formula_candles_fg_chain_ta_lag_1,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_950928ee0828cd0c9299529274a8cd55.rds"
```

```r
glm_model_candles_fg_chain_ta_lag_3 <- train_with_cache(formula_candles_fg_chain_ta_lag_3,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_1197d5a7e07e5603640c48925a3ef5cd.rds"
```

```r
glm_model_candles_fg_chain_ta_lag_5 <- train_with_cache(formula_candles_fg_chain_ta_lag_5,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_e8b4ec46cd84c79cd5cc650553b52bd8.rds"
```

```r
glm_model_candles_fg_chain_ta_lag_7 <- train_with_cache(formula_candles_fg_chain_ta_lag_7,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_20b24c6f306d90c7bb9e8dc3389712a1.rds"
```

```r
glm_model_candles_fg_chain_ta_lag_15 <- train_with_cache(formula_candles_fg_chain_ta_lag_15,
    train_set, "glm")
```

```
## [1] "Model loaded from cache: models/glm_195a16ef06b6ba7503521f913956918f.rds"
```

```r
rpart_model_candles_fg_chain_ta_lag_1 <- train_with_cache(formula_candles_fg_chain_ta_lag_1,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_950928ee0828cd0c9299529274a8cd55.rds"
```

```r
rpart_model_candles_fg_chain_ta_lag_3 <- train_with_cache(formula_candles_fg_chain_ta_lag_3,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_1197d5a7e07e5603640c48925a3ef5cd.rds"
```

```r
rpart_model_candles_fg_chain_ta_lag_5 <- train_with_cache(formula_candles_fg_chain_ta_lag_5,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_e8b4ec46cd84c79cd5cc650553b52bd8.rds"
```

```r
rpart_model_candles_fg_chain_ta_lag_7 <- train_with_cache(formula_candles_fg_chain_ta_lag_7,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_20b24c6f306d90c7bb9e8dc3389712a1.rds"
```

```r
rpart_model_candles_fg_chain_ta_lag_15 <- train_with_cache(formula_candles_fg_chain_ta_lag_15,
    train_set, "rpart")
```

```
## [1] "Model loaded from cache: models/rpart_195a16ef06b6ba7503521f913956918f.rds"
```

```r
rf_model_candles_fg_chain_ta_lag_1 <- train_with_cache(formula_candles_fg_chain_ta_lag_1,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_950928ee0828cd0c9299529274a8cd55.rds"
```

```r
rf_model_candles_fg_chain_ta_lag_3 <- train_with_cache(formula_candles_fg_chain_ta_lag_3,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_1197d5a7e07e5603640c48925a3ef5cd.rds"
```

```r
rf_model_candles_fg_chain_ta_lag_5 <- train_with_cache(formula_candles_fg_chain_ta_lag_5,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_e8b4ec46cd84c79cd5cc650553b52bd8.rds"
```

```r
rf_model_candles_fg_chain_ta_lag_7 <- train_with_cache(formula_candles_fg_chain_ta_lag_7,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_20b24c6f306d90c7bb9e8dc3389712a1.rds"
```

```r
rf_model_candles_fg_chain_ta_lag_15 <- train_with_cache(formula_candles_fg_chain_ta_lag_15,
    train_set, "rf")
```

```
## [1] "Model loaded from cache: models/rf_195a16ef06b6ba7503521f913956918f.rds"
```

```r
knn_model_candles_fg_chain_ta_lag_1 <- train_with_cache(formula_candles_fg_chain_ta_lag_1,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_950928ee0828cd0c9299529274a8cd55.rds"
```

```r
knn_model_candles_fg_chain_ta_lag_3 <- train_with_cache(formula_candles_fg_chain_ta_lag_3,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_1197d5a7e07e5603640c48925a3ef5cd.rds"
```

```r
knn_model_candles_fg_chain_ta_lag_5 <- train_with_cache(formula_candles_fg_chain_ta_lag_5,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_e8b4ec46cd84c79cd5cc650553b52bd8.rds"
```

```r
knn_model_candles_fg_chain_ta_lag_7 <- train_with_cache(formula_candles_fg_chain_ta_lag_7,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_20b24c6f306d90c7bb9e8dc3389712a1.rds"
```

```r
knn_model_candles_fg_chain_ta_lag_15 <- train_with_cache(formula_candles_fg_chain_ta_lag_15,
    train_set, "knn")
```

```
## [1] "Model loaded from cache: models/knn_195a16ef06b6ba7503521f913956918f.rds"
```

```r
gbm_model_candles_fg_chain_ta_lag_1 <- train_with_cache(formula_candles_fg_chain_ta_lag_1,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_950928ee0828cd0c9299529274a8cd55.rds"
```

```r
gbm_model_candles_fg_chain_ta_lag_3 <- train_with_cache(formula_candles_fg_chain_ta_lag_3,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_1197d5a7e07e5603640c48925a3ef5cd.rds"
```

```r
gbm_model_candles_fg_chain_ta_lag_5 <- train_with_cache(formula_candles_fg_chain_ta_lag_5,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_e8b4ec46cd84c79cd5cc650553b52bd8.rds"
```

```r
gbm_model_candles_fg_chain_ta_lag_7 <- train_with_cache(formula_candles_fg_chain_ta_lag_7,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_20b24c6f306d90c7bb9e8dc3389712a1.rds"
```

```r
gbm_model_candles_fg_chain_ta_lag_15 <- train_with_cache(formula_candles_fg_chain_ta_lag_15,
    train_set, "gbm")
```

```
## [1] "Model loaded from cache: models/gbm_195a16ef06b6ba7503521f913956918f.rds"
```

```r
results_candles_fg_chain_ta <- evaluate_models("candles_fg_chain_ta",
    test_set)
results_candles_fg_chain_ta
```

```
##                                     model model_type lag  accuracy rank
## 2     glm_model_candles_fg_chain_ta_lag_3        glm   3 0.5383195    1
## 24    gbm_model_candles_fg_chain_ta_lag_7        gbm   7 0.5360111    2
## 3     glm_model_candles_fg_chain_ta_lag_5        glm   5 0.5346260    3
## 22    gbm_model_candles_fg_chain_ta_lag_3        gbm   3 0.5337027    4
## 23    gbm_model_candles_fg_chain_ta_lag_5        gbm   5 0.5327793    5
## 5    glm_model_candles_fg_chain_ta_lag_15        glm  15 0.5300092    6
## 11  rpart_model_candles_fg_chain_ta_lag_1      rpart   1 0.5272392    7
## 12  rpart_model_candles_fg_chain_ta_lag_3      rpart   3 0.5272392    8
## 13  rpart_model_candles_fg_chain_ta_lag_5      rpart   5 0.5272392    9
## 14  rpart_model_candles_fg_chain_ta_lag_7      rpart   7 0.5272392   10
## 15 rpart_model_candles_fg_chain_ta_lag_15      rpart  15 0.5272392   11
## 21    gbm_model_candles_fg_chain_ta_lag_1        gbm   1 0.5267775   12
## 25   gbm_model_candles_fg_chain_ta_lag_15        gbm  15 0.5263158   13
## 10    rf_model_candles_fg_chain_ta_lag_15         rf  15 0.5253924   14
## 1     glm_model_candles_fg_chain_ta_lag_1        glm   1 0.5244691   15
## 7      rf_model_candles_fg_chain_ta_lag_3         rf   3 0.5244691   16
## 8      rf_model_candles_fg_chain_ta_lag_5         rf   5 0.5235457   17
## 4     glm_model_candles_fg_chain_ta_lag_7        glm   7 0.5226223   18
## 6      rf_model_candles_fg_chain_ta_lag_1         rf   1 0.5203139   19
## 16    knn_model_candles_fg_chain_ta_lag_1        knn   1 0.5161588   20
## 18    knn_model_candles_fg_chain_ta_lag_5        knn   5 0.5073869   21
## 9      rf_model_candles_fg_chain_ta_lag_7         rf   7 0.5004617   22
## 19    knn_model_candles_fg_chain_ta_lag_7        knn   7 0.5004617   23
## 17    knn_model_candles_fg_chain_ta_lag_3        knn   3 0.4990766   24
## 20   knn_model_candles_fg_chain_ta_lag_15        knn  15 0.4852262   25
```

```r
summary_stats_candles_fg_chain_ta <- aggregate(accuracy ~ model_type,
    data = results_candles_fg_chain_ta, FUN = function(x) c(mean = mean(x),
        sd = sd(x), max = max(x)))

summary_stats_candles_fg_chain_ta <- data.frame(model_type = summary_stats_candles_fg_chain_ta$model_typ
    mean_accuracy = summary_stats_candles_fg_chain_ta$accuracy[,
        "mean"], sd_accuracy = summary_stats_candles_fg_chain_ta$accuracy[,
        "sd"], max_accuracy = summary_stats_candles_fg_chain_ta$accuracy[,
        "max"])

knitr::kable(summary_stats_candles_fg_chain_ta, format = "simple",
    caption = "Summary statistics for candles features, fear and greed index, chain data and technical a
    digits = 4, col.names = c("Model Type", "Mean Accuracy",
        "SD Accuracy", "Max Accuracy"))
```

Table 16: Summary statistics for candles features, fear and greed index, chain data and technical analysis indicators

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
gbm	0.5285	0.0081	0.5416

Model Type	Mean Accuracy	SD Accuracy	Max Accuracy
glm	0.5329	0.0097	0.5411
knn	0.5133	0.0070	0.5231
rf	0.5152	0.0095	0.5282
rpart	0.5295	0.0103	0.5342

3.3 Models comparison

```
feature_sets <- c("OHLC", "candles", "candles_fg", "candles_fg_chain",
    "candles_fg_chain_ta")

# Function to get top models across all feature sets
get_top_models <- function(test_set, n = 10) {
    all_results <- data.frame()

    for (feature_set in feature_sets) {
        results <- evaluate_models(feature_set, test_set)
        all_results <- rbind(all_results, results)
    }

    # Sort by accuracy and get top n
    all_results <- all_results[order(-all_results$accuracy),
        ]
    head(all_results, n)
}

get_top_models(test_set)
```

```
##                                  model model_type lag  accuracy rank
## 41              glm_model_candles_lag_7        glm   7 0.5433980    1
## 242          gbm_model_candles_fg_lag_7        gbm   7 0.5415512    1
## 1                  glm_model_OHLC_lag_1        glm   1 0.5410896    1
## 32           glm_model_candles_fg_lag_5        glm   5 0.5410896    2
## 131           rpart_model_candles_lag_5      rpart   5 0.5401662    2
## 27           glm_model_candles_fg_lag_3        glm   3 0.5401662    3
## 26              glm_model_candles_lag_3        glm   3 0.5397045    3
## 3                  glm_model_OHLC_lag_5        glm   5 0.5392428    2
## 42           glm_model_candles_fg_lag_7        glm   7 0.5383195    4
## 29  glm_model_candles_fg_chain_ta_lag_3        glm   3 0.5383195    1
```

```
feature_set_summary <- data.frame()
for (feature_set in feature_sets) {
    results <- evaluate_models(feature_set, test_set)
    avg_accuracy <- mean(results$accuracy)
    sd_accuracy <- sd(results$accuracy)
    feature_set_summary <- rbind(feature_set_summary, data.frame(feature_set = feature_set,
        avg_accuracy = avg_accuracy, sd_accuracy = sd_accuracy))
}
feature_set_summary <- feature_set_summary[order(-feature_set_summary$avg_accuracy),
    ]
feature_set_summary
```

```
##           feature_set avg_accuracy sd_accuracy
## 2             candles    0.5272576  0.01015564
## 3          candles_fg    0.5240813  0.01142421
## 5 candles_fg_chain_ta    0.5214958  0.01303384
## 4    candles_fg_chain    0.5181717  0.01350556
## 1                OHLC    0.5082548  0.01909675
```

As expected we can see that OHLC doesn't perform well, suprisingly we can see that just the candles itself performs the best.

We will use the following three models to fine tune, first the number of lags, then using the fine tuning parameters.

- glm_model_candles_lag_7
- gbm_model_candles_fg_lag_7
- rpart_model_candles_lag_5