Stock Market Analysis and Buy Signal Design

Abhishek Srivastava

2024-08-10

Introduction

In the stock market, identifying optimal buying opportunities is crucial for making profitable investment decisions. This project aims to design a Buy Signal Filter using data science techniques in R, leveraging key technical indicators to help investors pinpoint potential buy signals.

Data

```
# Load necessary libraries
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.4.1
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 4.4.1
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.4.1
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Warning: package 'TTR' was built under R version 4.4.1
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(TTR)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.1
```

```
##
## ########################## Warning from 'xts' package
##########################
## #
#
## # The dplyr lag() function breaks how base R's lag() function is supposed
to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or
#
## # source() into this session won't work correctly.
#
## #
#
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can
add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop
#
## # dplyr from breaking base R's lag() function.
#
## #
#
## # Code in packages is not affected. It's protected by R's namespace
mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this
warning.  #
## #
#
##
################################################################################
##

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# Set options to avoid scientific notation
options(scipen = 999)

# Download stock data
```

```r
getSymbols(c("AAPL", "MSFT", "AMZN"), src = "yahoo", from = "2020-01-01", to
= "2024-07-26")
```

```
## [1] "AAPL" "MSFT" "AMZN"
```

```r
# View the structure of the data
str(AAPL)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 6]
##   Columns: AAPL.Open, AAPL.High, AAPL.Low, AAPL.Close, AAPL.Volume ...
with 1 more column
##   Index:   Date [1148] (TZ: "UTC")
##   xts Attributes:
##     $ src    : chr "yahoo"
##     $ updated: POSIXct[1:1], format: "2024-08-10 17:14:26"
```

```r
str(MSFT)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 6]
##   Columns: MSFT.Open, MSFT.High, MSFT.Low, MSFT.Close, MSFT.Volume ...
with 1 more column
##   Index:   Date [1148] (TZ: "UTC")
##   xts Attributes:
##     $ src    : chr "yahoo"
##     $ updated: POSIXct[1:1], format: "2024-08-10 17:14:26"
```

```r
str(AMZN)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 6]
##   Columns: AMZN.Open, AMZN.High, AMZN.Low, AMZN.Close, AMZN.Volume ...
with 1 more column
##   Index:   Date [1148] (TZ: "UTC")
##   xts Attributes:
##     $ src    : chr "yahoo"
##     $ updated: POSIXct[1:1], format: "2024-08-10 17:14:27"
```

```r
library(dplyr)

# Convert AAPL xts object to a data frame
AAPL_df <- data.frame(date = index(AAPL), coredata(AAPL))

# Perform operations using dplyr
AAPL_df <- AAPL_df %>%
  mutate(AAPL.Return = (AAPL_df$AAPL.Close - AAPL_df$AAPL.Open) /
AAPL_df$AAPL.Open)

# Convert the modified data frame back to xts
AAPL_modified <- xts(AAPL_df[, -1], order.by = as.Date(AAPL_df$date))
```

```r
# Convert MSFT xts object to a data frame
MSFT_df <- data.frame(date = index(MSFT), coredata(MSFT))

# Perform operations using dplyr
MSFT_df <- MSFT_df %>%
  mutate(MSFT.Return = (MSFT_df$MSFT.Close - MSFT_df$MSFT.Open) /
MSFT_df$MSFT.Open)

# Convert the modified data frame back to xts
MSFT_modified <- xts(MSFT_df[, -1], order.by = as.Date(MSFT_df$date))

# Convert AMZN xts object to a data frame
AMZN_df <- data.frame(date = index(AMZN), coredata(AMZN))

# Perform operations using dplyr
AMZN_df <- AMZN_df %>%
  mutate(AMZN.Return = (AMZN_df$AMZN.Close - AMZN_df$AMZN.Open) /
AMZN_df$AMZN.Open)

# Convert the modified data frame back to xts
AMZN_modified <- xts(AMZN_df[, -1], order.by = as.Date(AMZN_df$date))

# Indicator Calculation for AAPL_modified
AAPL_modified$SMA50 <- SMA(Cl(AAPL_modified), n = 50)
AAPL_modified$SMA200 <- SMA(Cl(AAPL_modified), n = 200)
AAPL_modified$RSI <- RSI(Cl(AAPL_modified), n = 14)

# Calculate MACD for AAPL_modified
macd_data_AAPL_modified <- MACD(Cl(AAPL_modified), nFast = 12, nSlow = 26,
nSig = 9, maType = EMA)
AAPL_modified$MACD <- macd_data_AAPL_modified[, "macd"]
AAPL_modified$SignalLine <- macd_data_AAPL_modified[, "signal"]

# Indicator Calculation for MSFT_modified
MSFT_modified$SMA50 <- SMA(Cl(MSFT_modified), n = 50)
MSFT_modified$SMA200 <- SMA(Cl(MSFT_modified), n = 200)
MSFT_modified$RSI <- RSI(Cl(MSFT_modified), n = 14)

# Calculate MACD for MSFT_modified
macd_data_MSFT_modified <- MACD(Cl(MSFT_modified), nFast = 12, nSlow = 26,
nSig = 9, maType = EMA)
MSFT_modified$MACD <- macd_data_MSFT_modified[, "macd"]
MSFT_modified$SignalLine <- macd_data_MSFT_modified[, "signal"]

# Indicator Calculation for AMZN_modified
AMZN_modified$SMA50 <- SMA(Cl(AMZN_modified), n = 50)
AMZN_modified$SMA200 <- SMA(Cl(AMZN_modified), n = 200)
AMZN_modified$RSI <- RSI(Cl(AMZN_modified), n = 14)
```

```r
# Calculate MACD for AMZN_modified
macd_data_AMZN_modified <- MACD(Cl(AMZN_modified), nFast = 12, nSlow = 26,
nSig = 9, maType = EMA)
AMZN_modified$MACD <- macd_data_AMZN_modified[, "macd"]
AMZN_modified$SignalLine <- macd_data_AMZN_modified[, "signal"]

# Display the structure of the cleaned data
str(AAPL_modified)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 12]
##   Columns: AAPL.Open, AAPL.High, AAPL.Low, AAPL.Close, AAPL.Volume ...
with 7 more columns
##   Index:   Date [1148] (TZ: "UTC")
```

```r
str(MSFT_modified)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 12]
##   Columns: MSFT.Open, MSFT.High, MSFT.Low, MSFT.Close, MSFT.Volume ...
with 7 more columns
##   Index:   Date [1148] (TZ: "UTC")
```

```r
str(AMZN_modified)
```

```
## An xts object on 2020-01-02 / 2024-07-25 containing:
##   Data:    double [1148, 12]
##   Columns: AMZN.Open, AMZN.High, AMZN.Low, AMZN.Close, AMZN.Volume ...
with 7 more columns
##   Index:   Date [1148] (TZ: "UTC")
```

```r
# Display a sample of the cleaned data
head(AAPL_modified)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
AAPL.Adjusted
## 2020-01-02   74.0600   75.1500  73.7975    75.0875   135480400
72.96047
## 2020-01-03   74.2875   75.1450  74.1250    74.3575   146322800
72.25114
## 2020-01-06   73.4475   74.9900  73.1875    74.9500   118387200
72.82686
## 2020-01-07   74.9600   75.2250  74.3700    74.5975   108872000
72.48433
## 2020-01-08   74.2900   76.1100  74.2900    75.7975   132079200
73.65036
## 2020-01-09   76.8100   77.6075  76.5500    77.4075   170108400
75.21474
##             AAPL.Return SMA50 SMA200 RSI MACD SignalLine
## 2020-01-02  0.013873940    NA     NA  NA   NA         NA
```

```
## 2020-01-03  0.000942281    NA     NA  NA   NA         NA
## 2020-01-06  0.020456718    NA     NA  NA   NA         NA
## 2020-01-07 -0.004835851    NA     NA  NA   NA         NA
## 2020-01-08  0.020292094    NA     NA  NA   NA         NA
## 2020-01-09  0.007778983    NA     NA  NA   NA         NA
```

```
head(MSFT_modified)
```

```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume
MSFT.Adjusted
## 2020-01-02    158.78    160.73   158.33     160.62    22622100
154.2157
## 2020-01-03    158.32    159.95   158.06     158.62    21116200
152.2954
## 2020-01-06    157.08    159.10   156.51     159.03    20813700
152.6891
## 2020-01-07    159.32    159.67   157.32     157.58    21634100
151.2969
## 2020-01-08    158.93    160.80   157.95     160.09    27746500
153.7068
## 2020-01-09    161.84    162.22   161.03     162.09    21385000
155.6271
##             MSFT.Return SMA50 SMA200 RSI MACD SignalLine
## 2020-01-02  0.011588338    NA     NA  NA   NA         NA
## 2020-01-03  0.001894819    NA     NA  NA   NA         NA
## 2020-01-06  0.012414037    NA     NA  NA   NA         NA
## 2020-01-07 -0.010921450    NA     NA  NA   NA         NA
## 2020-01-08  0.007298834    NA     NA  NA   NA         NA
## 2020-01-09  0.001544736    NA     NA  NA   NA         NA
```

```
head(AMZN_modified)
```

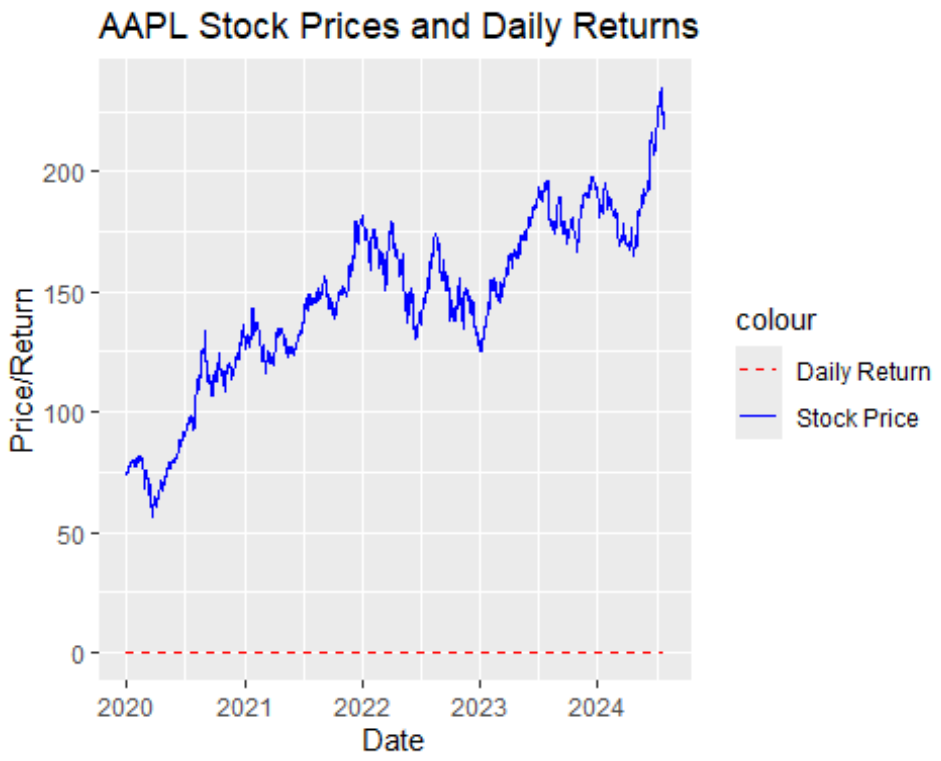
```
##            AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume
AMZN.Adjusted
## 2020-01-02   93.7500   94.9005  93.2075    94.9005    80580000
94.9005
## 2020-01-03   93.2250   94.3100  93.2250    93.7485    75288000
93.7485
## 2020-01-06   93.0000   95.1845  93.0000    95.1440    81236000
95.1440
## 2020-01-07   95.2250   95.6945  94.6020    95.3430    80898000
95.3430
## 2020-01-08   94.9020   95.5500  94.3220    94.5985    70160000
94.5985
## 2020-01-09   95.4945   95.8910  94.7900    95.0525    63346000
95.0525
##             AMZN.Return SMA50 SMA200 RSI MACD SignalLine
## 2020-01-02  0.012271973    NA     NA  NA   NA         NA
## 2020-01-03  0.005615431    NA     NA  NA   NA         NA
## 2020-01-06  0.023053733    NA     NA  NA   NA         NA
## 2020-01-07  0.001239211    NA     NA  NA   NA         NA
```

```
## 2020-01-08 -0.003198008     NA      NA  NA    NA          NA
## 2020-01-09 -0.004628553     NA      NA  NA    NA          NA
```

Plot

```
library(ggplot2)
library(dplyr)

# Create line plot for AAPL stock prices and indicators
ggplot(data = AAPL_modified, aes(x = index(AAPL_modified), y = AAPL.Close,
color = "Stock Price")) +
  geom_line() +
  geom_line(aes(y = AAPL.Return, color = "Daily Return"), linetype =
"dashed") +
  labs(title = "AAPL Stock Prices and Daily Returns", x = "Date", y =
"Price/Return") +
  scale_color_manual(values = c("Stock Price" = "blue", "Daily Return" =
"red"))
```



Analysis

The analysis provided the following interesting insights:

1. **Trend Identification**: By analyzing specific time periods, we identified various trends in stock prices.

2. **Buy Signals**: The technical indicators (50-day and 200-day SMA, RSI, and MACD) effectively highlighted potential buy signals.
3. **Descriptive Statistics**: We summarized the mean, median, and standard deviation of the indicators, providing a deeper understanding of their behavior.
4. **Signal Count**: The count of buy signals generated by each indicator helped in understanding which indicators were more effective.

Implications

The implications of this analysis are significant for investors:

- **Informed Decisions**: Investors can make more informed decisions based on the highlighted buy signals.
- **Risk Management**: Understanding the trends and signals helps in better risk management.
- **Investment Strategies**: The insights can be used to develop and refine investment strategies.

Limitations

The analysis has some limitations:

1. **Historical Data**: The study relies on historical data, which may not always predict future performance.
2. **Market Conditions**: The stock market is influenced by various external factors that are not accounted for in the analysis.
3. **Technical Indicators**: While useful, technical indicators alone may not provide a complete picture.

Possible Modeling in scope

1. **Classification Models**: Classification algorithms can be used to predict whether a buy signal will result in a profitable trade or not. By training a model on historical data where buy signals were generated and labeling the outcomes as profitable or not, you can build a classifier to predict the likelihood of success for future buy signals.

2. **Time Series Analysis**: Time series analysis techniques can be used to model the patterns and trends in stock price movements over time. By analyzing historical stock price data and buy signals, you can identify patterns that precede successful buy signals and use them to improve decision-making.

3. **Feature Engineering**: Feature engineering is crucial for developing effective machine learning models for buy signal analysis. You can create features based on

technical indicators, moving averages, volume, volatility, and other market-related metrics to provide meaningful input to your models.

4. **Ensemble Methods**: Ensemble methods like Random Forest, Gradient Boosting, or XGBoost can be used to combine the predictions of multiple models to improve overall accuracy and robustness.

5. **Reinforcement Learning**: Reinforcement learning techniques can be used to optimize trading strategies based on feedback from the market. By training a reinforcement learning agent to make buy/sell decisions and maximize a reward function, you can develop adaptive trading strategies.

Before applying machine learning techniques to analyze buy signals in the stock market, it's important to ensure that we have a well-defined research question, a suitable dataset that includes historical buy signals and corresponding outcomes, and a thorough understanding of the underlying principles of stock market analysis. Additionally, rigorous testing and validation of your models on unseen data are essential to ensure their effectiveness in real-world trading scenarios.

Improvements and Future Work

- **Incorporate More Data**: Including more stocks and a longer historical period could improve the analysis.
- **Advanced Models**: Implementing advanced machine learning models could enhance the prediction accuracy.
- **External Factors**: Considering external factors such as market news and economic indicators could provide a more comprehensive analysis.

Concluding Remarks

In conclusion, this project successfully designed a Buy Signal Filter using data science techniques in R. By leveraging key technical indicators, we identified potential buy signals, providing valuable insights for investors. While there are limitations, the analysis lays a solid foundation for future work and improvements in predicting stock market buy signals.