Arima Model for Stock Prediction

Rawi Baransy

2023-10-22

```
#load all packages
library(BatchGetSymbols)
```

```
## Loading required package: rvest
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

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

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

```
##
```

```
library(quantmod)
```

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

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

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

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

```
##
## ######################### 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       #
```

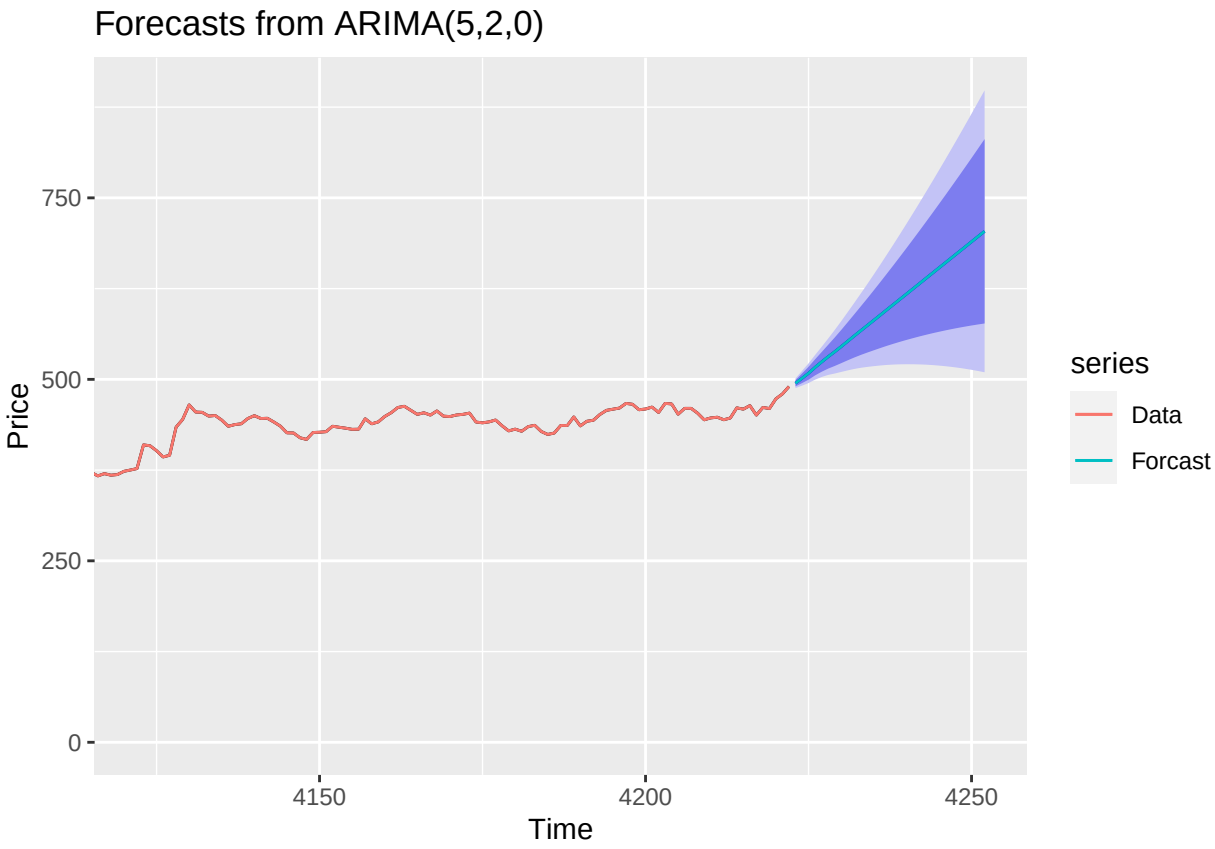
```
## # source() into this session won't work correctly.                    #
## #                                                                     #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to 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: 'xts'

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

## Loading required package: TTR

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

```r
library(forecast)
library(ggplot2)

#load S&P500 stock symbols
sp500 <- GetSP500Stocks()

name = sp500$Tickers
name[which(name == "BRK.B")] = "BRK-B"
name[which(name == "BF.B")] = "BF-B"


#Create vector to keep track of best stocks
arr = 0

#create a loop to fit arima model on all stocks
for(i in 1:length(name)){

  stock = as.data.frame(getSymbols(name[i], to = (Sys.Date()-10), env = NULL))

  #set prediction interval
  term = 30

  #stock closing price
  Price = stock[,4]

  #fit model and get forecast
  fit_model = auto.arima(Price, seasonal = TRUE)

  fcast= forecast(fit_model, h = term)
```

```
  #find which stocks are optimal
    if(fcast$lower[term,1]  >= Price[length(Price)]){
      arr[i] = 1
    }else{
      arr[i] = 0
    }
  #track how long program is running
    # print(i)
}

#find optimal stock index
name[which(arr == 1)]
```

```
## [1] "CDNS" "CTAS" "CPRT" "LLY"  "MSI"  "ODFL" "SNPS"
```

```
#plot results
if(length(which(arr == 1) >0)){
  for(j in seq_along(which(arr == 1)))

  stock = as.data.frame(getSymbols(name[which(arr == 1)][j], to = (Sys.Date()-10), env = NULL))
  term = 30

  Price = stock[1:(nrow(stock)),4]

  fit_model = auto.arima(Price, seasonal = TRUE)

  fcast= forecast(fit_model, h = term)

  autoplot(fcast)+
    autolayer(ts(stock[,4]), series="Data")+
    autolayer(fcast$mean, series="Forcast")+
    coord_cartesian(xlim = c(nrow(stock)-100,(nrow(stock)+term)), ylim = c(0,Price[(nrow(stock)+50)]))

}
```


Forecasts from ARIMA(5,2,0)
Price
750
500
250
0
4150
4200
4250
Time
series
Data
Forcast