Stock Prediction System

Riddhanand Mohapatra

21/12/2020

Predictive Modeling

Predictive modeling is an area concerned with forecasting probabilities and trends.It uses previous known results to create, model and validate a statistical model that can be used to make future predictions.

This article is a good example of an application of predictive modeling. This article talks uses predictive modeling to create a model that attempts to predict the next day price change of a stock using various technical indicators. A brief explanation of technical indicators that are used as predictors will also be mentioned in this article.

Technical Indicators

In technical analysis, a technical indicator is a mathematical based signal produced by historic price, volume etc of a security that technical analysts use to forecast financial market direction. The technical indicators used in this model are explained below.

Force Index

Force Index measures the amount of power used to move the price of an asset. It uses price and volume to determine the amount of strengh behind a price move. The index is an oscillator that fluctuates between positive and negative territory.

A rising force index above zero can indicate rising prices and a falling force index below zero can help confirm a breakout in price. The formula for the force index is given below -

$$F.I = (CCP - PCP) * (V)$$

where, CCP is the closed current price, PCP is the prior closed price and V is the volume

Williams Percentage Range

Williams percentage range also known as Williams % R is a type of momentum indicator that moves between zero and hundred and measures over bought or over sold levels.A reading above -20 is overbought and a reading below -80 is oversold. The formula is given by -

$$W\%5 = \frac{Highest\ High - Close}{Highest\ High - Lowest\ Low}$$

Relative Strength Indicator

Relative strength indicator (RSI) is another type of momentum indicator that measures the magnitude of a previous changes to evaluate over bought or over sold situations.This is displayed as an oscillator and can have a reading from 0 to 100. The RSI provides traders signals about bullish or bearish price momentum.An asset is usually considered overbought when the RSI is above 70% and oversold when it is below 30%. The formula is given by -

$$RSI = 100 - [\frac{100}{1 - \frac{average\ gain}{average\ loss}}]$$

Price Rate of Change

The Price rate of change (ROC) is also a momentum based technical indicator that measures the percentage of change in price between the current price and the price a certain number of periods ago. The formula is of ROC is given by -

$$ROC = \frac{(CP_p - CP_n)}{CP_n}$$

where, CP_p is the current closing price and CP_n is the closing price before 'n' periods.

Average True Range

Average true range (ATR) measures market volatility by decomposing the entire range of asset price for that period. Simply put, a stock experiencing a high level of volatility has a higher ATR, and a low volatility stock has a lower ATR.

Creating the dataset

First a dataset is obtained from Yahoo Finance. This is the Amazon stock data that comprises of - Date, Open, High, Low, Close and Volume. Using these, values of technical indicators are obtained. The formulas used to obtain these are mentioned in the previous section. A code is also written to dtermine the price change. The price change is classified as class - Up or Down.

```
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")
if(!require(data.table)) install.packages("data.table", repos =
"http://cran.us.r-project.org")
if(!require(quantmod)) install.packages("quantmod", repos =
"http://cran.us.r-project.org")
if(!require(pROC)) install.packages("pROC", repos = "http://cran.us.r-
project.org")
if(!require(TTR)) install.packages("TTR", repos = "http://cran.us.r-
```

```
project.org")

library(tidyverse)
library(dplyr)
library(quantmod)
library(TTR)
library(caret)
library(corrplot)
library(pROC)
library(ggplot2)

#Scrape data from yahoo finance (STOCK & INDEX)
AM <- getSymbols("AMZN",from="2015-08-01",to="2020-12-05")

#Graph capturing this years amazon's price chart (also shows bollinger bands,
volume traded and MACD)
Graph <- AMZN%>%chartSeries(TA='addBBands();addVo();addMACD()',subset='2020')
```

Amazon Stock Prices

```
#Determine price change for stock and classify them as up or down
price <- AMZN$AMZN.Close - AMZN$AMZN.Open
class <- ifelse(price > 0, 'UP','DOWN')

#add serial number to data
AMZN$id <-  1:nrow(AMZN)
AMZN <- AMZN[, c(7, 1, 2, 3, 4,5,6)]
```

```
#Computing various technical indexes
#F.I indicator
forceindex <- (AMZN$AMZN.Close - AMZN$AMZN.Open) * AMZN$AMZN.Vol


# Buy & Sell signal Indicators (Williams %R and ROC)
Will5 <- WPR(AMZN[,c("AMZN.High","AMZN.Low","AMZN.Close")], n = 5)
Will10 <- WPR(AMZN[,c("AMZN.High","AMZN.Low","AMZN.Close")], n = 10)



RSI5  <- RSI(AMZN$AMZN.Close, n = 5,maType="WMA")
RSI10 <- RSI(AMZN$AMZN.Close, n = 10,maType="WMA")

#Other momentum indicators
ROC5 <- ROC(AMZN$AMZN.Close, n = 5,type ="discrete")*100
ROC10 <- ROC(AMZN$AMZN.Close, n = 10,type ="discrete")*100

# Volatility signal Indicator (ATR)
ATR5 = ATR(AMZN[,c("AMZN.High","AMZN.Low","AMZN.Close")], n = 5,
maType="WMA")[,2]
ATR10 = ATR(AMZN[,c("AMZN.High","AMZN.Low","AMZN.Close")], n = 10,
maType="WMA")[,2]

#combing all
df =
data.frame(class,forceindex,Will5,Will10,RSI5,RSI10,ROC5,ROC10,ATR5,ATR10)
df <- na.omit(df)
ddf <- as.tibble(df)
```

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
ddf <- ddf %>% rename(Class = AMZN.Close, Forceindex = AMZN.Close.1, WillR5 =
AMZN.Close.2, willR10 = AMZN.Close.3, RSI_5 = rsi, RSI_10 = rsi.1, ROC_5 =
AMZN.Close.4, ROC_10 = AMZN.Close.5, ATR5 = atr, ATR10 = atr.1)
```

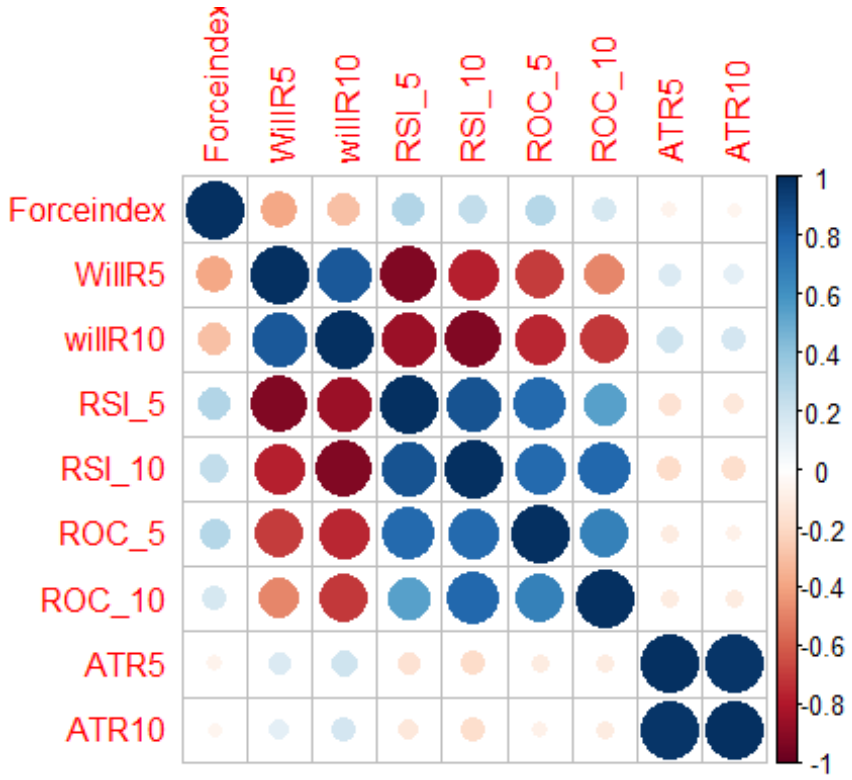
Understanding and visualising the data

This is important as understanding the data helps in choosing the right model and improving the accuracy. This can be done by using the summary function. The summary functions provides information such as the mean, median, standard deviation etc. A graph showing the correlation of all the predictors us also plotted below. This can give us an idea of importance of each predictor.

```
#Visualizing data by viewing correlation
summary(ddf)
```

```
##     Class              Forceindex             WillR5          willR10      
##  Length:1337        Min.   :-1.679e+09   Min.   :0.0000   Min.   :0.0000  
##  Class :character   1st Qu.:-3.580e+07   1st Qu.:0.1371   1st Qu.:0.1261  
##  Mode  :character   Median : 0.000e+00   Median :0.3609   Median :0.3265  
##                     Mean   :-5.699e+06   Mean   :0.4115   Mean   :0.3882  
##                     3rd Qu.: 3.504e+07   3rd Qu.:0.6768   3rd Qu.:0.6217  
##                     Max.   : 1.494e+09   Max.   :1.0000   Max.   :1.0000  
##      RSI_5            RSI_10           ROC_5              ROC_10         
##  Min.   :  0.00   Min.   : 1.306   Min.   :-15.5961   Min.   :-20.702  
##  1st Qu.: 35.01   1st Qu.:41.895   1st Qu.: -1.2886   1st Qu.: -1.443  
##  Median : 60.35   Median :57.886   Median :  0.9066   Median :  1.568  
##  Mean   : 57.90   Mean   :57.716   Mean   :  0.7599   Mean   :  1.507  
##  3rd Qu.: 83.19   3rd Qu.:74.453   3rd Qu.:  2.9836   3rd Qu.:  4.642  
##  Max.   :100.00   Max.   :99.692   Max.   : 17.8752   Max.   : 26.235  
##       ATR5             ATR10       
##  Min.   :  4.962   Min.   :  5.252  
##  1st Qu.: 14.032   1st Qu.: 14.226  
##  Median : 25.201   Median : 26.043  
##  Mean   : 36.511   Mean   : 36.441  
##  3rd Qu.: 46.739   3rd Qu.: 46.998  
##  Max.   :185.597   Max.   :159.663
```

```
correlation <- cor(ddf[,c(2:10)])
corrplot(correlation)
```



Choosing and training different algorithms

Since the stock prediction aims at predicting the price change, that is, predict a price up or a price down, it is a classification problem.Therefore, we will use algorithms such K-nearest neighbors (KNN) and other discriminative approaches such a Linear discriminant analysis (LDA) and Quadratic discriminant analysis (QDA).

We will use the CreatePartition function and train function from the caret package to create a test set and train set and fit predictive models.

```
#create train and test data
set.seed(5)
test_index <- createDataPartition(ddf$Class,times=1,p=0.5,list=FALSE)
test <- ddf[test_index,]

## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

train <- ddf[-test_index,]


#Trying Classification Algorithms
#KNN
set.seed(5)
train_knn_cv <- train(Class ~. , method = "knn",
                      data = train,  preProc=c("range"))

KNN <- confusionMatrix(data = predict(train_knn_cv, test),
                reference = as.factor(test$Class))$overall["Accuracy"]

# QDA and LDA
set.seed(5)
train_lda<- train(Class~., data=train, method="lda")

lda <- confusionMatrix(data = predict(train_lda, test),
                reference = as.factor(test$Class))$overall["Accuracy"]

train_qda <- train(Class~., data=train, method="qda")

qda <- confusionMatrix(data = predict(train_qda, test),
                reference = as.factor(test$Class))$overall["Accuracy"]
```

Evaluating the models

The accuracy with which the model predicts the outcome is taken into account while evaluating the models. We trained 3 models - KNN, LDA and QDA. These models are compared and the best one is picked for further tuning.

```
#create train and test data

results <- c(KNN = KNN, LDA = lda, QDA = qda)
results

## KNN.Accuracy LDA.Accuracy QDA.Accuracy
##    0.7563528    0.7757848    0.7503737
```

Tuning Parameters

Since KNN is our best performing, we will tune the KNN algorithm over values ranging from 0 to 100. The codes for this are given below.

```
set.seed(5)
grid = expand.grid(k=seq(1,100,1))
train_knn_cv <- train(Class ~. , method = "knn", tuneGrid = grid,
                      data = train,  preProc=c("range"))
KNN <- confusionMatrix(data = predict(train_knn_cv, test),
                       reference = as.factor(test$Class))$overall["Accuracy"]
KNN

##  Accuracy
## 0.7443946
```

It can be seen that the tuning process had no effect on the KNN algorithm. This can be done for the other models as well but is beyond the scope of this article.

Results

As we can see, the KNN model predicts the price change on the test set with an accuracy of 75%.

Conclusion and further work

This article starts of with a brief discussion of various well known technical indicators followed by codes that are used to obtain them. Then the correlation of the indicators with other indicators is studied. Three different algorithms that are specially geared to handle classification problems are used to train the models. Accuracy of each model is then evaluated and the best model is used for further tuning.

To improve this model, other important technical indicators can be used in addition to the ones that have been used here. for example, this model has left out a very important indicator - The volatility of the market index indicator. This can be calculated from the data of the stock market index such as the NASDAQ index.

Furthermore, the performance of this algorithm can be determined for other asset classes as well.

Since this is a classification problem, decision trees method can also be used to train models.

Bibliography

1.) investopedia.com

2.) Introduction to Data Science: Data Analysis and Prediction Algorithms with R by Rafael Irizzary