Machine Learning Analysis applied to Investment strategies

Université Paris 1 Panthéon-Sorbonne

M2 IRFA Engineering of Financial Mathematics

X5I22919 Data Science Software

Module leader: Prof. Bertrand K. Hassani

Antonio Malatesta* Karan Soneja† Michele D'Ambrosio‡

Contents

*E-mail:antonio.malatesta@outlook.com
†E-mail:sonejakaran@gmail.com
‡E-mail:mi.dambrosio105@gmail.com

1 Introduction

The present report shows an implementation in R [1] of three different Machine learning approaches to exploit historical stock price data for making predictions on the stocks' future price levels. We aim at understanding whether it is possible to obtain profitable trading indications leading to a higher return than a passive strategy. The remaining of the report is structured as follows. In Section 3, trading operations are performed based on technical analysis indicators. In section 4, we use instead neural network analysis to develop an investment strategy, while in Section 5 a Random forest algorithm is implemented.

2 Data

The data used in this report consists of the stock price of some of the 10 biggest publicly quoted French companies, as of September 2019 [2].

Table 1: Top ten firms in CAC40 by market cap, September 2019

Company	MNEMO	Sector	Weight%
TOTAL	FP	Oil and Gas	9,54
LVMH	MC	Personal and Household Goods	7,97
SANOFI	SAN	Health Care	7,54
AIRBUS	AIR	Industrial Goods and Services	5,47
L'OREAL	OR	Personal and Household Goods	5,10
AIR LIQUIDE	AI	Chemicals	4,40
DANONE	BN	Food and Beverage	4,14
VINCI	DG	Construction and Materials	3,97
BNP PARIBAS ACT.A	BNP	Banks	3,95
SAFRAN	SAF	Industrial Goods and Services	3,72

With the following command, we can download the daily open, high, low, close and adjusted prices, as well as the volume, of the stocks we consider in our analysis, from 28/12/2012 to 31/12/2018.

```
quantmod::getSymbols("COMPANY CODE", src="yahoo",
                     from="2012-12-28", to="2018-12-31")
```

In some sections, we add economic and financial variables to test whether they can increase the predictive power of the analysis. The following table offers a detailed list of the variables considered.

Table 2: Additional economical variables

Economic Factors	Source
Daily interest rate	sdw.ecb.europa.eu
Daily Exchange rate (EUR/USD)	exchangerates.org
Political Stability (yearly)	databank.worldbank.org
CPI Infaltion (monthly)	www.inflation.eu

[1] https://cran.r-project.org/

[2] The full list here: https://live.euronext.com/en/product/indices/FR0003500008-XPAR/market-information

Our hypothesis is that factors like inflation and interest rate might have a big impact on companies' stock prices on a day-by-day basis. On the other hand, we consider financial statement data that change only on a yearly basis, checking whether black-box models can find connections between the prices and the variables over a long-term horizon.

Table 3: Additional financial statement variables

Yearly Financial Statements Variables	Source
Dividend	
Revenue	
Net income	dividenmax.com
Basic earnings per share	
Diluted earnings per share	airbus.com
Total assets	
Intangible assets	airliquide.com
PPE	
Cash	danone.com
Total equity	
Non current liability	vinci.com
Cash flows from operating activities	
Net cash flows from investing activities	invest.bnpparibas.com
Net cash flows from financing activities	
Net cash and cash equivalents at end of period	

As a last disclaimer, the code shown in the report has only an explanatory purpose. We further attach the complete code to reproduce the analysis.

3 Technical analysis: implementation of a naive strategy in R

As Brock and Lakonishok [1992] explain, *technical analysis* (TA) is an "attempt to forecast prices by the study of past prices and a few other related summary statistics about security trading" (pag 2). This practice however clashes with the Efficient Market Hypotesis lied out by Fama [1970] , according to which market prices incorporate all available information on the securities at any times.

Despite Fama's paper, which soon got traction among academia, the work by Taylor and Allen [1992] reports that practioners in the world largest financial centres declare making use of Technical analysis, especially for intraday and short-term period trading. This revamped a large stream of research on the topic, to the point where complex optimization algorithms (namely, of the class of evolutionary algorithms) were introduced to get better predictions in terms of global optima search. For example, genetic programming is implemented to avoid the human bias in creating trading rules, letting the algorithms evaluate a series of rules implemented at each search stage, picking the best features and discarding the least performing ones. For a lengthier treatment on the topic, the reader can refer to Neely et al. [1997], or the more recent Mousavi et al. [2014] .

Neftci [1991] gives a proper formal charaterization of the main classes of technical analysis indicators. He claims that "any well-defined technical analysis rule has to pass the test of being a Markov time", otherwise "the procedure would be using future information in order to issue such signals" (pag. 8). Hence, relying heavily on chart analysis can be misleading. A class of signals that appear to be stopping times are those generated by moving average indicators, which are those we use in the present section.

One caveat one should always keep in mind when talking about technical analysis is that its predictive power can be biased upward by its intrinsic self-fulfilling feature: if agents believe that TA can have some predictive power, then following the trends will indeed let the predicted events happen, as all agents will implement similar decisions, given that they base their analysis on similar indicators.

3.1 Technical analysis indicators

We will offer both a formal definition and a visual representation of each TA indicator used. In the following charts, prices are represented by the so-called 'candlesticks'. The rectangular area will show the day open and close, whereas the wicks represent the day high and low. The green filling indicates that the security closed at a higher price than the opening, whereas orange indicates a negative performance on a day-basis. Trading volumes are also shown in the bottom part of the first chart.

The first indicator is the *simple moving average*. It is defined as

$$SMA = \sum_{t=1}^{n} \frac{x_t}{n} \tag{1}$$

where x_t = the t-th price observation, for $t \in \{1, ..., n\}$.

```
chartSeries(AIR.PA,# need '.PA' suffix for Paris CAC40
            type="matchsticks",
            name = "HLOC Candle chart and SMA, Air France",
            theme=chartTheme('white')) # chart setting
```

```
quantmod::addSMA(n=50,on=1,col = 'blue') # to overimpose SMA(50), in blue
quantmod::addSMA(n=200,on=1,col = 'purple') # TA functions from 'TTR'
```

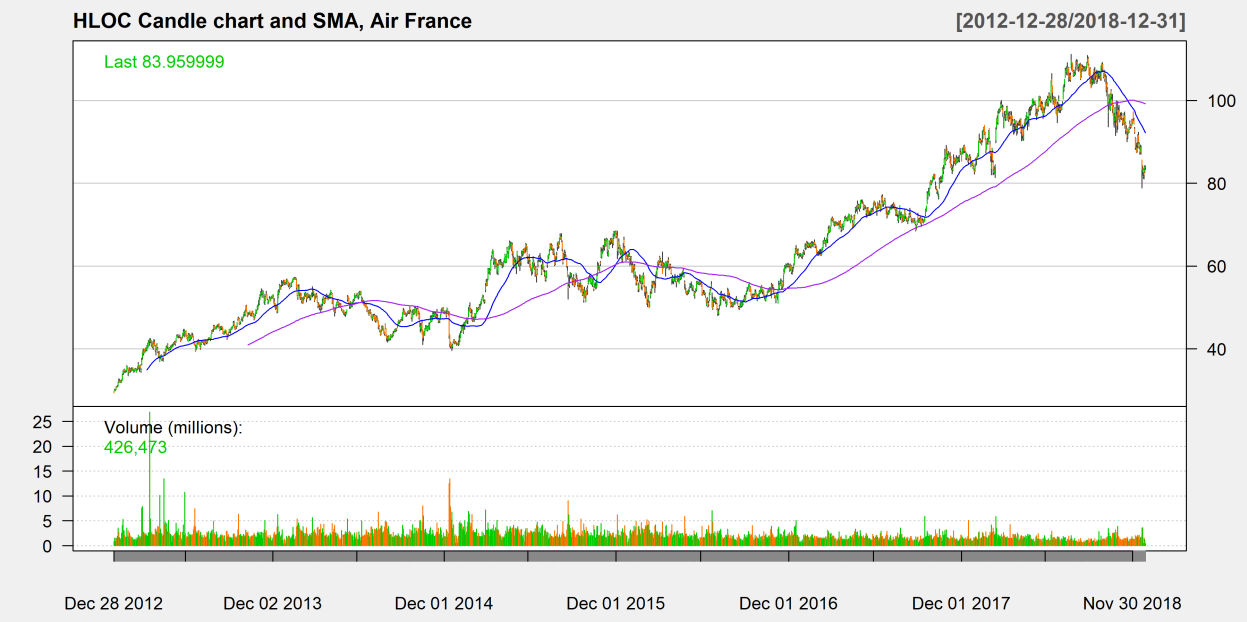

Figure 1: Daily HLOC Chart price, with SMA(50) in blu, and SMA(200) in purple.

The SMA is used to smooth out the price trends from the short-term fluctations. Usually, (at least) two SMA's of different time-lags are used together to generate trading signals. The short-term average crossing from below the long-term average is considered a buy signal, whereas a crossing in the other direction is taken as an indication to sell. One common choice for the lags are the 50- and the 200-days averages. One could also consider shorter time windows, to retain the short-term fluctations in the SMAs.

The second indicator is the **Relative Strength Index**, or RSI . It is computed as

$$RSI = 100 - \left(\frac{100}{(1 + \frac{\Delta_u}{\Delta_d})} \right) \tag{2}$$

where Δ_u = Average of Upward Price Change, and Δ_d = Average of Downward Price Change [3]. (See also Hsu et al. [2016] for an example of different parametrizations.)

The RSI is an oscillator, since its value is in the range [0,100]. It gives indications on the **momentum**, that is, the "the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. [4]" Generally, $RSI > 70$ is taken as an indication for a security to be overbought, whereas $RSI < 30$ has the opposite meaning.

[3] https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/RSI
[4] https://www.investopedia.com/terms/r/rsi.asp

```
quantmod::addRSI(n=14, maType= 'SMA')
# the RSI is shown, albeit not overimposed to the price pattern
```

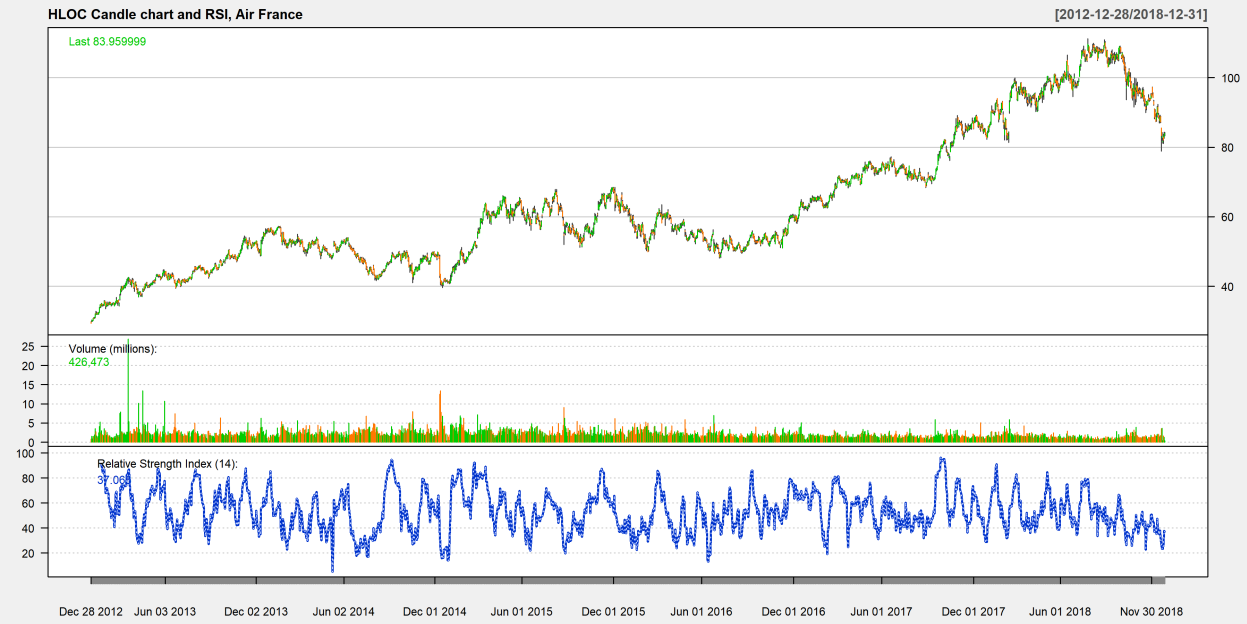

Figure 2: Daily HLOC Chart price with RSI(14)

The third indicator are the **Bollinger Bands** [5]. These are defined by the plot of a simple moving average, as defined below by the μ, along with its standard deviation around it, defining the upper and lower bands. These are defined respectively as

$$B_u = MA(\mu, n) + m \cdot \sigma(\mu, n) \tag{3}$$

and

$$B_d = MA(\mu, n) - m \cdot \sigma(\mu, n) \tag{4}$$

with

$HLC_t := High_t + Low_t + Close_t$;

$\mu = \frac{HLC_t}{3}$;

$\sigma_n = \sqrt{\frac{\sum_{t=1}^{n} (HLC_t - \mu)^2}{n}}$ the standard deviation of HLC over n days;

$n \in \mathbb{N}^*$ the number of days over which the SMA is computed;

$m \in \mathbb{N}^*$ the number of standard deviations used to create the width of the bands.

A common parametrization of this indicator is one with two-standard-deviation-wide bands around a 20-day simple moving average (i.e. $m = 2$,$n = 20$).

[5] https://www.bollingerbands.com/

```
addBBands(n=20, sd=2, maType= "SMA")
```

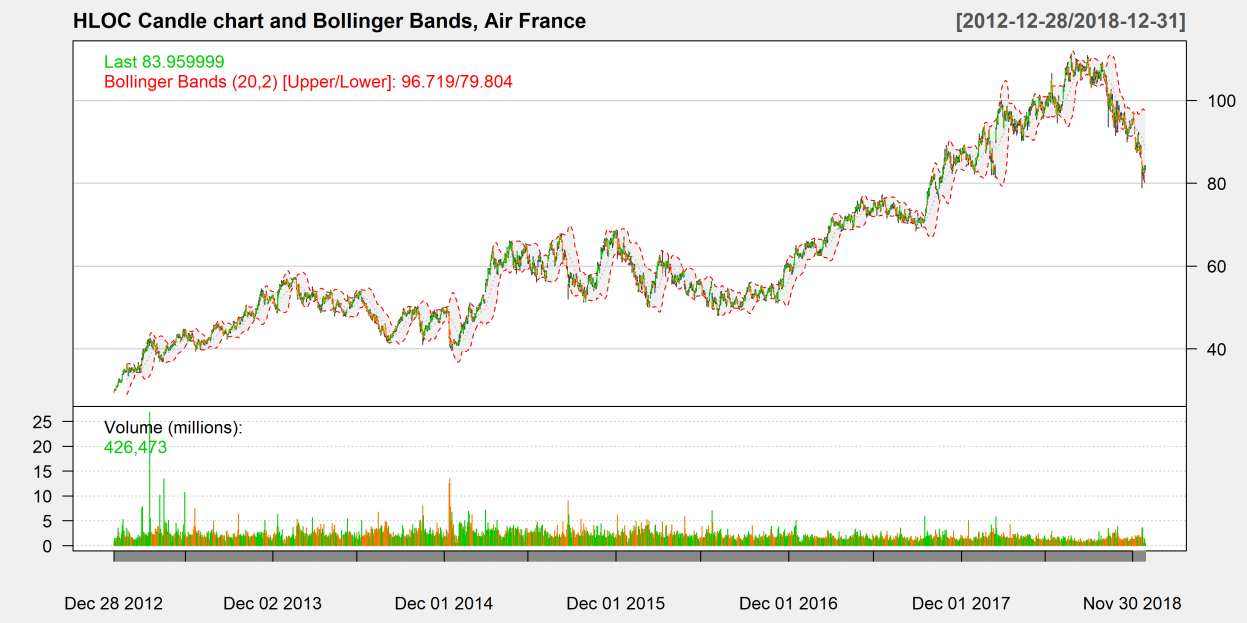

Figure 3: Daily HLOC Chart price with Bollinger Bands (20,2)

3.2 quantmod, blotter and quantstrat packages

One important building block towards the implementation of Technical Analysis in R is the package quantmod [6], which allows the user to download stock prices data from Yahoo!, and to visualize different types of price charts, which are of great support in a first, rough analysis of the evolution of price trends over time. For the sake of offering a visual understanding of TA indicators, these functionalities were already exploited in the previuous paragraph, using the functions `quantmod::getSymbols()` and `quantmod::chartSeries`, along with `quantmod::addSMA`, `quantmod::addRSI`, and `quantmod::addBBands`.

Moreover, R supports a package named `quantstrat`, a "transaction-oriented infrastructure for constructing trading systems and simulation" [7]. However, this package counts several dependencies, and therefore we'll need our machine to be equipped with all these packages, as brilliantly shown in Yu [2019].

```
install.packages("quantmod")
install.packages("FinancialInstrument")
install.packages("PerformanceAnalytics")
install.packages("foreach")
install.packages("devtools")
devtools::install_github("braverock/blotter")
devtools::install_github("braverock/quanstrat")
```

[6] https://github.com/joshuaulrich/quantmod
[7] https://www.rdocumentation.org/packages/quantstrat/versions/0.16.6

3.3 Portfolio management with quantstrat

We build an actively managed portfolio made of the top five firms of the French CAC40 using six years of price data, from 2012-12-28 to 2018-12-31, for a total of 1534 trading days. The choice of the stocks fell simply on those we consider to be liquid enough, so that the chance of observing large sudden jumps in prices is low. In fact, fluctuations would negatively affect the TA predictive perfomance.

Starting with an initial capital of €100000, we will buy or sell 100 stocks of each equity, whenever some conditions on past prices are met. These conditions are later referred to as *signals*, and are linked to the indicators whose mathematical essence is shown in the previuos part. Our main purpose is to create a plausible setup that, upon further improvements, can help an agent to infer some information on future prices movements based on past prices analysis. We will deliberately leave this model at a simplistic and unsophisticated stage, for the sake of outlying its main basic features with clarity. Nevertheless, several improvements would be needed before a proper real-life implementation. We will mention them in the remaining of this chapter.

3.3.1 Preliminary setup

We first retrieve data from Yahoo! to the R global environment:

```
from ="2012-12-28"
to ="2018-12-31"
symbols = c("MC.PA","BN.PA","AIR.PA","BNP.PA","DG.PA" )
# the .PA label is needed to specify the Paris CAC40 stock exchange
quantmod::getSymbols(symbols,
                     from=from, to=to)
```

We then set up the strategy, the portfolio and the account objects:

```
initEq=100000 # capital to invest

strategy.st <- "strat.full.limit" # name the objects
portfolio.st <- "portf.full.limit"
account.st <- "acct.full.limit"

blotter::initPortf(portfolio.st, symbols) #initialize portfolio
blotter::initAcct(account.st, # need specifing also linked portfolio
         portfolios=portfolio.st,
         initEq = initEq) # capital level
quantstrat::initOrders(portfolio.st) # the orders 'container'
quantstrat::strategy(strategy.st, store=TRUE) # save our strategy in
# the global environement
```

3.3.2 Creating indicators

As illustrated earlier, we make use of three simple indicators: simple moving average, relative strength index and Bollinger Bands. The indicators will be linked to the startegy object previoulsy defined. The input dataframe is called *mktdata*, an object that will be created automatically when we validate the strategy.

```
quantstrat::add.indicator(
  strategy.st, name="SMA", # "name" must correspond to an R function
  arguments= list(x= quote(Cl(mktdata)), n=50), # input data are closing prices
  # set input data (mktdata) and indicator parameters through function 'arguments'
  label="sma50") ## unique reference 'mktdata' column name

add.indicator(strategy = strategy.st,
              name = 'RSI', # normally, we'll be using functions
              # from the TTR package
              arguments = list(price = quote(Cl(mktdata)), n=14),
              # among the 'arguments', we need to set the
              # parameter (number of days of the SMA) for the RSI
              label = 'rsi.14')

add.indicator(strategy = strategy.st,
              name= "BBands",
              arguments =
                list(HLC = quote(HLC(mktdata)),n=20, maType= "SMA", sd=2),
              # BBands need High,Low and Closing Prices as an Input
              label = "bb.20.2")
```

3.3.3 Combining indicators to create trading signals

The third step, after setting up the blotter objects and creating the indicators, is creating buy and sell signals out of an appropriate blend of indicators. Below we offer a summary of the signals that will be used here.

Table 4: Summary of signals and related trading executions

Type of Signal	Signal	Execution
Crossover	SMA(50)>SMA(200)	NA
Crossover	SMA(50)<SMA(200)	NA
Threshold	RSI<30	NA
Threshold	RSI>70	NA
Crossover & Threshold	SMA(50)>SMA(200) & RSI<30	BUY
Crossover & Threshold	SMA(50)<SMA(200) & RSI>70	SELL
Crossover	Closing > Upper BBand	NA
Crossover	Closing < Lower BBand	NA
Crossover & Threshold	Closing < Lower BBand & RSI<30	BUY
Crossover & Threshold	Closing > Upper BBand & RSI>70	SELL

Despite having defined ten signals, only four of these will trigger actual execution orders. This is because it looks safer to rely on a mixed signal - i.e. one making use of more than one indicator - rather than only on a simple one.

Here is some code showing how a mixed signal is created, based on the evaluation of two simple ones, which in turn may have different level of complexities, depending on the kind of indicators

used to build them. Notice that the function `sigFormula` can take more than two arguments, evaluating therefore several conditions at once.

```
## 'Bull market' (i.e. market considered underpriced --> indication to buy)
# if SMA50>SMA200
quantstrat::add.signal(
  strategy.st, name="sigCrossover", ## "name" is a function taking on
           # specific arguments
           arguments= list(columns=c("sma50","sma200"),relationship="gt"),
            # 'columns' refers to the indicators defined above
           label="smabuy") # 'label' will be the reference for
# the execution phase hereafter.

## Specifies all instance when RSI is below 30
## (indication of asset being oversold)
add.signal(strategy.st, name = "sigThreshold",
           arguments =
             list(column = "rsi.14", threshold = 30, relationship = "lt",
                  cross = TRUE), # signals
           label = "rsi14buy")

## sigFormula which indicates that both smabuy and rsi14buy must evaluate TRUE.
add.signal(strategy.st, name = "sigFormula",
           arguments = list(formula = "smabuy & rsi14buy",cross = TRUE),
           # notice that the formula is evaluating two booleans
           label = "entry1")
```

3.3.4 Executing tradings

The very last step of our strategy will be defining the trading operations to execute, once the relevant signals are observed. As shown in Table 4, only four signals trigger an order execution. We define accordingly four trading execution rules.

```
## Enter the position when SMA(50)>SMA(200), and the RSI<30
quantstrat::add.rule(
  strategy.st, name = "ruleSignal",
         arguments = list(sigcol = "entry1", sigval = TRUE,
                          orderqty = 1000, ordertype = "market",
                          orderside = "long", replace = FALSE,
                          prefer = "Open",
                          tradeSize = tradesize, maxSize = tradesize),
         type = "enter")


## Exit the position when SMA(50)<SMA(200), and the RSI>70
add.rule(strategy.st, name = "ruleSignal",
         arguments = list(sigcol = "exit1", sigval = TRUE,
                          orderqty = -1000 , ordertype = "market",
                          orderside = "long", replace = FALSE,
```

```
                          prefer = "Open",
                          tradeSize = tradesize,
                          maxSize = tradesize),
         type = "exit")


## Enter the position when (Closing price < Lower BBand) and RSI<30
add.rule(strategy = strategy.st, name= 'ruleSignal',
         arguments = list(sigcol = "entry2", sigval= TRUE,
                          orderqty = 1000, ordertype= 'market',
                          orderside= NULL,threshold=NULL ),
         type= "enter")


## Exit the position when (Closing price > Upper BBand) and RSI>70
add.rule(strategy = strategy.st, name= 'ruleSignal',
         arguments = list(sigcol = "exit2", sigval= TRUE,
                          orderqty = -1000, ordertype= 'market', orderside= NULL,
                          threshold=NULL ),
         type= "exit")
```

3.3.5 Running the model

At this stage, we are ready to run our model. It is safer to check first whether the code for the strategy we seek to implement is 'bug-free'. To do this, we run the `quantstrat::applyStrategy` command within the `base::try` function, which allows to "run an expression that might fail and allow the user's code to handle error-recovery" [8].

```
testing_strat <- base::try( quantstrat::applyStrategy(strategy.st,
                        portfolios=portfolio.st ))
## try is a function in one of the basis R packages

## We test whether our entire startegy was coded correctly
```

```
## [1] "Set capital to 100,000 Euros "
## initalize portfolio: portf.fullinitalize account: acct.fullinitializie strategy: strat.full
## [1] "2013-08-01 00:00:00 AIR.PA -100 @ 45.275002"
## [1] "2013-10-07 00:00:00 AIR.PA -100 @ 50.299999"
## [1] "2014-07-17 00:00:00 AIR.PA 100 @ 44.98"
## [1] "2014-09-05 00:00:00 AIR.PA -100 @ 49.189999"
## [1] "2015-01-26 00:00:00 AIR.PA -100 @ 50.139999"
## [1] "2015-02-19 00:00:00 AIR.PA -100 @ 51.990002"
## [1] "2015-03-02 00:00:00 AIR.PA -100 @ 56.09"
## [1] "2016-06-14 00:00:00 AIR.PA 100 @ 50.98"
## [1] "2016-06-17 00:00:00 AIR.PA 100 @ 50.830002"
## [1] "2016-12-14 00:00:00 AIR.PA -100 @ 62.400002"
## [1] "2016-12-16 00:00:00 AIR.PA -100 @ 64.18"
```

[8] https://stat.ethz.ch/R-manual/R-devel/library/base/html/try.html

```
## [1] "2017-02-24 00:00:00 AIR.PA -100 @ 68.349998"
## [1] "2017-10-25 00:00:00 AIR.PA -100 @ 83.470001"
## [1] "2017-11-01 00:00:00 AIR.PA -100 @ 87.379997"
## [1] "2018-06-15 00:00:00 AIR.PA -100 @ 103.599998"
## [1] "2013-06-25 00:00:00 FP.PA 100 @ 36.294998"
## [1] "2013-08-26 00:00:00 FP.PA -100 @ 42"
## [1] "2013-08-29 00:00:00 FP.PA -100 @ 42.509998"
## [1] "2014-05-09 00:00:00 FP.PA -100 @ 52.259998"
## [1] "2014-10-16 00:00:00 FP.PA 100 @ 42.555"
## [1] "2017-07-10 00:00:00 FP.PA 100 @ 42.764999"
## [1] "2018-05-18 00:00:00 FP.PA -100 @ 54.48"
## [1] "2014-01-09 00:00:00 MC.PA 100 @ 122.949997"
## [1] "2014-04-11 00:00:00 MC.PA -100 @ 141.199997"
## [1] "2014-07-28 00:00:00 MC.PA 100 @ 131.550003"
## [1] "2014-11-24 00:00:00 MC.PA -100 @ 144.300003"
## [1] "2014-12-18 00:00:00 MC.PA 100 @ 130.25"
## [1] "2015-02-05 00:00:00 MC.PA -100 @ 153.5"
## [1] "2015-03-12 00:00:00 MC.PA -100 @ 170.600006"
## [1] "2016-07-28 00:00:00 MC.PA -100 @ 153.350006"
## [1] "2016-10-12 00:00:00 MC.PA -100 @ 164.350006"
## [1] "2016-10-17 00:00:00 MC.PA -100 @ 165.399994"
## [1] "2017-01-20 00:00:00 MC.PA -100 @ 190.949997"
## [1] "2017-03-20 00:00:00 MC.PA -100 @ 200"
## [1] "2017-04-03 00:00:00 MC.PA -100 @ 204.050003"
## [1] "2017-04-25 00:00:00 MC.PA -100 @ 223.149994"
## [1] "2017-10-27 00:00:00 MC.PA -100 @ 253.949997"
## [1] "2018-04-11 00:00:00 MC.PA -100 @ 278.25"
## [1] "2018-10-11 00:00:00 MC.PA 100 @ 261.950012"
## [1] "2013-01-30 00:00:00 OR.PA -100 @ 113.550003"
## [1] "2013-04-03 00:00:00 OR.PA -100 @ 127.300003"
## [1] "2018-02-09 00:00:00 OR.PA 100 @ 172.350006"
## [1] "2018-10-11 00:00:00 OR.PA 100 @ 187.600006"
## [1] "2013-04-03 00:00:00 SAN.PA -100 @ 80.050003"
## [1] "2013-11-07 00:00:00 SAN.PA -100 @ 78.660004"
## [1] "2014-10-16 00:00:00 SAN.PA 100 @ 78.800003"
## [1] "2014-10-29 00:00:00 SAN.PA 100 @ 71.150002"
## [1] "2015-02-16 00:00:00 SAN.PA -100 @ 85.720001"
## [1] "2015-02-20 00:00:00 SAN.PA -100 @ 87.389999"
## [1] "2015-02-24 00:00:00 SAN.PA -100 @ 88.75"
## [1] "2015-04-10 00:00:00 SAN.PA -100 @ 98.75"
## [1] "2016-06-15 00:00:00 SAN.PA 100 @ 68.110001"
## [1] "2016-11-10 00:00:00 SAN.PA -100 @ 76.980003"
## [1] "2017-04-05 00:00:00 SAN.PA -100 @ 85.43"
## [1] "2017-05-05 00:00:00 SAN.PA -100 @ 89.690002"
## [1] "2017-12-06 00:00:00 SAN.PA 100 @ 73.25"
## [1] "2018-07-05 00:00:00 SAN.PA -100 @ 72.440002"
## [1] "2018-08-01 00:00:00 SAN.PA -100 @ 75.599998"
```

Since the strategy appears to run smoothly, we can update our portfolio and account to register the transactions and variations in the level of capital owned. We are now ready to extract the data on the performance, in the form of summary statistics and charts. Unfortunately, it seems that, due to an apparently still unfixed bug in the quantstrat package [9], it is not possible to export the strategy log. This is why we kept it printed to screen, as a full output of the command that runs the relevant strategy implementation script.

3.3.6 Performance evaluation

As summarized in the first line of Table 5, our strategy tested on a five-year panel with five stocks seems 'conservative', in that the maximum number of tradings on a stock is 17 (for MC.PA), with an average of roughly 12 across the five stocks. This may be due to the usage of long-term SMAs, which incorporate a low level of volatility. As the table below shows, for three out of five stocks, the strategy net performance is negative. In particular, trading on LVHM register a loss above €50000. On the other hand, the strategy yields a positive return for Total and Sanofi. The lines "Percent.Positive" and "Percent.Negative", combined with "Num.Trades", gives an idea of how many open positions were closed during the observed time frame, and if the operation carried a positive or negative yield. By looking at the transaction log, one can see that the strategy on both Air France was particualry unsuccessful because several short positions were opened (hence, with the 'expectation' of a fall in the prices), and were (forcibly) closed only 'at the end of the world' i.e. when we close the account and the portfolio at the end of our observed period, after prices had rather steadly surged. For LVHM we see a slightly different story, in which the first three couples of operations are three profitable buy and sell, after which the algorithms becomes inefficient and determines the same pattern as AIR.PA to be observed. Sanofi looks like the example of a successful implementation of our algorithm.

There are now some issues regarding our strategy that we would like to highlight, and that would require further inspection and study:

- the first n days of observations (where n is the highest parameter value across all indicators making use of moving averages) , should be left out. This is because during those n days only one out of two rules is implemented. This can clearly be source of biases. However, in this R framework , this option does not seem to be offered, since the input data is fed through the object `mktdata`, which is created automatically when applying the strategy;
- no transaction fees are considered. Hence, any possible gain obtained by this strategy is in fact a gross profit;
- the algorithm parameters were set by a 'thumb-rule' type of choice, i.e. no action was taken to determine whether any further restrictons were to be put in place to obtain a better strategy performance. For a better treatment on the code-related side of this matter, we refer the reader to Trice [2016] for a reference to parameter optimization and backtesting ;
- no in-depth study on the market conditions over the six years observed was made, neither on the specific stocks cases. Table 1 shows that the firms considered belong to four different industrial sectors - hence, some sector specificity may play a role in determining the evolution of prices;
- there is clearly the need for some constraints that prevent opening an excessive number of short positions - especially when the algorithm returns signals contrasting with price trend.

[9] https://github.com/braverock/quantstrat/issues/57

One remark on Table 5: it is generated through `blotter::tradeStats`, and its output cannot be truncated. Moreover, the function developement seems unfinished, as one can notice from the uncomplete description of the statistics [10] .

Table 5: Startegy trading statistics

Measure	AIR.PA	FP.PA	MC.PA	OR.PA	SAN.PA
Num.Txns	16	7	17	4	15
Num.Trades	1	3	4	1	2
Net.Trading.PL	-17778	2346	-52745	-11910	9853
Avg.Trade.PL	-17778	781.8	-13186	-11910	4926
Med.Trade.PL	-17778	830	1550	-11910	4926
Largest.Winner	0	570.5	2325	0	2204
Largest.Loser	-176.4	0	-6154	-6718	0
Gross.Profits	0	2346	5425	0	9853
Gross.Losses	-17778	0	-58170	-11910	0
Std.Dev.Trade.PL	NA	191.8	29992	NA	5728
Std.Err.Trade.PL	NA	110.8	14996	NA	4050
Percent.Positive	0	100	75	0	100
Percent.Negative	100	0	25	100	0
Profit.Factor	0	NA	0.09	0	NA
Avg.Win.Trade	NA	781.8	1808	NA	4926
Med.Win.Trade	NA	830	1825	NA	4926
Avg.Losing.Trade	-17778	NA	-58170	-11910	NA
Med.Losing.Trade	-17778	NA	-58170	-11910	NA
Avg.Daily.PL	-147.2	505.2	-182.4	-5955	1116
Med.Daily.PL	-161.4	483	1550	-5955	1102
Std.Dev.Daily.PL	38.33	57.55	4004	1078	907.1
Std.Err.Daily.PL	22.13	33.22	2002	762.5	453.5
Ann.Sharpe	-60.98	139.3	-0.72	-87.67	19.53
Max.Drawdown	-45318	-1522	-115215	-16030	-10475
Profit.To.Max.Draw	-0.39	1.54	-0.46	-0.74	0.94
Avg.WinLoss.Ratio	NA	NA	0.03	NA	NA
Med.WinLoss.Ratio	NA	NA	0.03	NA	NA
Max.Equity	1440	2440	9345	1045	16406
Min.Equity	-43878	-856.5	-105870	-14985	-3323
End.Equity	-17778	2346	-52745	-11910	9853

As one could have speculated by looking at the strategy log, only FP.PA and SAN.PA yield a positive return, under the active strategy we implemented. On the other hand, the three other stocks yield a negative return, with MC.PA being the worst among the three.

A quick glance at the risk measure indicator seems to suggest that a high volatility may have an implication on the accuracy of our automated trading strategy - indeed, AIR.PA and MC.PA were traded unprofitably, and their standard deviation is the highest. One may guess that the algorithm

[10] Full account of the statistics here: https://rdrr.io/rforge/blotter/man/tradeStats.html

was thrown off by the frequent changes in prices. There is clearly the need for a more thorough analysis of these statistics, which however goes beyond the scope of our analysis.

Table 6: Active strategy returns' statistics

Measure	AIR.PA DailyEqPL	FP.PA DailyEqPL	MC.PA DailyEqPL	OR.PA DailyEqPL	SAN.PA DailyEqPL
Cumulative Return	-0.202	0.023	-0.55	-0.121	0.09
Annualized Return	-0.036	0.004	-0.123	-0.021	0.014
Annualized Sharpe Ratio	-0.291	0.35	-0.411	-0.372	0.228
Annualized StdDev	0.125	0.011	0.299	0.056	0.063

Finally, we consider one very last measure of "fitness" of our startegy, which is by far the simplest: the level of capital left in our account, once all positions have been closed. We see from Figure 4 that the initial capital of €100000 has decreased to below €50000 (or $5e+04$). Moreover, there seems to be the need for a constraint on the capital invested, since towards the end of the observed period, one would actually get indebted. To our knowledge, one should hard-code an appropriate function to prevent more short positions being taken once the level of capital goes below zero.

Figure 4: Account Capital level evolution of active positions

3.3.7 Benchmarking with buy-and-hold strategy

The negative performance of our active portfolio strategy prompts the comparison with a passive strategy. We now implement a buy and hold strategy, over the same timeframe as before, with the same stocks we considered for our active strategy. Here we will allocate equal parts of the initial capital to each of the five stocks, and execute a buy order for each on the first day of observations. On the last day, we sell all the stocks we had bought. Thus, the increase(reduction) in the stock price will determine our gains(losses).

```
## [1] "Set capital to 100,000 Euros "
## [1] "2012-12-28 00:00:00 FP.PA 514 @ 38.91"
## [1] "2012-12-28 00:00:00 MC.PA 145 @ 137.800003"
## [1] "2012-12-28 00:00:00 SAN.PA 282 @ 70.730003"
## [1] "2012-12-28 00:00:00 AIR.PA 680 @ 29.395"
## [1] "2012-12-28 00:00:00 OR.PA 190 @ 104.800003"
## [1] "2018-12-31 00:00:00 FP.PA -514 @ 46.18"
## [1] "2018-12-31 00:00:00 MC.PA -145 @ 258.200012"
## [1] "2018-12-31 00:00:00 SAN.PA -282 @ 75.660004"
## [1] "2018-12-31 00:00:00 AIR.PA -680 @ 83.959999"
## [1] "2018-12-31 00:00:00 OR.PA -190 @ 201.199997"
```

Since each stock price has risen during the five years examined, it can be easily noted that this buy and hold strategy appears rather profitable: our initial € 100.000 investment stake has increased by about 80%. At this stage, it is difficult to think that an active strategy could have beaten the market. On the other hand, an in-depth study is required to examine what factors determined such increases in prices - whether it was that these five companies were particularly successful, or perhaps the French economy was revamping after the financial crisis. This in turn highlights the importance of an appropriate market study that should accompany the creation of the trading strategy. Clearly, investing is far from gambling, and coding or mathemathical skills cannot make up for a poor understanding of the markets and the economy.

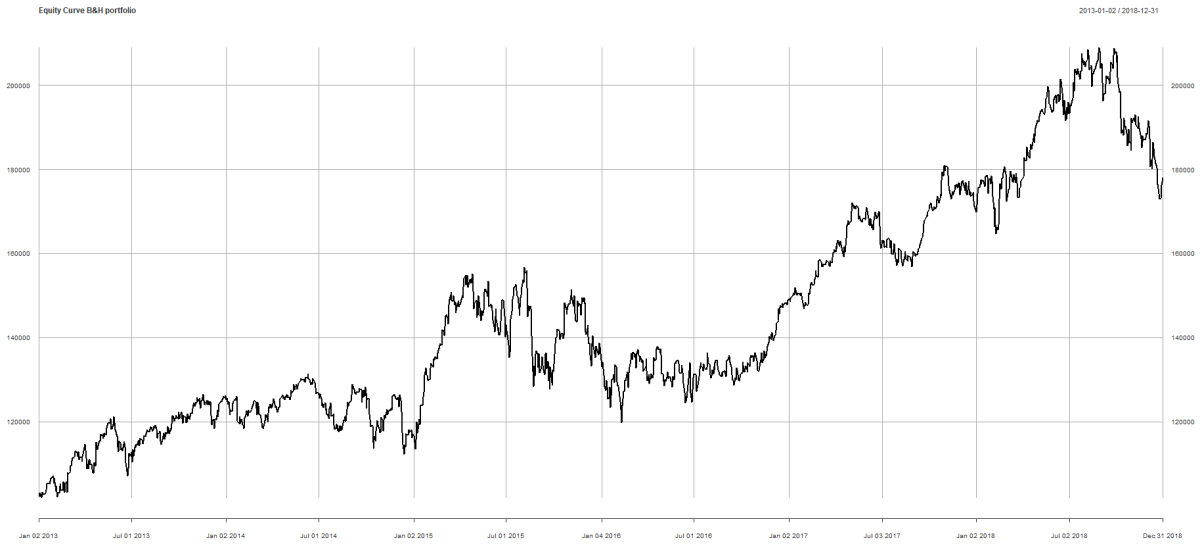

Figure 5: Account Capital level evolution of long positions in B&H strategy

3.4 Final comments and future steps

It is clear that our strategy implementation lacks of several important feautres, on which we already touched above. We left one important point untackled - namely, the absence of a proper testing strategy. As Guy Yollin illustrates [11], implementing a trading strategy would first require the so-called *walk-forward analysis*, which allows for a dynamic parametrization of the model, with a lower impact of overfitting [12]. Nevertheless, we would rather highlight the importance of having created a realistic model, which can constitute a first, solid building block towards a proper in-depth study of TA optimized applications to trading.

[11] http://www.r-programming.org/files

[12] https://algotrading101.com/learn/what-is-walk-forward-optimization/

4 Neural networks

Neural networks (NN) is a "black box" technique that finds non-linear connections and recognize patterns between one or multiple inputs to generate an output [13] . We will test a 1-input NN [14] and a multiple-input NN [15] in the attempt to predict the next-day stock price of our 5 stocks, over a period of 30 trading days, from the 16/11/2018 to the 31/12/2018. Based on the predictions, we will develop an investing strategy where everyday, for the above-mentioned 30 days, we will invest all our capital (starting at 10,000 euro) into the stock that will have the biggest predicted increase in price the next day, according to our model. Consequently, the next day we will sell all our stocks bought the previous day, and invest again all our capital with the same reasoning. In this analysis, all commission and brokerage fees are ignored.

4.1 Benchmark

To evaluate the performance of the NN as an investment decision-maker, instead of using a ROC Curve, we decided to compare it with a diversified, long-term investment strategy, where the initial capital of 10.000 euro is invested equally in the 5 stocks (2.000 euro per stock) at the beginning of the period (16/11/2018) and all stocks are then sold at the end of the period (31/12/2018).

Table 7: Benchmark strategy outcome

Dates	AIR.PA	FP.PA	MC.PA	OR.PA	SAN.PA		Capital
16/11/2018	95.86	92.88	45.19	65.09	77.09	stock price(euro)	Total budget(euro)
	2000	2000	2000	2000	2000	amount invested(euro)	10000
	20.86	21.53	44.25	30.72	25.94	N. of shares bought	
31/12/2018	98.59	83.95	39.47	61.5	72.01	stock price(euro)	Total budget(euro)
	2056.89	1807.72	1746.87	1889.7	1868.22	value in stocks(euro)	9369.43

As can be seen, the benchmark strategy has a return of -6%. Clearly, the higher the return of the NN-based investments, the more successful the NN algorithms will be. Moreover, in case of a return lower than -6% (lower than the "naive" strategy), the NN model will be considered as not useful in predicting stock prices.

4.2 1-input NN

The first NN model is a single hidden layer neural network. In this model there is one layer of input nodes that send weighted inputs to a subsequent layer of receiving nodes. The `nnetar` function in the forecast package fits a single hidden layer neural network model to a timeseries. The function model approach is to use lagged values of the time series as input data, reaching to a non-linear autoregressive model. The model takes as input the stock prices starting from the 31/12/2016 to 31/12/2018. The goal is to predict the stock prices between 16/11/2018 and 31/12/2018.

[13] https://pathmind.com/wiki/neural-network

[14] http://rpubs.com/kapage/523169?fbclid=IwAR3RW_tVA_7SgCah7M0nmZL7anIL4gS_ZZ3dP_i_w8AOyQXoTMwXC_wQsoE

[15] https://www.analyticsvidhya.com/blog/2017/09/creating-visualizing-neural-network-in-r/?fbclid=IwAR0z9tD0-WFxG3zVs8YmFaLF7RGnszbizQrhyGdyZG1-GL-D5coqBOnTiZE

```
library(prophet)
library(quantmod)
library(forecast)
library(xlsx)
library(tseries)
library(timeSeries)
library(dplyr)
library(fGarch)
#Download the prices for the 5 stocks for the desired time frame
getSymbols("AI.PA", src="yahoo", from="2016-12-31", to="2018-12-31")
getSymbols("AIR.PA", src="yahoo", from="2016-12-31", to="2018-12-31")
getSymbols("BN.PA", src="yahoo", from="2016-12-31", to="2018-12-31")
getSymbols("BNP.PA", src="yahoo", from="2016-12-31", to="2018-12-31")
getSymbols("DG.PA", src="yahoo", from="2016-12-31", to="2018-12-31")

alpha <- 1.5^(-10)
```

We generate the predicted price of the 30 days for the 5 stocks with a `for` loop, where the prices are predicted one at the time. Although more computationally intense, in this way we are able to predict the price of the day given all the "information" available until the day before.

```
for (n in 1:30){
  close_price <- as.numeric(AIR.PA[1:479+n,'AIR.PA.Close'])
#Hidden layers creation
  alpha <- 1.5^(-10)
  hn <- length(close_price)/(alpha*(length(close_price)+1))
#Fitting nnetar
  lambda <- BoxCox.lambda(close_price)
  dnn_pred <- nnetar(close_price, size= hn, lambda = lambda)
  dnn_forecast <- forecast(dnn_pred, h= 1, PI = TRUE)
  AIR.PA.P[n] = dnn_forecast[["mean"]]
}

for (n in 1:30){
   close_price <- as.numeric(BN.PA[1:479+n,'BN.PA.Close'])
   hn <- length(close_price)/(alpha*(length(close_price)+1))
   lambda <- BoxCox.lambda(close_price)
   dnn_pred <- nnetar(close_price, size= hn, lambda = lambda)
   dnn_forecast <- forecast(dnn_pred, h= 1, PI = TRUE)
   BN.PA.P[n] = dnn_forecast[["mean"]]
 }

 #We run the same for loop above also for AI.PA, BNP.PA and DG.PA
```

We create a table with the predicted daily increase in price for every stock during the 30 days. After that, we will identify the stock with the highest predicted increase in price for every day. This will be the stock where to invest all the capital for the day.

```
AIR.LOL <- AIR.PA.P - as.numeric(AIR.PA[480:509,'AIR.PA.Close'])
BN.LOL <- BN.PA.P - as.numeric(BN.PA[480:509,'BN.PA.Close'])
BNP.LOL <- BNP.PA.P - as.numeric(BNP.PA[480:509,'BNP.PA.Close'])
DG.LOL <- DG.PA.P - as.numeric(DG.PA[480:509,'DG.PA.Close'])

df = data.frame(AI.LOL,AIR.LOL,BN.LOL,BNP.LOL,DG.LOL)

for (n in 1:30){
   if (max(df[n,]) == df[n,"AI.LOL"]){
   AI.DIF[n] = "AI.PA"
   } else if (max(df[n,]) == df[n,"AIR.LOL"]){
   AI.DIF[n] = "AIR.PA"
   } else if (max(df[n,]) == df[n,"BN.LOL"]){
   AI.DIF[n] = "BN.PA"
   } else if (max(df[n,]) == df[n,"BNP.LOL"]){
   AI.DIF[n] = "BNP.PA"
   } else if (max(df[n,]) == df[n,"DG.LOL"]){
   AI.DIF[n] = "DG.PA"
   }
}
```

Table 8: 1-input NN strategy outcome

Day	Invest in	date	AI	AIR	BNP	BN	DG	#shares bought	€ Value
1	BNP	16/11/2018	95.86	92.88	45.19	65.09	77.09	221.26	10000
2	BNP	19/11/2018	95.40	92.57	45.28	64.70	76.80	221.26	10019.91
3	BNP	20/11/2018	93.5	91.07	44.30	64.98	76.37	221.26	9803.07
4	AI	21/11/2018	94.22	93.16	44.79	65.41	77.01	105.17	9910.38
5	AIR	22/11/2018	93.31	92.33	44.28	65.27	76.62	106.30	9814.77
6	AI	23/11/2018	93.86	93.41	44.36	65.87	76.86	105.78	9929.57
7	AI	26/11/2018	95.86	93.58	45.34	65.82	78.54	105.78	10141.15
8	AIR	27/11/2018	94.86	93.69	45.08	66.41	78.16	107.10	10035.36
9	BNP	28/11/2018	94.36	93.5	44.75	65.41	77.95	223.75	10013.94
10	AI	29/11/2018	94.68	94.86	44.77	65.54	77.58	105.81	10018.42
11	AI	30/11/2018	97.04	94.62	44.37	66.05	77.09	105.81	10268.51
12	AI	03/12/2018	97.27	95.69	44.84	65.41	75.40	105.81	10292.56
13	AI	04/12/2018	97.45	94.26	43.86	65.52	76.66	105.81	10311.81
14	BNP	05/12/2018	96.45	92.30	43.41	64.58	74.66	235.05	10206.00
15	DG	06/12/2018	94.63	88.63	41.78	63.43	70.98	138.37	9821.69
16	AI	07/12/2018	95.86	89.07	41.59	63.79	72.68	104.90	10056.92
17	DG	10/12/2018	95.77	87.48	40.40	62.97	71.66	140.20	10047.38
18	DG	11/12/2018	96.45	88.59	40.68	62.81	73.69	140.20	10333.41
19	AI	12/12/2018	97.54	91.48	41.90	64.12	74.77	107.48	10484.84
20	BN	13/12/2018	97.63	89.97	42.18	64.32	74.5	163.16	10494.62
21	BNP	14/12/2018	97.45	88.66	41.59	64.08	73.51	251.36	10455.46
22	DG	17/12/2018	96.68	87.65	40.58	63.23	72.36	140.96	10200.32

Day	Invest in	date	AI	AIR	BNP	BN	DG	#shares bought	€ Value
23	DG	18/12/2018	95.68	88.65	40.55	62.68	71.33	140.96	10056.54
24	AI	19/12/2018	97.86	87.19	40.88	62.82	71.59	103.13	10093.19
25	DG	20/12/2018	97.09	83.33	39.40	62.25	71.41	140.20	10013.50
26	AI	21/12/2018	97.81	83.09	39.5	62.23	71.41	102.36	10013.50
27	AIR	24/12/2018	97.09	81.62	38.79	61.61	70.5	121.75	9939.04
28	AIR	27/12/2018	95.31	82.09	38.54	60.27	70.63	121.75	9996.27
29	AI	28/12/2018	96.59	83.76	39.37	60.66	71.95	105.58	10198.39
30	AI	31/12/2018	98.59	83.95	39.47	61.50	72.01		10409.55

4.2.1 Evaluation of 1-input NN

As can be seen, the investment strategy based on the 1-input NN has a return of +4%, which is an extremely good result considering the benchmark strategy return of -6%. This implies that, even though all the 5 stocks had negative performances during the time frame, the NN successfully identified the right stock at the right time, leading to a positive return.

4.3 Multiple-input NN

The second NN model uses the `neuralnet` library for the analysis. It has 3 neurons in its hidden layer and the weights are calculated using a back propagation algorithm. The model takes several inputs that for simpliticy have been coded according to the following table:

Table 9: Variables coding

Variable	Identifier	Variable	Identifier	Variable	Identifier
Close price	A1	CPI monthly inflation	A10	PPE	A18
Open price	A2	Dividend	A11	Cash	A19
High price	A3	Revenue	A12	Total equity	A20
Low price	A4	Net Income	A13	Non current liability	A21
Volume	A5	Basic earnings per share	A14	Cash flows (used in)/from operating activities	A22
Adjusted price	A6	Diluted earnings per share	A15	Net cash flows (used in)/from investing activities	A23
Interest rate	A7	Total assets	A16	Net cash flows (used in)/from financing activities	A24
Exchange rate (1 USD)	A8	Intangible assets	A17	Net cash and cash equivalents at end of period	A25
Political stability	A9				

The goal is always to predict stock prices of the 30 days, hoping that the inclusion of additional variables like high price, low price, inflation and financial statements data will provide more accurate predictions.

```
#Download "AI1.csv" "AIR1.csv" "BN1.csv" "BNP1.csv"
#Download "DG1.csv" "PREDICTION SIMPLE.xlsx"
library(readr)
AI1 <- read_csv2("D:/2017-2018/data_analysis/technical_analysis/report/AI1.csv")
data <- AI1[, c(2:26)]
#we take all the data frame as training data to predict the next-day stock price
datatrain = data[ 1:1531, ]
datatest = data[ 1532, ]
#we normalize the values to avoid results based only on the disporportionate
#size of certain inputs
max = apply(data , 2 , max)
min = apply(data, 2 , min)
scaled = as.data.frame(scale(data, center = min, scale = max - min))
# install library
install.packages("neuralnet ")
#load library
library(neuralnet)
# creating training and test set
# the test set will be the closing price of a single day
# the training set will be all the data of the days before the day of the test set
trainNN = scaled[1:1531 , ]
testNN = scaled[1532 , ]
```

We want to predict every next day price for the 5 stocks. But first, we want to check weather the the model gives more accurate results with or without the financial statement data as inputs.

```
# fit neural network
set.seed(2)
# Develop a NN model with tomorrow closing price as output and
# all the other variables (taken today) as input
NN = neuralnet( A1 ~ A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10
+ A11 + A12 + A13 + A14 + A15 + A16 + A17 + A18 + A19 + A20 + A21
+ A22 + A23 + A24 + A25, trainNN, hidden = 3 , linear.output = T )
# plot neural network
plot(NN)
```

In the image below we can see a graphical representation of the neuronal net: It has 3 neurons in its hidden layer. The black lines show the connections with weights. The weights are calculated using a back propagation algorithm. The blue line displays the bias term.

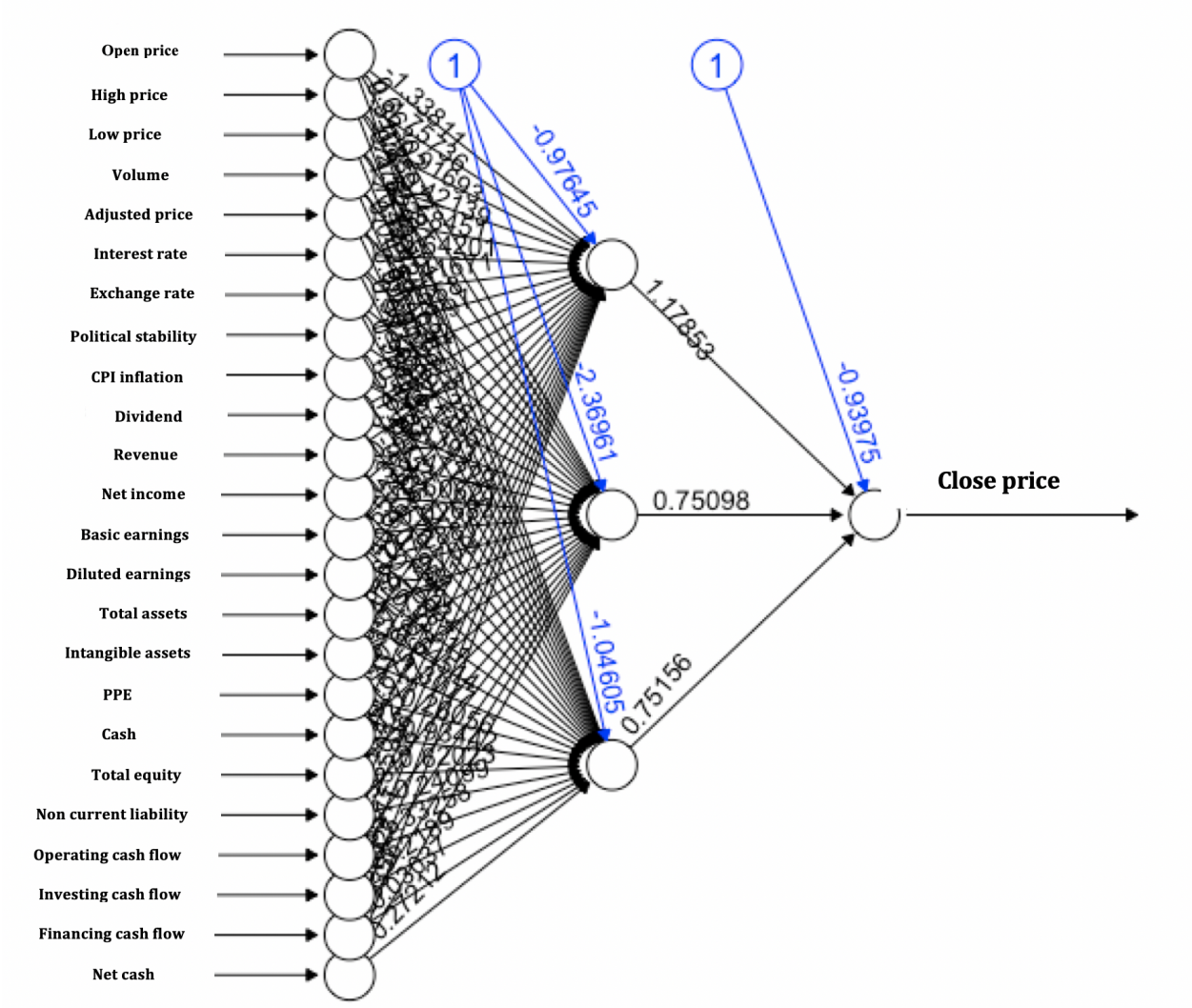

Figure 6: Neuronal net representation

```
predict_testNN = compute(NN, testNN[,c(2:25)])
# We descale the output to obtain the results in the real scale
predict_testNN = (predict_testNN$net.result * (max(data$A1) - min(data$A1)))
+ min(data$A1)
# We create an array where to store the results per day for each stck
AI.P <- c()
# We run the model for every single day per stock, to predict the prices
# one by one
View(predict_testNN)
for (n in 1:30){
  datatrain = data[ 1:1501+n, ]
  datatest = data[ 1502+n, ]
  max = apply(data , 2 , max)
  min = apply(data, 2 , min)
```

```
  scaled = as.data.frame(scale(data, center = min, scale = max - min))
  trainNN = scaled[1:1501+n , ]
  testNN = scaled[1502+n , ]
# model with financial statements data
  NN = neuralnet(A1 ~ A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10
  + A11 + A12 + A13 + A14 + A15 + A16 + A17 + A18 + A19 + A20 + A21
  + A22 + A23 + A24 + A25, trainNN, hidden = 3 , linear.output = T )
  predict_testNN = compute(NN, testNN[,c(2:25)])
  predict_testNN = (predict_testNN$net.result * (max(data$A1) - min(data$A1)))
  + min(data$A1)
  AI.P[n] <- predict_testNN
}
AI.P2 <- c()
for (n in 1:30){
  datatrain = data[ 1:1501+n, ]
  datatest = data[ 1502+n, ]
  max = apply(data , 2 , max)
  min = apply(data, 2 , min)
  scaled = as.data.frame(scale(data, center = min, scale = max - min))
  trainNN = scaled[1:1501+n , ]
  testNN = scaled[1502+n , ]
# model without financial statements data
  NN = neuralnet(A1 ~ A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10,
  trainNN, hidden = 3 , linear.output = T )
  predict_testNN = compute(NN, testNN[,c(2:10)])
  predict_testNN = (predict_testNN$net.result * (max(data$A1) - min(data$A1)))
  + min(data$A1)
  AI.P2[n] <- predict_testNN
}
# Calculate Root Mean Square Error (RMSE)
Reals <- data[1503:1532,1]
RMSE <- sqrt(sum((Reals-AI.P)^2)) # RMSE = 6.072
RMSE2 <- sqrt(sum((Reals-AI.P2)^2)) # RMSE2 = 6.064
```

As can be seen, the model without the financial statement variables (RMSE = 6.072) is more accurate than the model with them (RMSE2 = 6.064) according to the RMSE value. Hence, we will continue without taking into account the financial statement variables.

```
#We predict the prices of the other 4 stocks one by one,
#without the financial statement variables as input
library(readr)
AIR1 <- read_csv2("D:/2017-2018/data_analysis/technical_analysis/data/AIR1.csv")
data <- AIR1
AIR.P2 <- c()
for (n in 1:30){
  datatrain = data[ 1:1501+n, ]
  datatest = data[ 1502+n, ]
  max = apply(data , 2 , max)
```

```
  min = apply(data, 2 , min)
  scaled = as.data.frame(scale(data, center = min, scale = max - min))
  trainNN = scaled[1:1501+n , ]
  testNN = scaled[1502+n , ]
  NN = neuralnet(A1 ~ A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10,
  trainNN, hidden = 3 , linear.output = T )
  predict_testNN = compute(NN, testNN[,c(2:10)])
  predict_testNN = (predict_testNN$net.result * (max(data$A1) - min(data$A1)))
  + min(data$A1)
  AIR.P2[n] <- predict_testNN
}
#We run the same for loop above also for BN.PA, BNP.PA and DG.PA
```

We want to create a table with the predicted daily increase in price for every stock during the 30 days. After that, we will identify the stock with the highest predicted increase in price for every day. This will be the stock where to invest all the capital for the day.

```
AI.R <- AI1[1502:1531, c(2)]
AIR.R <- AIR1[1502:1531, c(1)]
BN.R <- BN1[1502:1531, c(1)]
BNP.R <- BNP1[1502:1531, c(1)]
DG.R <- DG1[1502:1531, c(1)]
AI.DIF <- AI.P2 - AI.R
AIR.DIF <- AIR.P2 - AIR.R
BN.DIF <- BN.P2 - BN.R
BNP.DIF <- BNP.P2 - BNP.R
DG.DIF <- DG.P2 - DG.R
df = data.frame(AI.DIF,AIR.DIF,BN.DIF,BNP.DIF,DG.DIF)
#identify the stock with the highest predicted increase in price for every day
X <- c()
for (n in 1:30){
  if (max(df[n,]) == df[n,"A1"]){
    X[n] = "AI.PA"
  } else if (max(df[n,]) == df[n,"A1.1"]){
    X[n] = "AIR.PA"
  } else if (max(df[n,]) == df[n,"A1.2"]){
    X[n] = "BN.PA"
  } else if (max(df[n,]) == df[n,"A1.3"]){
    X[n] = "BNP.PA"
  } else if (max(df[n,]) == df[n,"A1.4"]){
    X[n] = "DG.PA"
  }
}
```

Table 10: Multiple-input NN strategy outcome

Day	Invest in	date	AI	AIR	BNP	BN	DG	#shares bought	€ Value
1	DG	16/11/2018	95.86	92.88	65.09	45.19	77.09	129.70	10000
2	DG	19/11/2018	95.40	92.57	64.70	45.28	76.80	129.70	9961.09
3	DG	20/11/2018	93.5	91.07	64.98	44.30	76.37	129.70	9906.61
4	AIR	21/11/2018	94.22	93.16	65.41	44.79	77.01	107.21	9989.62
5	AIR	22/11/2018	93.31	92.33	65.27	44.28	76.62	107.21	9899.55
6	DG	23/11/2018	93.86	93.41	65.87	44.36	76.86	130.30	10015.35
7	DG	26/11/2018	95.86	93.58	65.82	45.34	78.54	130.30	10234.27
8	AIR	27/11/2018	94.86	93.69	66.41	45.08	78.16	108.69	10184.75
9	DG	28/11/2018	94.36	93.5	65.41	44.75	77.95	130.36	10163.01
10	DG	29/11/2018	94.68	94.86	65.54	44.77	77.58	130.36	10113.47
11	DG	30/11/2018	97.04	94.62	66.05	44.37	77.09	130.36	10050.90
12	DG	03/12/2018	97.27	95.69	65.41	44.84	75.40	130.36	9829.29
13	DG	04/12/2018	97.45	94.26	65.52	43.86	76.66	130.36	9993.54
14	AIR	05/12/2018	96.45	92.30	64.58	43.41	74.66	105.43	9732.82
15	DG	06/12/2018	94.63	88.63	63.43	41.78	70.98	131.66	9345.87
16	DG	07/12/2018	95.86	89.07	63.79	41.59	72.68	131.66	9569.70
17	AIR	10/12/2018	95.77	87.48	62.97	40.40	71.66	107.84	9435.40
18	DG	11/12/2018	96.45	88.59	62.81	40.68	73.69	129.64	9555.11
19	AI	12/12/2018	97.54	91.48	64.12	41.90	74.77	99.39	9695.13
20	DG	13/12/2018	97.63	89.97	64.32	42.18	74.5	130.25	9704.18
21	AIR	14/12/2018	97.45	88.66	64.08	41.59	73.51	108.00	9576.52
22	AIR	17/12/2018	96.68	87.65	63.23	40.58	72.36	108.00	9466.36
23	AIR	18/12/2018	95.68	88.65	62.68	40.55	71.33	108.00	9574.36
24	DG	19/12/2018	97.86	87.19	62.82	40.88	71.59	131.53	9417.76
25	AIR	20/12/2018	97.09	83.33	62.25	39.40	71.41	112.73	9394.08
26	AIR	21/12/2018	97.81	83.09	62.23	39.5	71.41	112.73	9368.16
27	AI	24/12/2018	97.09	81.62	61.61	38.79	70.5	94.78	9202.44
28	AIR	27/12/2018	95.31	82.09	60.27	38.54	70.63	110.04	9034.42
29	DG	28/12/2018	96.59	83.76	60.66	39.37	71.95	128.08	9217.09
30	AIR	31/12/2018	98.59	83.95	61.50	39.47	72.01		9224.77

4.3.1 Evaluation of Multiple-input NN

As can be seen, the multiple-input based investment has a return of -7%. This shows that this method cannot be used to predict stock prices. We believe this is because the method predicts prices based on just the previous-day values of several variables, hence without considering the time-serie behind the price trend. Instead, the 1-input NN took into account the whole trend of the prices during the previous 1 year and a half.

4.4 Neuronal Network Conclusion

To conclude, we can argue that the 1-input NN could be used to invest in stocks. We recommend to further test and validate this method over a larger number of stocks (100) and over a longer time frame (1 year of predicted prices) before investing in real life with this model.

5 Random forest

Random forest is "a predictor consisting of a collection of randomized base regression trees" (Biau [2012] (p. 2)) , which determine in expectation the estimate of a random parameter. In our framework, random forest is used for stock price prediction. This is a supervised learning that randomly creates and merges multiple decision trees into one common forest. The benefit of this technique is that, rather than relying on a single tree, it combines all the trees at once, giving the optimum result.

The basic training principle of decision trees is the recursive partitioning of the feature space using a tree structure, where each root node is split until pure nodes, i.e nodes which contain samples of a single class, are achieved (to Denil et al. [2014]).

5.1 Why Random Forest?

As it is termed as one of the easiest machine learning algorithms according to Saranya and Anandan [2019], it gives accurate results and reduces overfitting of the model with a good accuracy in the prediction. Nonetheless, given the high volatility and instability in the stock markets, prediction has become very challenging. The random forest algorithm randomly selects different observations and features to build several decision trees and then takes aggregate of all the decision trees. It can be classified into two types: classification and regression. In our model, we are doing a regression based on some continuous variables that might have a relevant effect on stock prices.

5.2 Methodology

70% of the data is used to train the model and 30% is used to test it. The basic approach of the model is to learn the pattern and relationships in the data from the training set and then to reproduce it in the test set. In this model, we are computing the random forest regression by considering 19 variables that could affect the stock closing prices.

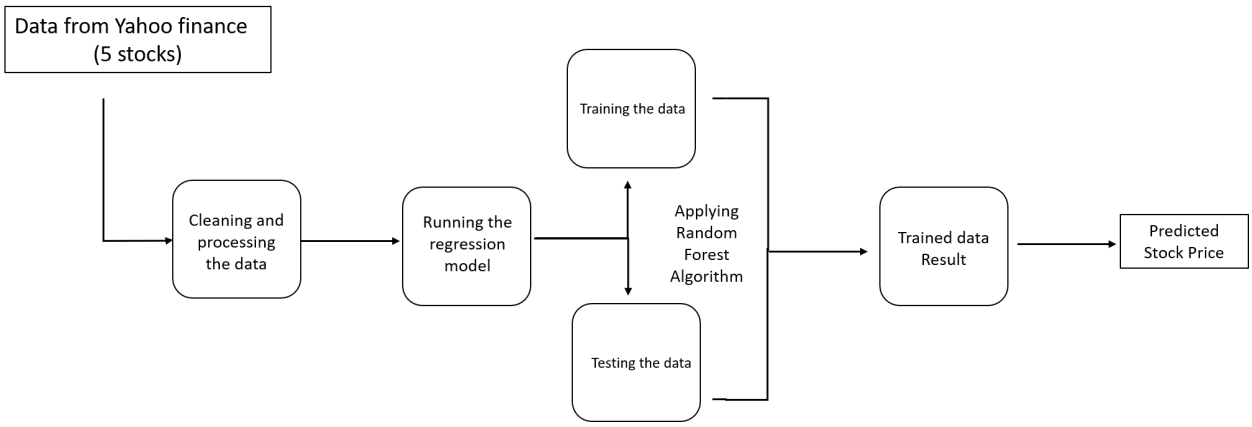

Figure 7: Flow chart of the process

5.2.1 Summarizing the Data

To begin with, we import the data of all the 5 stock prices from Yahoo finance. In order to have a summary of the data we use different functions such as `summary`, which is a generic function used to

produce result summaries of various model fitting functions. Then we use the `str` function, which is a solid way to display the structure of an R object. `str` will give the output of the information on one line for each basic structure. The `dim` function gives the number of dimensions in the data.

This file shows all the prices of Air Liquide with all the 19 variables for six years, from 2013-2018.

```
##Air Liquide

AI_FINALDATA

dim(AI_FINALDATA)    #dimension of the matrix
summary(AI_FINALDATA)  #summary of the data
str(AI_FINALDATA)  #structure of the data
```

Therefore, the above code just gives an idea about the dataset which we will be using in our model.

–>

5.2.2 Training the machine

This phase consists of feeding the algorithm with the training data so that the model can understand the pattern of the data. Here is the code for dividing the data into train and test data. Before dividing, we need some packages which are required to get the result we want.

```
library(party)
#install caret package
install.packages('caret')
#load package
set.seed(123)
library(caret)
trainAI = createDataPartition(AI_FINALDATA$AI.PA.Close,
                              p=0.7, list=FALSE,times=1)

train_1 = AI_FINALDATA[1:1075,]    #1075 obs out of 1533
test_1 = AI_FINALDATA[1076:1533,]    #458 obss out of 1533
```

`party` (library) - used for recursive partitioning

`caret` - (Classification And Regression Training) is a set of functions that attempt to streamline the process for creating predictive models. This package is very important for dividing the data into two.

`set.seed` - Set the seed of R's random number generator, which is useful for creating simulations or random objects that can be reproduced.

`createDataPartition` - A series of test/training partitions are created using createDataPartition while createResample creates one or more bootstrap samples. The main function which is used to split the data.

Now we will use these two datasets for predicting the stock prices and applying the random forest algorithm.

5.2.3 Data Scoring

The method of applying a predictive model to a dataset is called the scoring data process according to Saranya and Anandan [2019]. The last part of this module describes how the result of the model can infer whether a stock will rise or sink, based on certain parameters. It also shows the vulnerabilities of a particular stock or entity.

```
#random forest parameters
#fitting the model or creating the forest
library(randomForest)
output.forestAI <- randomForest(AI.PA.Close ~  Interest.rates +
ExchangeRate + PoliticalStability + Infaltion + AirLiquideDividend
+ REVENUE + NET.INCOME + Basic.earnings.per.share + Diluted.earnings.per.share
+ TOTAL.ASSETS  + INTANGIBLE.ASSETS + PPE + CASH  +  TOTAL.EQUITY
+ NON.CURRENT.LIABILITY + Cash.flows.from.operating.activities +
Net.cash.flows.from.investing.activities +
Net.cash.flows.from.financing.activities +
Net.cash.and.cash.equivalents, train_1)

# View the forest results.
print(output.forestAI)
```

`mtry` = number of variables selected at each split. lower `mtry` means less correlation between trees

`randomForest`(library) - implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

the output gives an object of class randomForest, which is a list with the following components:

`call`:the original call to randomForest

`type`: regression

`predicted`: the predicted values of the input data based on out-of-bag samples.

Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 6

From the plot we can observe that in the case of Air Liquide, Interest rate has a significant impact on closing price.

```
VI_F=importance(output.forestAI)

VI_F

write.csv(VI_F , "VI_FAI.csv")


varImpPlot(output.forestAI,type=2)
```

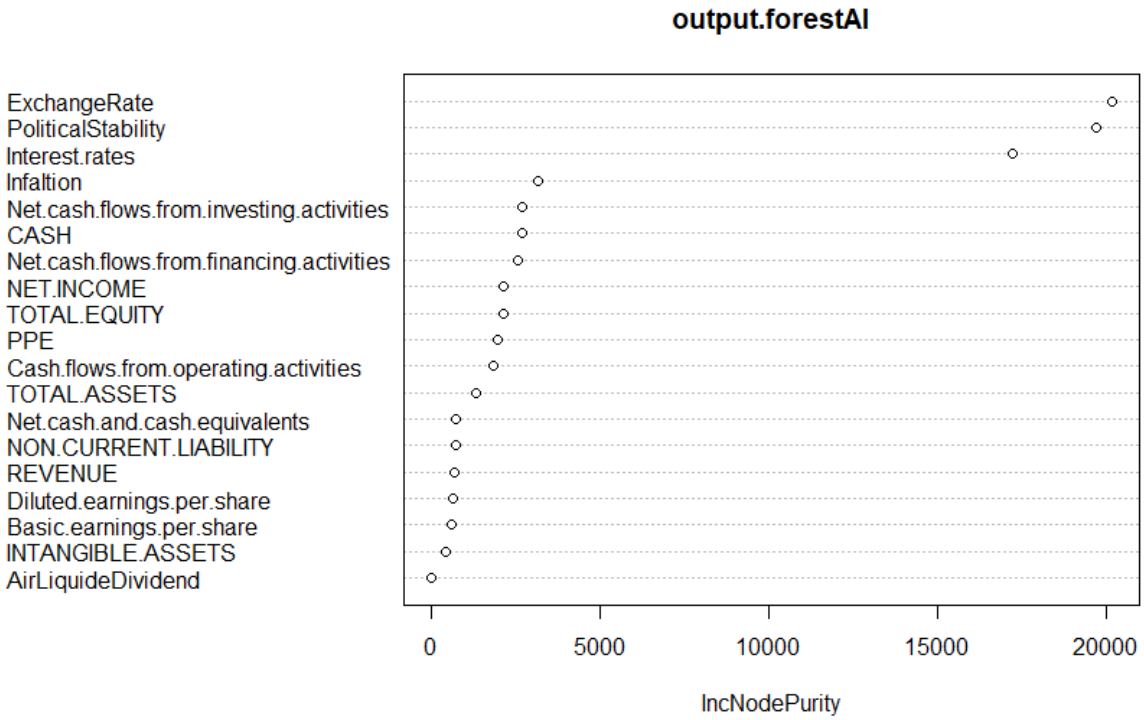

Figure 8: Random Forest of Air Liquide

`IncNodePurity` relates to the loss function that determines how splits are chosen.

`importance` - A matrix that measures the importance of each predictor variable. Columns show different measures of importance. `varImplot` - the importance of the variables that were plotted.

The result we get by VI_F is that again Interest rate is the factor by which we get the best splits to understand our data.

5.2.4 Experimental Results

Now, with the help of our random forest and datasets , we will predict the future prices and we will check how accurate are the results. This will be done by getting a confusion matrix. A confusion matrix calculates a cross-tabulation of observed and predicted classes with associated statistics. In short, it compares the actual values and the predicted values so that we can get to know how reliable are our predictions.

```
#Predictions

PredictionsAdjusted <- predict(output.forestAI, test_1, type='response')
t <- table(predictions=PredictionsAdjusted, actual=test_1$AI.PA.Close)
t
write.csv(t , "ConfusionMatrix_AI.csv")
```

This gives the Confusion Matrix for Air Liquide.

```
#Accuracy Metric
sum(diag(t))/sum(t)
```

The accuracy metric in random forest gives the percentage of how accurate the predictions are. In our case, the accuracy metric is given below:

```
Accuracy Metric: 0.01310044 # or 1.31%
```

This has a very low value which means that we are only 1% accurate in predicting our prices. The reason behind this low accuracy rate is that we have only considered a few variables, as in the real life scenario there are many variables including the opening, high and low prices of the same day, which are ignored in this case. Furthermore, there are other variables which cannot be quantified leading to lack of accuracy. However, another reason is that with an increase in the number of trees and `mtry` (number of splits) we increase the overfitting and complexity of the model which leads to low accuracy.

```
#high strength of tree =Low error rate of individual tree classifier

PredictionWithProbs <- predict(output.forestAI, test_1, type = "response")
PredictionWithProbs          #predicting stock prices for 458 days
write.csv(t , "PredictionsPrices_AI.csv")
```

This function gives 458 predicted values on the basis of the train data of 1075 observations. This data can be used to develop a trading strategy where we decide where to invest based on the predictions.

```
# to find the best "mtry": m try means number of splits at each point.
bestmtryAI <- tuneRF(train_1,train_1$AI.PA.Close, ntreeTry = 200,
stepFactor = 1.5, improve = 0.01, trace = T, plot= T)
```

`OOB Error` : Out-of-bag error, also called out-of-bag estimate, is a method to measure the prediction error of random forests, boosted decision trees.

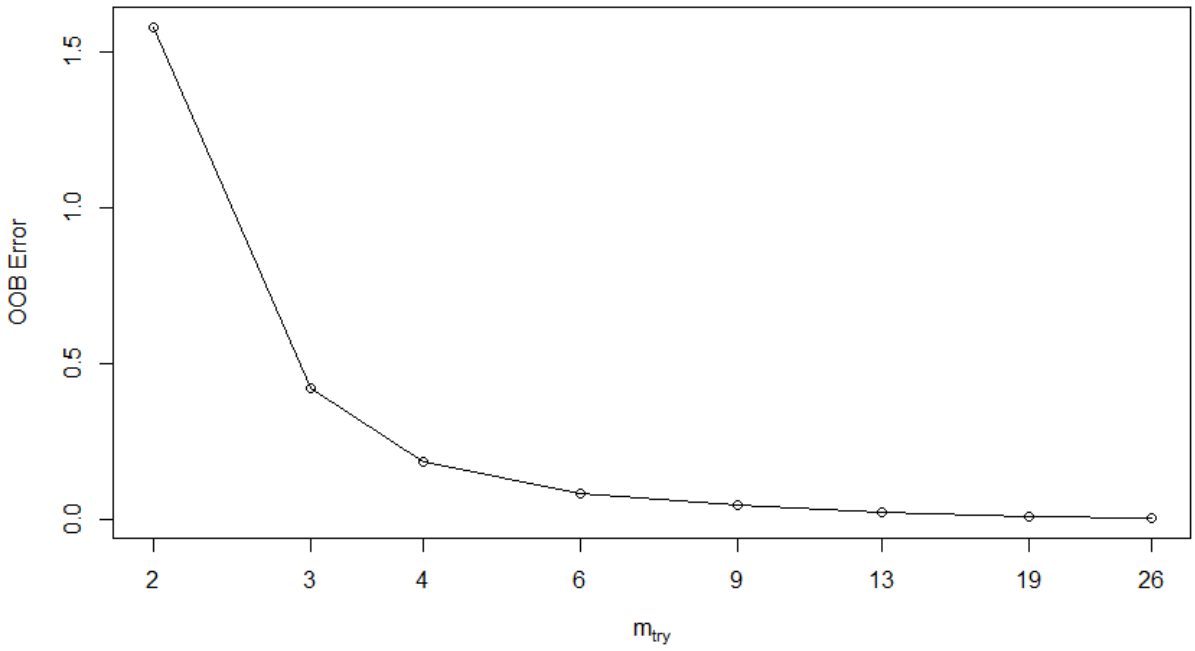

Figure 9: OOB Error of Air Liquide

We saw the whole alogrithm applied on the dataset of Air Liquide with 5 years data. Now, we will repeat all the steps above for all other 4 stocks and we will interpret the results for all of them. The output given below shows the results for the other stocks by using the same code applied for Air Liquide.

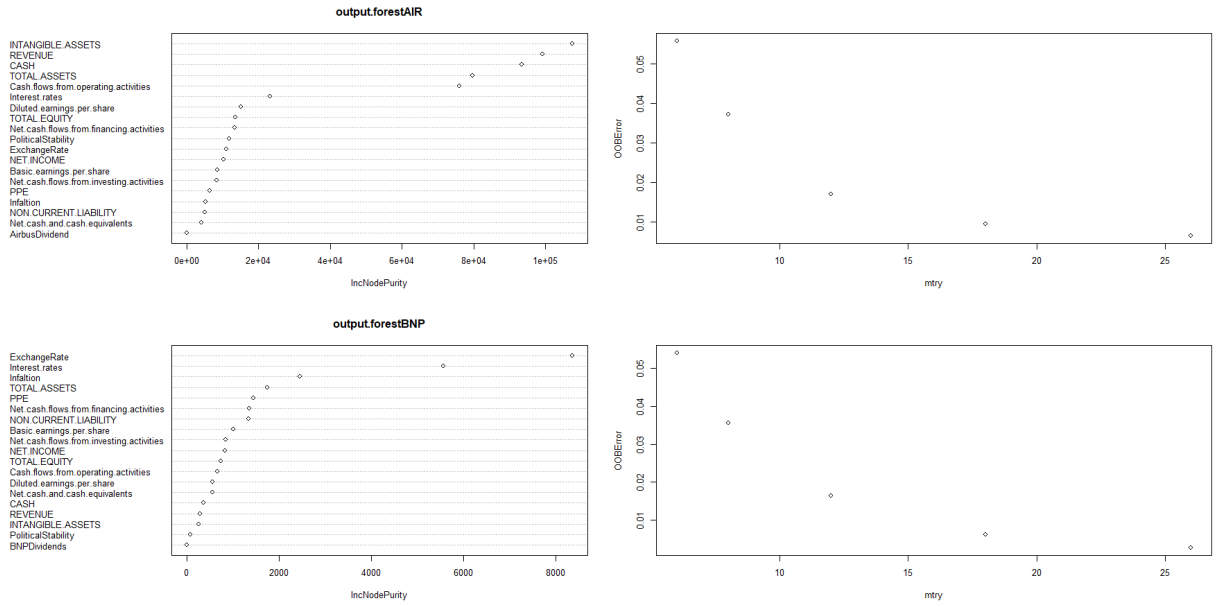

Figure 10: RANDOM FOREST AND OOB ERRORS FOR AIRBUS AND BNP PARIBAS

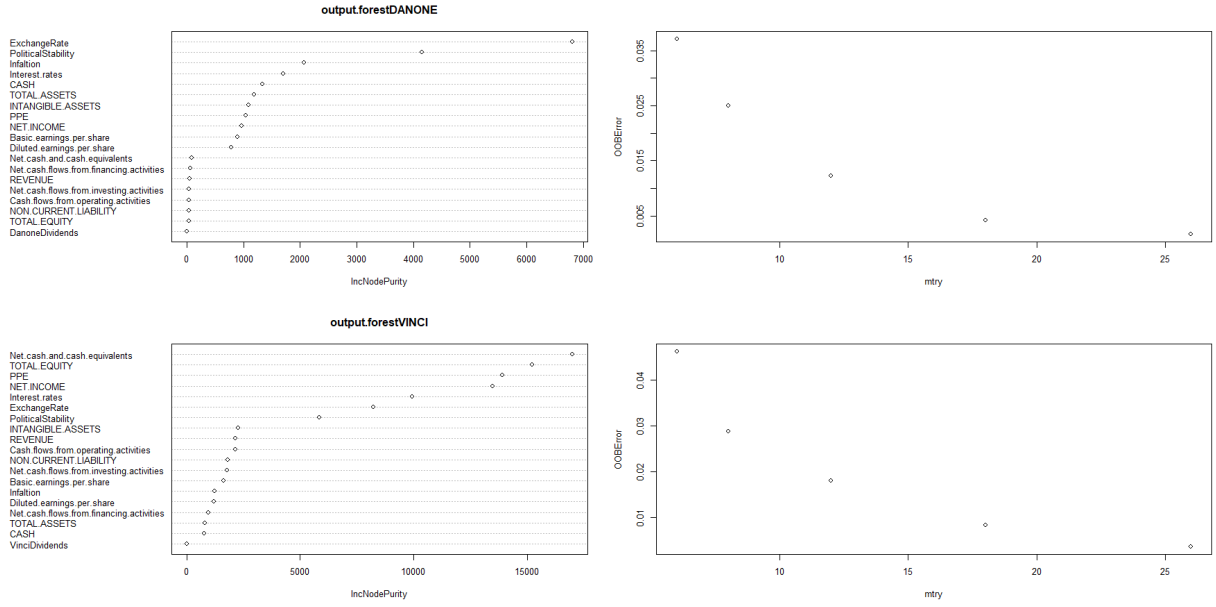

Figure 11: RANDOM FOREST AND OOB ERRORS FOR DANONE AND VINCI

5.3 Investment strategy

In general, the accuracy level of our model is very low for all the five stocks. Nonetheless, to have a clearer view on its performance, we will develop an investment strategy based on it and analyze the results. The graph below compares the predictions of the 5 stocks from 15/03/2017 to 31/12/2018.

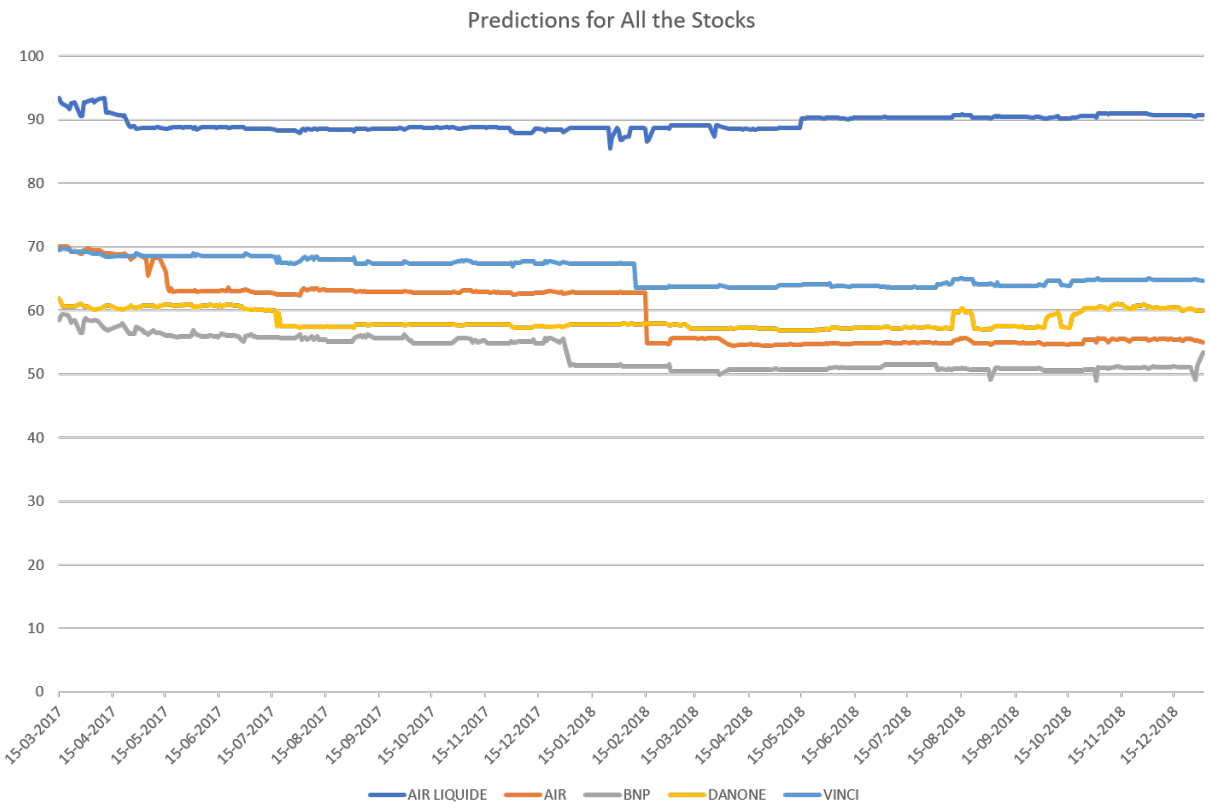

Figure 12: Predictions of All the 5 Stocks

The investing strategy executed is the same one used in Part 4 for the neuronal network models. We buy today the stock that, according to our model, will have the higher increase in price tomorrow, and sell the stock the day after, repeating the process every day.

The benchmark strategy consists of a diversified buy and hold investment:

Table 11: Diversified Strategies

X	AI.PA	AIR.PA	BNP.PA	BN.PA	DG.PA	X.1	Capital
15-03-2017	94.55	70.43	60.09	62.62	69.45	stock price(euro)	Total budget(euro)
	2000	2000	2000	2000	2000	amount invested(euro)	10000
	21.15	28.4	33.28	31.94	28.8	N. of shares bought	
	NA	NA	NA	NA	NA		
31-12-2018	98.59	83.96	39.47	61.51	72.02	stock price(euro)	Total budget(euro)
	2086	2384	1314	1965	2074	value in stocks(euro)	9822.21

The table above shows the diversified strategy according to which, we would get a negative return on our investment of -0.018. This implies that the diversified strategy would result in a loss for the investor.

The file PA contains all the predictions we found out from 15/03/2017 to 31/12/2018 for all the 5 stocks. The file DAT contains all the real closing prices of all the 5 stocks from 14/03/2017 to 28/12/2018. In the code below, we first calculate the predicted day-by-day returns for the 5 stocks and then simulate an investing strategy where we buy everyday the stock with the highest predicted return, and sell it the day after.

```
PA
DAT

RETU <- (PA-DAT)/DAT
View(RETU)
I <- c()
for (n in 1:458){
    if (max(RETU[n,]) == RETU[n,1]){
        I[n] = 1
    } else if (max(RETU[n,]) == RETU[n,2]){
        I[n] = 2
    } else if (max(RETU[n,]) == RETU[n,3]){
        I[n] = 3
    } else if (max(RETU[n,]) == RETU[n,4]){
        I[n] = 4
    } else if (max(RETU[n,]) == RETU[n,5]){
        I[n] = 5
    }
}
X = 10000
for (n in 1:456){
    if (I[n] == 1){
        Y = 1
    } else if (I[n] == 2){
        Y = 2
    } else if (I[n] == 3){
        Y = 3
    } else if (I[n] == 4){
        Y = 4
    } else if (I[n] == 5){
        Y = 5
    }
    X = X*(DAT[n+2,Y]/DAT[n+1,Y])
}
View(X)
```

We see that, following the random forest algorithm, starting with 10,000 euro, we would end up with 10014.39 euro after 1 year and 8 months of trading. This shows that the strategy outperforms the passive investment but a return of 0.14% after 20 months is far from being an ideal scenario.

5.4 Comments and Future Implementation

Future steps for this analysis would be to get rid of drawbacks of the model and try to attain a higher level of accuracy by using more complex random forest functions and adding other parameters which could impact price. The additional use of traditional algorithms and data mining techniques could also improve the predictability power of the model.

References

Gérard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13: 1063–1095, 2012. ISSN 15324435.

William Brock and Josef Lakonishok. Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *The Journal of Finance*, 47(5):1731–1764, 1992. URL http://www.jstor.org/stable/2328994.

Misha Denil, David Matheson, and Nando De Freitas. Narrowing the gap: Random forests in theory and in practice. In *31st International Conference on Machine Learning, ICML 2014*, volume 2, pages 1006–1016, 2014. ISBN 9781634393973.

Eugene F Fama. Efficient Capital Markets: A Review of Theory and Empirical Work Author(s):. *The Journal of Finance*, 25(2):383–417, 1970. URL https://www.jstor.org/stable/2325486Accessed:.

Po Hsuan Hsu, Mark P. Taylor, and Zigan Wang. Technical trading: Is it still beating the foreign exchange market? *Journal of International Economics*, 102:188–208, 2016. ISSN 18730353. doi: 10.1016/j.jinteco.2016.03.012. URL http://dx.doi.org/10.1016/j.jinteco.2016.03.012.

Somayeh Mousavi, Akbar Esfahanipour, and Mohammad Hossein Fazel Zarandi. A novel approach to dynamic portfolio trading system using multitree genetic programming. *Knowledge-Based Systems*, 66:68–81, 2014. ISSN 09507051. doi: 10.1016/j.knosys.2014.04.018. URL http://dx.doi.org/10.1016/j.knosys.2014.04.018.

Christopher Neely, Paul Weller, and Rob Dittmar. Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. *The Journal of Financial and Quantitative Analysis*, 32(4):405, 1997. ISSN 00221090. doi: 10.2307/2331231.

Salih N. Neftci. Naive Trading Rules in Financial Markets and Wiener-Kolmogorov Prediction Theory: A Study of "Technical Analysis". *The Journal of Business*, 64(4):549, 1991. ISSN 0021-9398. doi: 10.1086/296551.

A. Saranya and R. Anandan. Stock market prediction using machine learning algorithms. *International Journal of Recent Technology and Engineering*, 8(2 Special Issue 4):280–283, 2019. ISSN 22773878. doi: 10.35940/ijrte.B1052.0782S419. URL https://www.ijeat.org/wp-content/uploads/papers/v8i4/D6321048419.pdf.

Mark P. Taylor and Helen Allen. The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, 11(3):304–314, 1992. ISSN 02615606. doi: 10.1016/0261-5606(92)90048-3.

Tim Trice. Backtesting Strategies with R, 2016. URL https://timtrice.github.io/backtesting-strategies/basic-strategy.html.

Ko Chiu Yu. Techincal Analysis with R, 2019. URL https://bookdown.org/kochiuyu/Technical-Analysis-with-R/.