Data science report V1

Karan Soneja

19/12/2019

Abstract

A short summary of the work.

Contents

1 Introduction

What motivates us to do this work. Touch on important finding in literature.

2 Data

Present data used in the analysis

2.1 Stocks

2.2 Other economic data

3 Technical analysis: a naive implementation

Introduce work in quantmod with tracked portfolios

4 Neural networks

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 according 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. At first, we run a preliminary linear multiple regression for all the stocks separately to detect which input is affecting the prices the most. Then, we divide the dataset into testing data and training data.

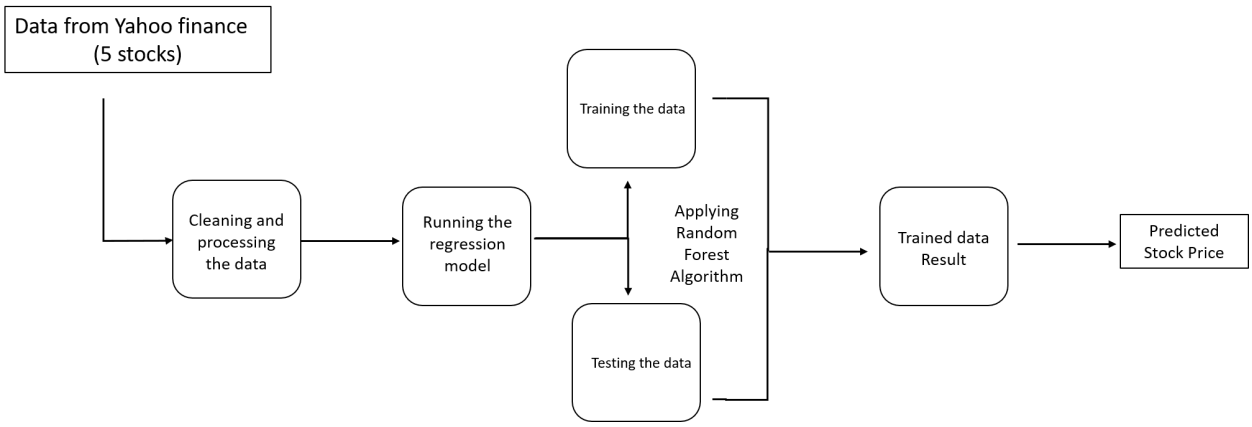

Figure 1: FLOW CHART

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 Regression Model

Next step is to run a Linear regression, which is used to predict the value of a continuous variable Y (Closing Prices in our case) based on one or more input predictor variables X.

Table 1: Additional variables

	Variables Used		Variables Used
1	AI.PA.Close	11	TOTAL.ASSETS
2	Interest.rates	12	INTANGIBLE.ASSETS
3	ExchangeRate	13	PPE
4	PoliticalStability	14	CASH
5	Infaltion	15	TOTAL.EQUITY
6	AirLiquideDividend	16	NON.CURRENT.LIABILITY
7	REVENUE	17	Cash.flows.from.operating.activities
8	NET.INCOME	18	Net.cash.flows.from.investing.activities
9	Basic.earnings.per.share	19	Net.cash.flows.from.financing.activities
10	Diluted.earnings.per.share	20	Net.cash.and.cash.equivalents

The aim is to establish a mathematical formula between the response variable (Y) and the predictor variables $(X_1, X_2, ..., X_{19})$. We use the multiple linear regression function for each stock price to infer the effect of each variable on the respective stock price. The generalized equation in mathematical terms for a linear regression is given by:

$Y = \beta_1 + X\beta_2 + \epsilon$, where

Y: dependent variable or target variable (Closing price);

X: Independent variable or predictor variable(a panel with the 19 chosen variables);

β_2: Coefficient(amount) of the variable affecting Y;

ϵ: error term or the residual.

```
lm([target variable] ~ [predictor variables], data = [data source])
```

After applying the regression we can summarize the results with the following code:

```
##Air Liquide
##regression of the Air Liquide adjusted price with respect to other varaibles

fitAI <- lm(AI_FINALDATA$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, AI_FINALDATA)

summary(fitAI) # show results

# exporting data into csv
write.csv(summary(fitAI) , "fitAI.csv")



# Other useful functions
coefficients = coefficients(fitAI) # model coefficients
confint(fitAI, level=0.95) # CIs for model parameters
fitted(fitAI) # predicted values
residuals(fitAI) # residuals
anova(fitAI) # anova table
vcov(fitAI) # covariance matrix for model parameters
influence(fitAI) # regression diagnostics
```

```
combined <- read.csv(
  "D:/2017-2018/data_analysis/technical_analysis/report/fitAI.csv")
pander::pander(combined, caption = "Regression",
               justify= "left", split.table= Inf)
```

Table 2: Regression

X	x
(Intercept)	-704.2
Interest.rates	27.17
ExchangeRate	113
PoliticalStability	9.982
Infaltion	1.686
AirLiquideDividend	-0.06095
REVENUE	-0.07478
NET.INCOME	-0.5111

X	x
Basic.earnings.per.share	48626
Diluted.earnings.per.share	-48509
TOTAL.ASSETS	0.04458
INTANGIBLE.ASSETS	NA
PPE	NA
CASH	NA
TOTAL.EQUITY	NA
NON.CURRENT.LIABILITY	NA
Cash.flows.from.operating.activities	NA
Net.cash.flows.from.investing.activities	NA
Net.cash.flows.from.financing.activities	NA
Net.cash.and.cash.equivalents	NA

Below is an explanation of the functions used below:

`coefficients`- coef is a function which extracts the coefficients of the model from objects returned by modeling functions. It gives the value of the variable that has affected the target variable. This effect can be either positive or negative which means that either it increases with increase in that variable or vice-versa.

`confint` - This computes confidence intervals for parameters in a fitted model.

`fitted` - It extracts the fitted values from the objects returned by modeling functions.

`residuals` - It extracts residuals of the model from objects returned by modeling functions

`anova` - It computes analysis of variance (or deviance) tables for one or more fitted model objects.

`vcov` - It is used for obtaining the estimated variance-covariance matrix of parameter estimates in a fitted model.

`influence` - This function provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of regression fits.

5.2.3 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 partytioning

`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.

5.2.4 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)
```

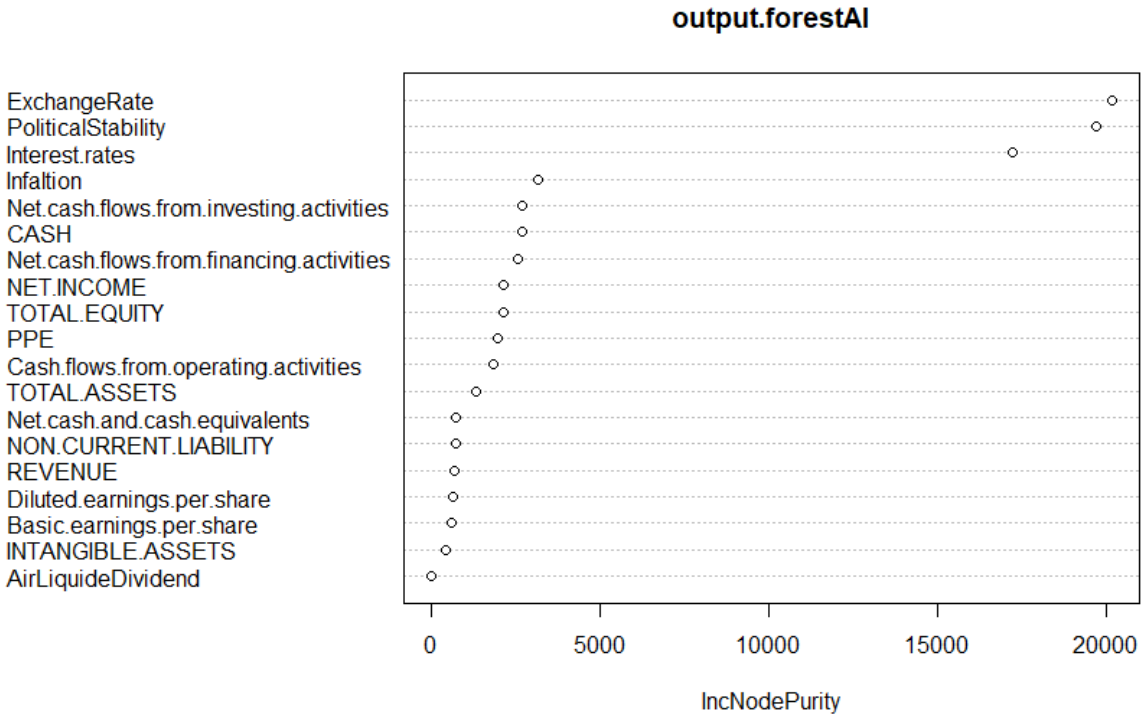

Figure 2: Random Forest of Air Liquide

`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.

5.2.5 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.

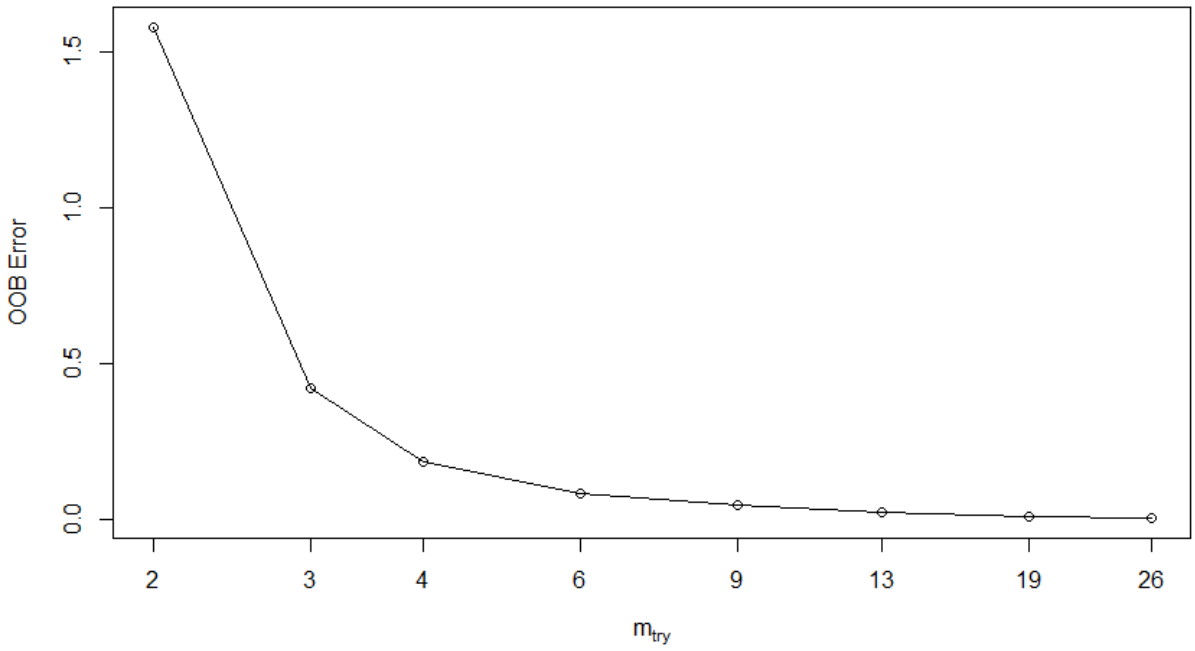

Figure 3: OOB Error of Air Liquide

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.

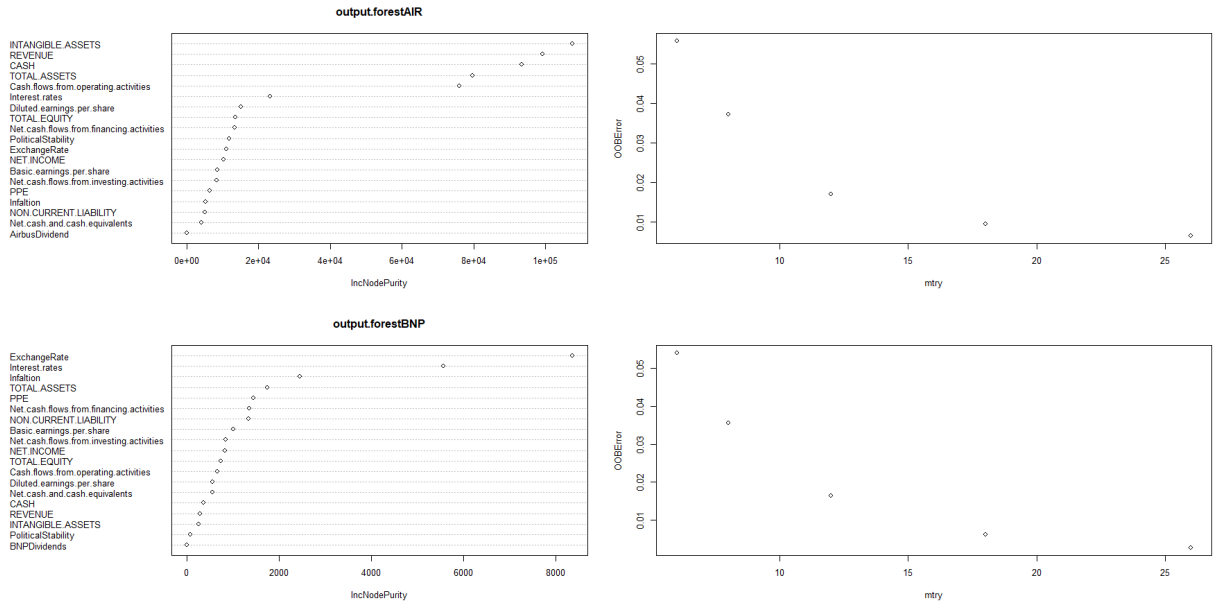

Figure 4: RANDOM FOREST AND OOB ERRORS FOR AIRBUS AND BNP PARIBAS

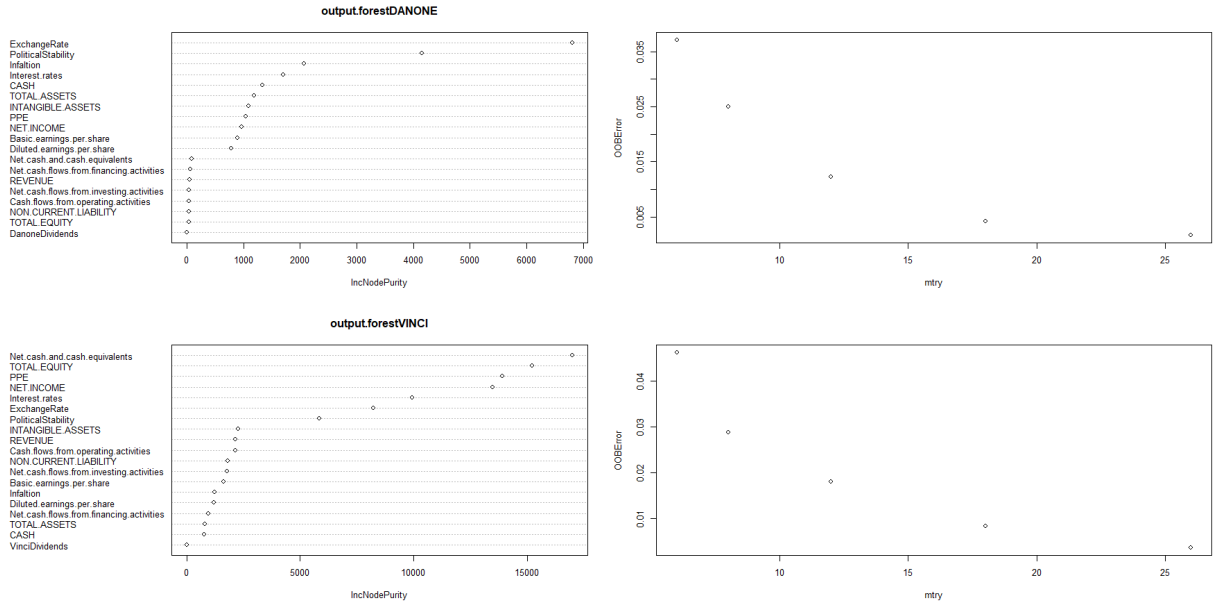

Figure 5: RANDOM FOREST AND OOB ERRORS FOR DANONE AND VINCI

5.2.6 Conclusion

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.

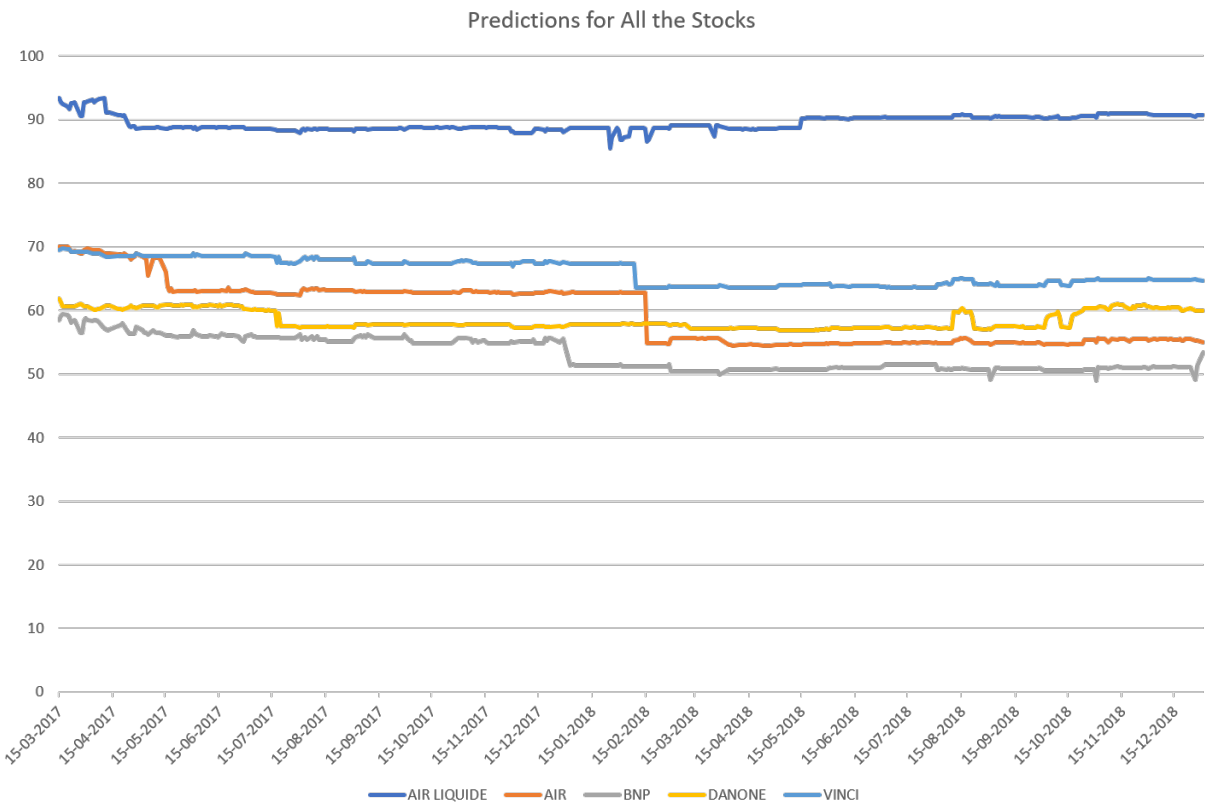

Figure 6: Predictions of All the 5 Stocks

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 3: 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.

5.2.7 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.

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.

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.