A PROJECT REPORT ON

Stock Market Prediction Using Random Forest Algorithm

Submitted in partial fulfillment of the course CSE4027 Data Analysis

Under the guidance of
Dr. N. Senthil Murugan,
Senior Assistant Professor, VIT-AP University.

SUBMITTED BY:

NAOMI GEORGE	19BCE7572
KOTHA SAI LAXMAN	19BCE7505
YANDRA HEMANTH	19BCE7567
CHINTA SURYA VARSHIT	19BCE7494

TABLE OF CONTENTS

ABSTRACT

In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The report focuses on the use of Random forest based model to predict stock values. Factors considered are open, close, low, high and volume. An algorithm with high accuracy we do the process of comparison for the accuracy of each of the model and finally is considered as better algorithm for predicting stock price. As share market is a vague domain we cannot predict the conditions occur, and also share market can never be predicted, this job can be done easily and technically through this work and the main aim of this paper is to apply algorithms in predicting the stock prices.

OVERVIEW

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

The random forest means data about data estimator. It fits a number decision trees on various sub samples of the given data. It control over-fitting. It improve the predictive accuracy.

PROBLEM STATEMENT

Stock Exchanges are financial institutions which allow transferability of different goods (monetary values, actions, precious metals) between stock broker components. With a turnover of trading around thousands of billions of dollars, this gets people's eager attention of making a profit. Goods are traded on the market, following their subsequent value to determine if the transaction has generated profit or not. In general the value of a stock is given by its entry on the stock exchange and the volume of its transactions. The more a share is transacted, the more it is valuable, and conversely, if a share is put into transaction in a low volume, it is not so important for some traders and by default its value decreases. This anticipation of the market can generate profits or losses. Depending on the power to predict future values. Therefore the problem becomes: for a given Stock market history, determine the moment of buying/entry or the selling/exit the good for generating profit. An aspect that has attracted researchers is predicting the values of the goods. Thus an algorithm used to train models, namely Random Forest, is used to tackle the problem.

WORKING OF RANDOM FOREST:

Algorithm:
Step 1: From the dataset pick N random records.
Step 2: Based on N records, build a decision tree.
Step 3: From the algorithm, choose the number of trees and repeat steps 1 & 2
Step 4: In case of a regression problem, for a new record, each tree in the forest predicts a value for Y(output)

SOFTWARE REQUIREMENTS

Code:

##READING THE DATASET##

```
stock <- read.csv("D://R/Machine Learning/Stock Market Prediction
Using random Forest/istanbul_stock_exchange.csv")

print(head(stock))

print(str(stock))

print(summary(stock))
```

## DATA MANIPULATION ##

```
library(dplyr)

max.exchange <- filter(stock, TL.BASED.ISE > 0 & USD.BASED.ISE > 0)
print(head(max.exchange))

for(i in 1:dim(stock)) {
  if(stock$TL.BASED.ISE[i] < 0 && stock$USD.BASED.ISE[i] < 0) {
    stock$Result[i] <- 'Loss'
  } else {
    stock$Result[i] <- 'Profit'
  }
}
print(stock$Result)
```

## DATA CLEANING ##

```
print(is.na(stock))
print(sum(is.na(stock$date)))
print(sum(is.na(stock$TL.BASED.ISE)))
print(sum(is.na(stock$USD.BASED.ISE)))
print(sum(is.na(stock$SP)))
print(sum(is.na(stock$DAX)))
print(sum(is.na(stock$FTSE)))
print(sum(is.na(stock$NIKKEI)))
print(sum(is.na(stock$BOVESPA)))
print(sum(is.na(stock$EU)))
print(sum(is.na(stock$EM)))
```

## DATA VISUALIZATION ##

```
library(ggplot2)
library(plotly)

pl <- ggplot(max.exchange, aes(TL.BASED.ISE, USD.BASED.ISE)) +
geom_point(color='red', alpha=0.5)
pl2 <- ggplotly(pl)
print(pl2)

p <- ggplot(stock, aes(USD.BASED.ISE)) +
geom_histogram(aes(fill=Result), color='black', bins=50, alpha=0.6) +
theme_bw() + xlab('USD Limit') + ylab('Count') + ggtitle('Market
Share')
# print(p)
plot <- ggplot(stock, aes(x=stock$TL.BASED.ISE,
y=stock$USD.BASED.ISE))
plot2 <- plot + geom_hex() + xlab('Trade Limit') + ylab('USD Limit')
+ ggtitle('Stock Trade v USD')
print(plot2 + scale_fill_gradient(high='red', low='blue'))
```

## APPLYING ML ##
## TRAIN TEST SPLIT ##

```
library(caTools)
set.seed(101)
sample <- sample.split(stock$Result, SplitRatio = 0.70)
train <- subset(stock, sample==T)
test <- subset(stock, sample==F)
```

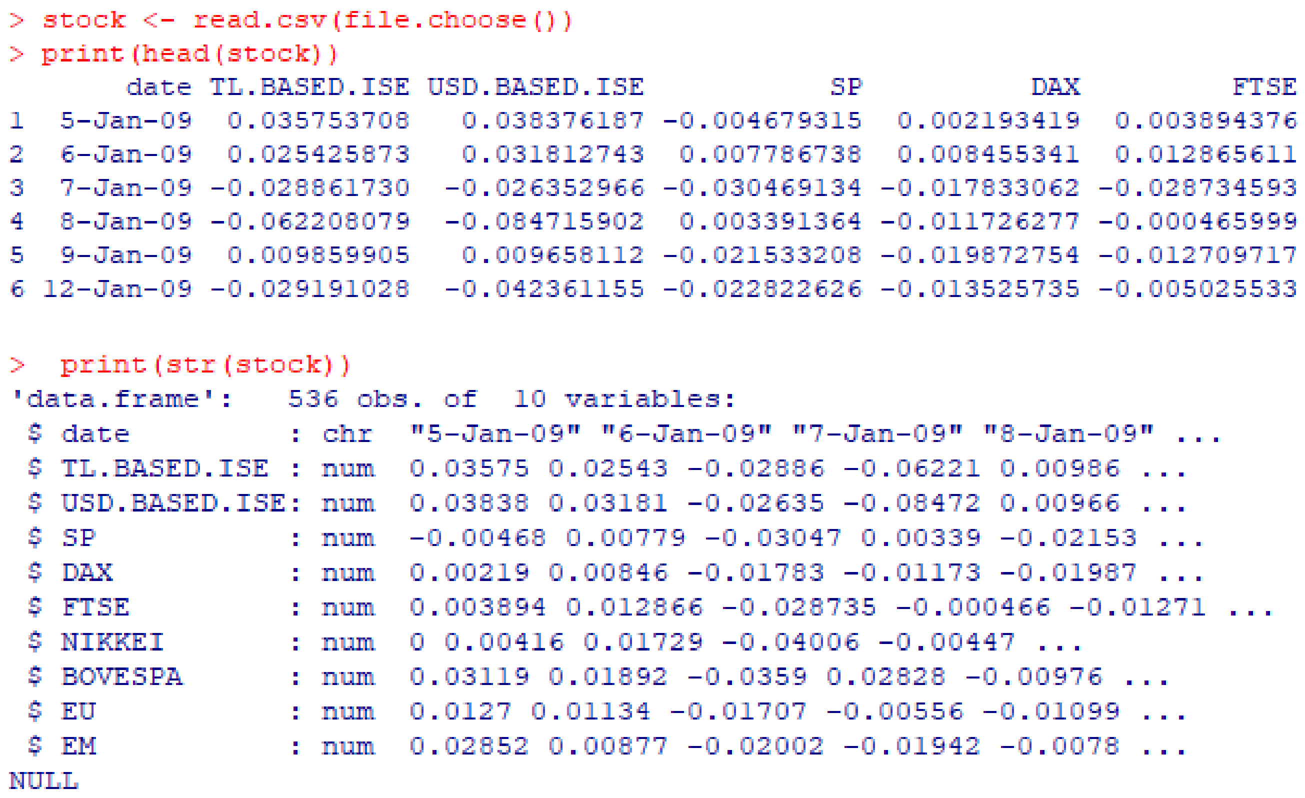
## TRAIN MODEL ##

```
library(randomForest)
rf.model <- randomForest(as.factor(Result) ~ ., data = train,
importance = TRUE)

# print(rf.model$confusion)

# print(rf.model$importance)
```

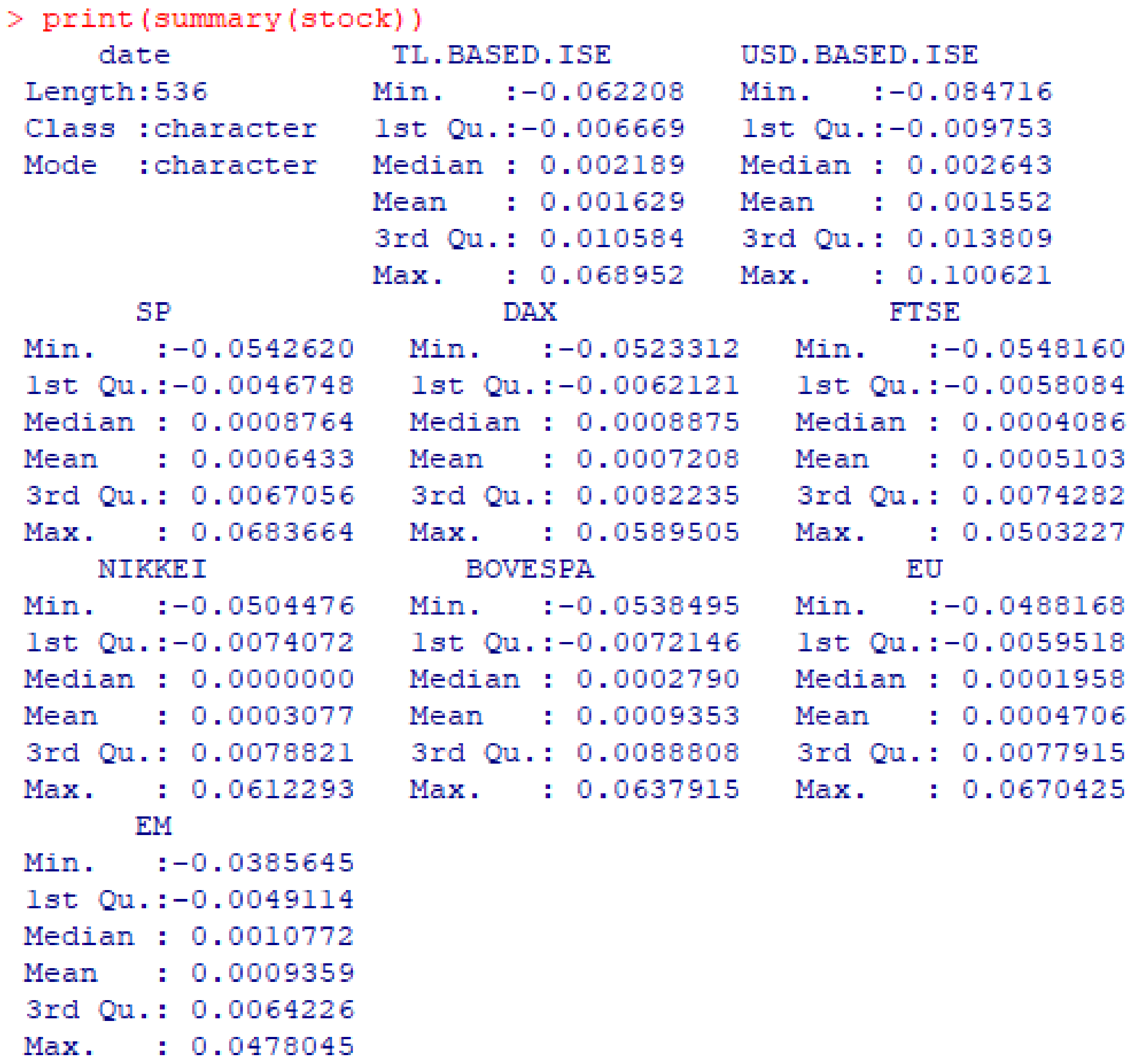
## PREDICTIONS ##

```
rf.preds <- predict(rf.model, test)
print(table(rf.preds, test$Result))
```

OUTPUT:

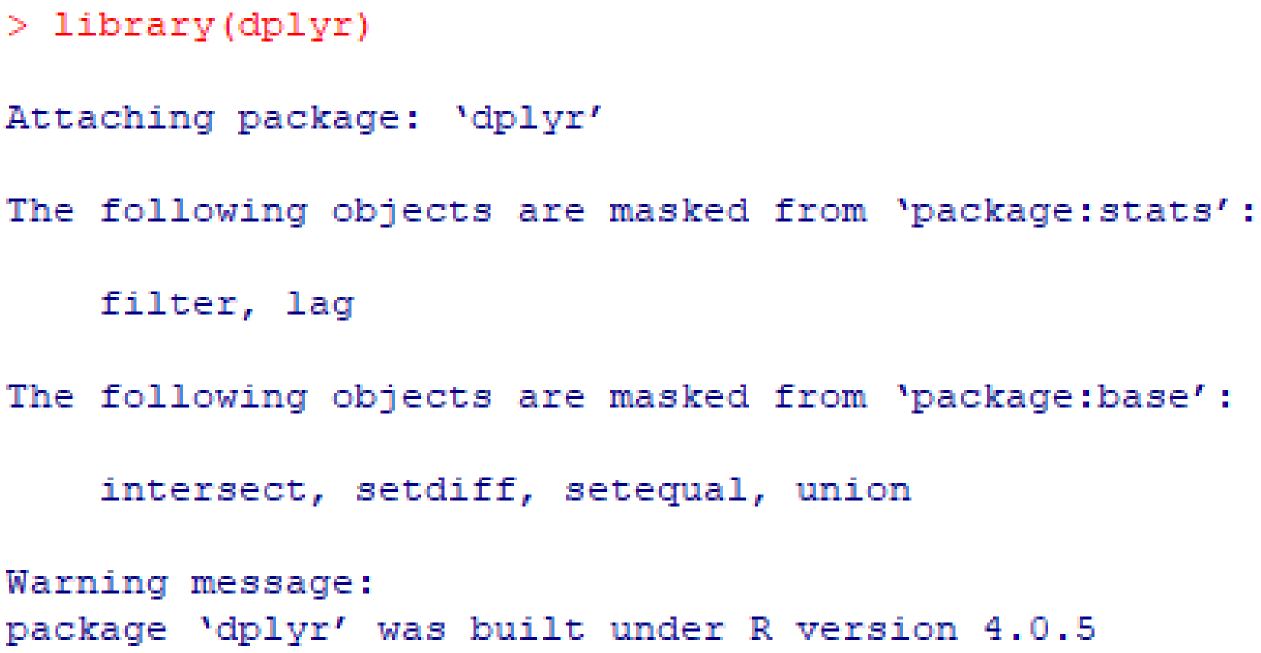


```
> stock <- read.csv(file.choose())
> print(head(stock))
       date TL.BASED.ISE USD.BASED.ISE           SP          DAX         FTSE
1  5-Jan-09  0.035753708   0.038376187 -0.004679315  0.002193419  0.003894376
2  6-Jan-09  0.025425873   0.031812743  0.007786738  0.008455341  0.012865611
3  7-Jan-09 -0.028861730  -0.026352966 -0.030469134 -0.017833062 -0.028734593
4  8-Jan-09 -0.062208079  -0.084715902  0.003391364 -0.011726277 -0.000465999
5  9-Jan-09  0.009859905   0.009658112 -0.021533208 -0.019872754 -0.012709717
6 12-Jan-09 -0.029191028  -0.042361155 -0.022822626 -0.013525735 -0.005025533

>  print(str(stock))
'data.frame':   536 obs. of  10 variables:
 $ date         : chr  "5-Jan-09" "6-Jan-09" "7-Jan-09" "8-Jan-09" ...
 $ TL.BASED.ISE : num  0.03575 0.02543 -0.02886 -0.06221 0.00986 ...
 $ USD.BASED.ISE: num  0.03838 0.03181 -0.02635 -0.08472 0.00966 ...
 $ SP           : num  -0.00468 0.00779 -0.03047 0.00339 -0.02153 ...
 $ DAX          : num  0.00219 0.00846 -0.01783 -0.01173 -0.01987 ...
 $ FTSE         : num  0.003894 0.012866 -0.028735 -0.000466 -0.01271 ...
 $ NIKKEI       : num  0 0.00416 0.01729 -0.04006 -0.00447 ...
 $ BOVESPA      : num  0.03119 0.01892 -0.0359 0.02828 -0.00976 ...
 $ EU           : num  0.0127 0.01134 -0.01707 -0.00556 -0.01099 ...
 $ EM           : num  0.02852 0.00877 -0.02002 -0.01942 -0.0078 ...
NULL
```

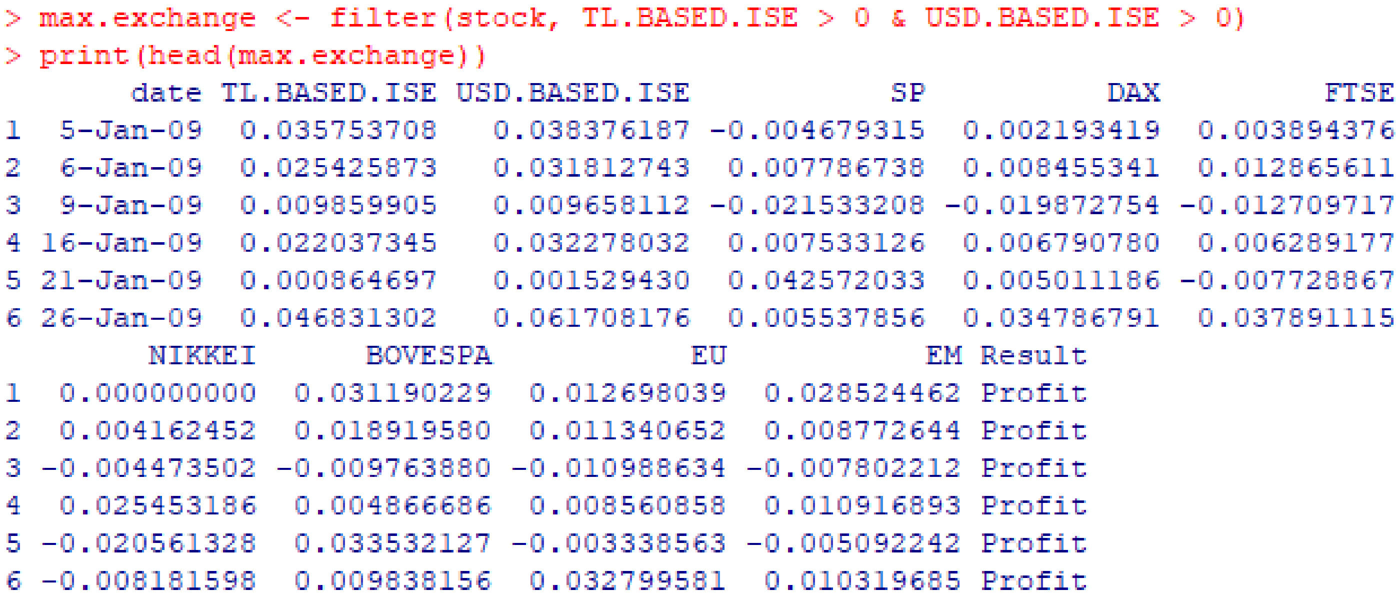


```
> print(summary(stock))
     date            TL.BASED.ISE       USD.BASED.ISE
 Length:536         Min.   :-0.062208   Min.   :-0.084716
 Class :character   1st Qu.:-0.006669   1st Qu.:-0.009753
 Mode  :character   Median : 0.002189   Median : 0.002643
                    Mean   : 0.001629   Mean   : 0.001552
                    3rd Qu.: 0.010584   3rd Qu.: 0.013809
                    Max.   : 0.068952   Max.   : 0.100621
       SP                  DAX                  FTSE
 Min.   :-0.0542620   Min.   :-0.0523312   Min.   :-0.0548160
 1st Qu.:-0.0046748   1st Qu.:-0.0062121   1st Qu.:-0.0058084
 Median : 0.0008764   Median : 0.0008875   Median : 0.0004086
 Mean   : 0.0006433   Mean   : 0.0007208   Mean   : 0.0005103
 3rd Qu.: 0.0067056   3rd Qu.: 0.0082235   3rd Qu.: 0.0074282
 Max.   : 0.0683664   Max.   : 0.0589505   Max.   : 0.0503227
     NIKKEI              BOVESPA                 EU
 Min.   :-0.0504476   Min.   :-0.0538495   Min.   :-0.0488168
 1st Qu.:-0.0074072   1st Qu.:-0.0072146   1st Qu.:-0.0059518
 Median : 0.0000000   Median : 0.0002790   Median : 0.0001958
 Mean   : 0.0003077   Mean   : 0.0009353   Mean   : 0.0004706
 3rd Qu.: 0.0078821   3rd Qu.: 0.0088808   3rd Qu.: 0.0077915
 Max.   : 0.0612293   Max.   : 0.0637915   Max.   : 0.0670425
       EM
 Min.   :-0.0385645
 1st Qu.:-0.0049114
 Median : 0.0010772
 Mean   : 0.0009359
 3rd Qu.: 0.0064226
 Max.   : 0.0478045
```

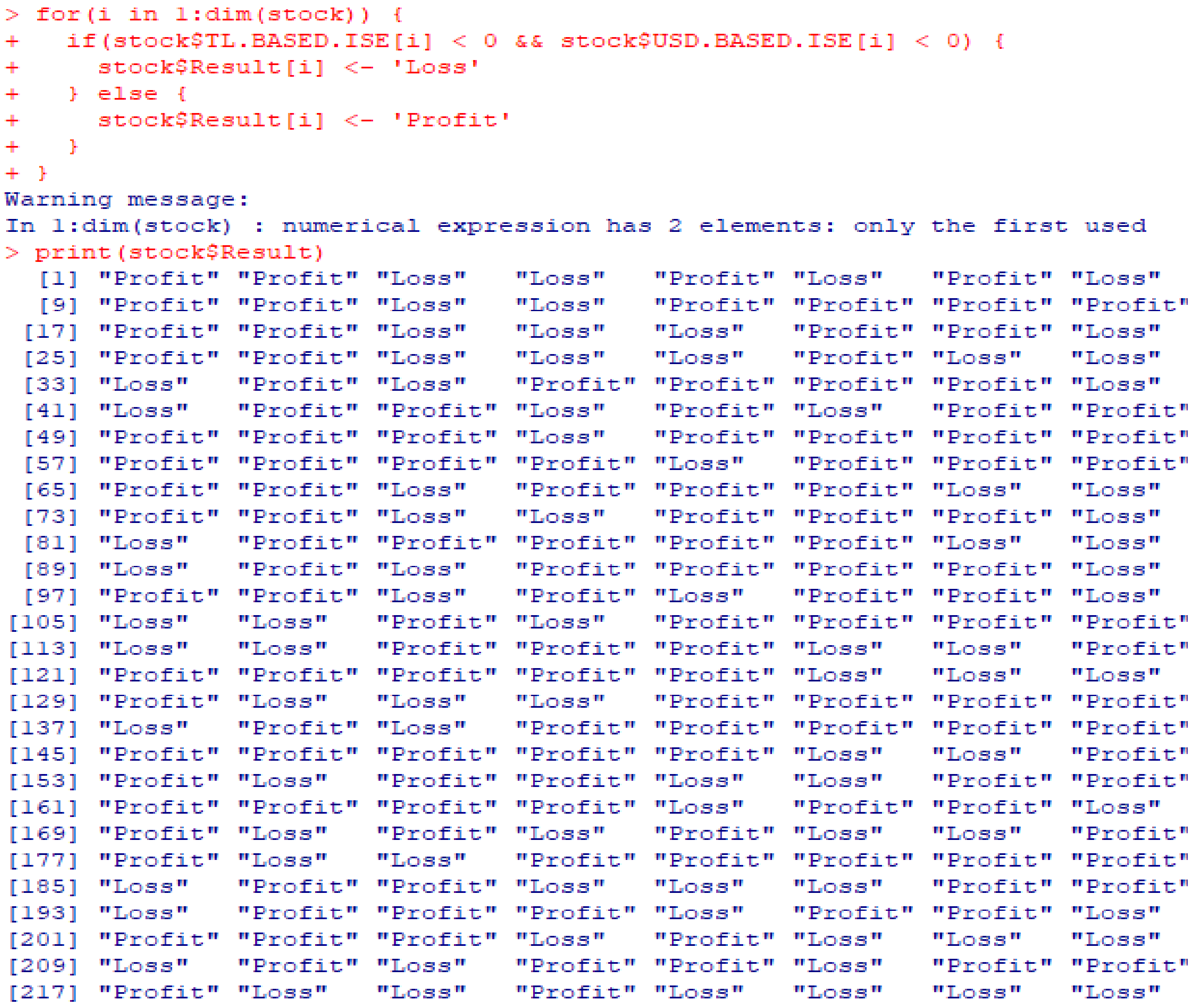
DATA MANIPULATION:



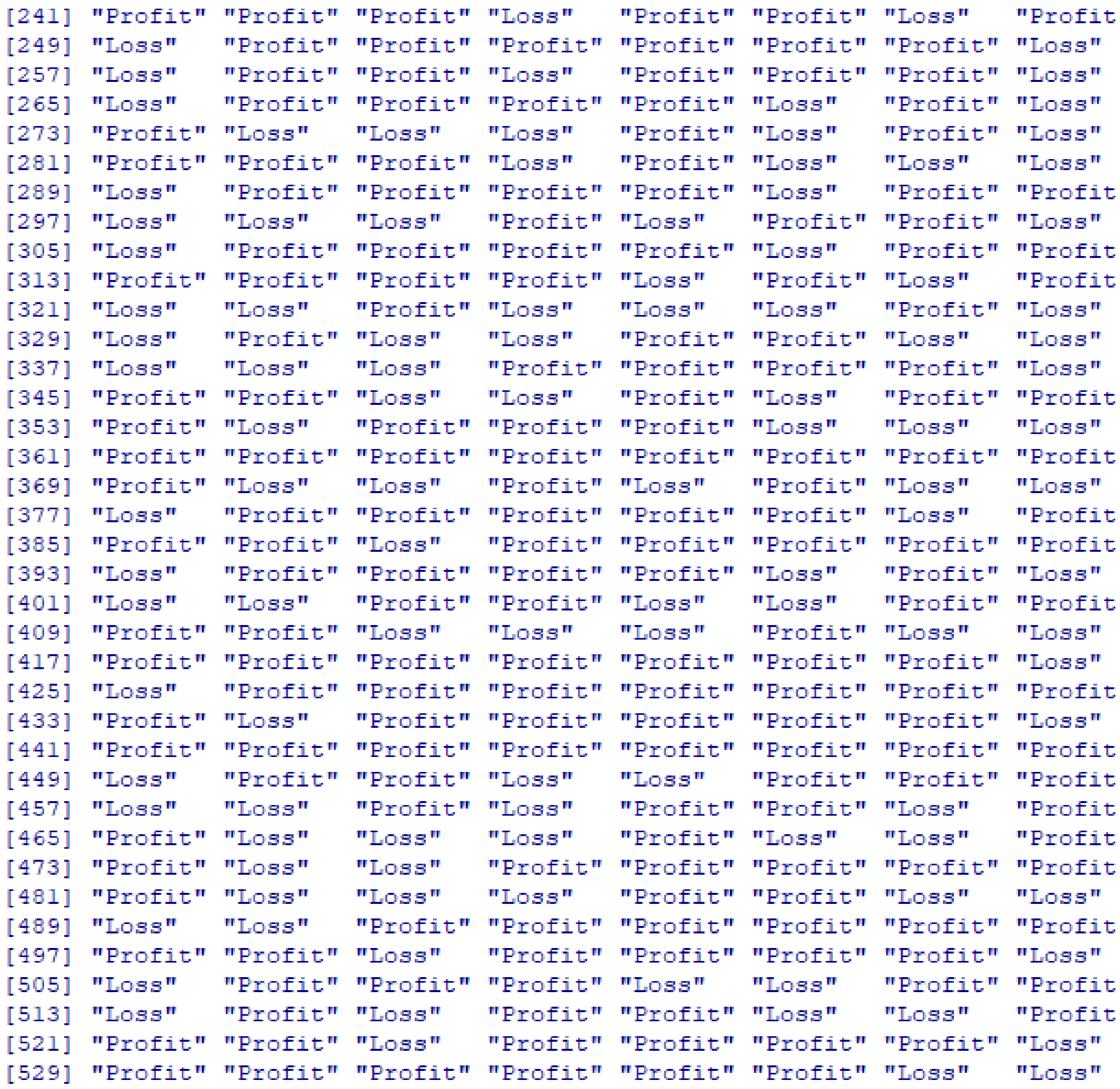
```
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 4.0.5
```



```
> max.exchange <- filter(stock, TL.BASED.ISE > 0 & USD.BASED.ISE > 0)
> print(head(max.exchange))
       date TL.BASED.ISE USD.BASED.ISE           SP          DAX         FTSE
1  5-Jan-09  0.035753708   0.038376187 -0.004679315  0.002193419  0.003894376
2  6-Jan-09  0.025425873   0.031812743  0.007786738  0.008455341  0.012865611
3  9-Jan-09  0.009859905   0.009658112 -0.021533208 -0.019872754 -0.012709717
4 16-Jan-09  0.022037345   0.032278032  0.007533126  0.006790780  0.006289177
5 21-Jan-09  0.000864697   0.001529430  0.042572033  0.005011186 -0.007728867
6 26-Jan-09  0.046831302   0.061708176  0.005537856  0.034786791  0.037891115
        NIKKEI      BOVESPA           EU           EM Result
1  0.000000000  0.031190229  0.012698039  0.028524462 Profit
2  0.004162452  0.018919580  0.011340652  0.008772644 Profit
3 -0.004473502 -0.009763880 -0.010988634 -0.007802212 Profit
4  0.025453186  0.004866686  0.008560858  0.010916893 Profit
5 -0.020561328  0.033532127 -0.003338563 -0.005092242 Profit
6 -0.008181598  0.009838156  0.032799581  0.010319685 Profit
```

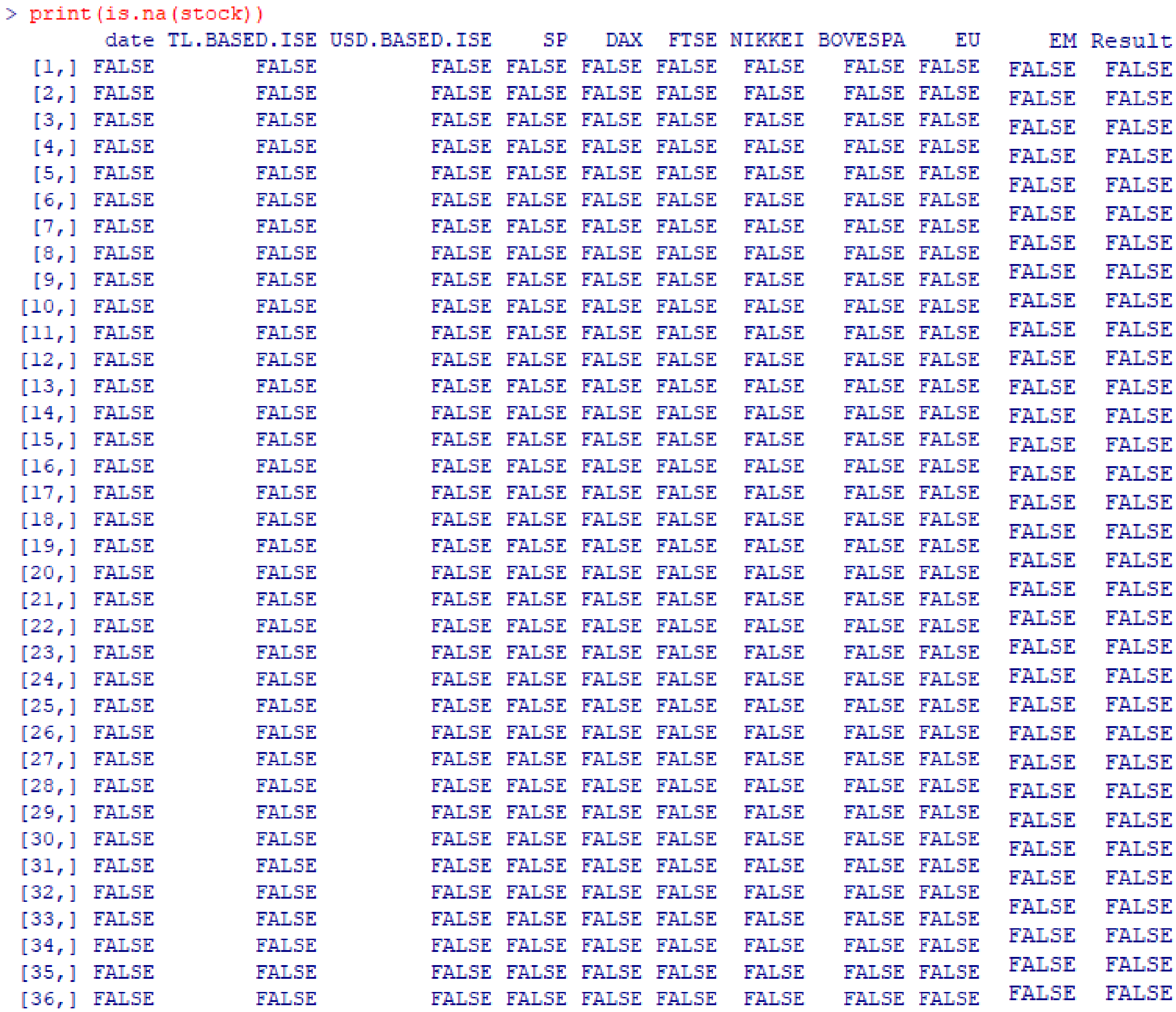


```
> for(i in 1:dim(stock)) {
+   if(stock$TL.BASED.ISE[i] < 0 && stock$USD.BASED.ISE[i] < 0) {
+     stock$Result[i] <- 'Loss'
+   } else {
+     stock$Result[i] <- 'Profit'
+   }
+ }
Warning message:
In 1:dim(stock) : numerical expression has 2 elements: only the first used
> print(stock$Result)
  [1] "Profit" "Profit" "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Loss"  
  [9] "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit"
 [17] "Profit" "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Profit" "Loss"  
 [25] "Profit" "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Loss"   "Loss"  
 [33] "Loss"   "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"  
 [41] "Loss"   "Profit" "Profit" "Loss"   "Profit" "Loss"   "Profit" "Profit"
 [49] "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit"
 [57] "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit"
 [65] "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Loss"   "Loss"  
 [73] "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit" "Profit" "Loss"  
 [81] "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"   "Loss"  
 [89] "Loss"   "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"  
 [97] "Profit" "Profit" "Loss"   "Profit" "Loss"   "Profit" "Profit" "Loss"  
[105] "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit"
[113] "Loss"   "Loss"   "Profit" "Profit" "Profit" "Loss"   "Loss"   "Profit"
[121] "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"   "Loss"   "Loss"  
[129] "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit"
[137] "Loss"   "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit"
[145] "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"   "Loss"   "Profit"
[153] "Profit" "Loss"   "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit"
[161] "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit" "Loss"  
[169] "Profit" "Loss"   "Profit" "Loss"   "Profit" "Loss"   "Loss"   "Profit"
[177] "Profit" "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit"
[185] "Loss"   "Profit" "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Profit"
[193] "Loss"   "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit" "Loss"  
[201] "Profit" "Profit" "Profit" "Loss"   "Profit" "Loss"   "Loss"   "Loss"  
[209] "Loss"   "Profit" "Loss"   "Profit" "Profit" "Loss"   "Profit" "Profit"
[217] "Profit" "Loss"   "Loss"   "Profit" "Loss"   "Loss"   "Loss"   "Loss"  
```



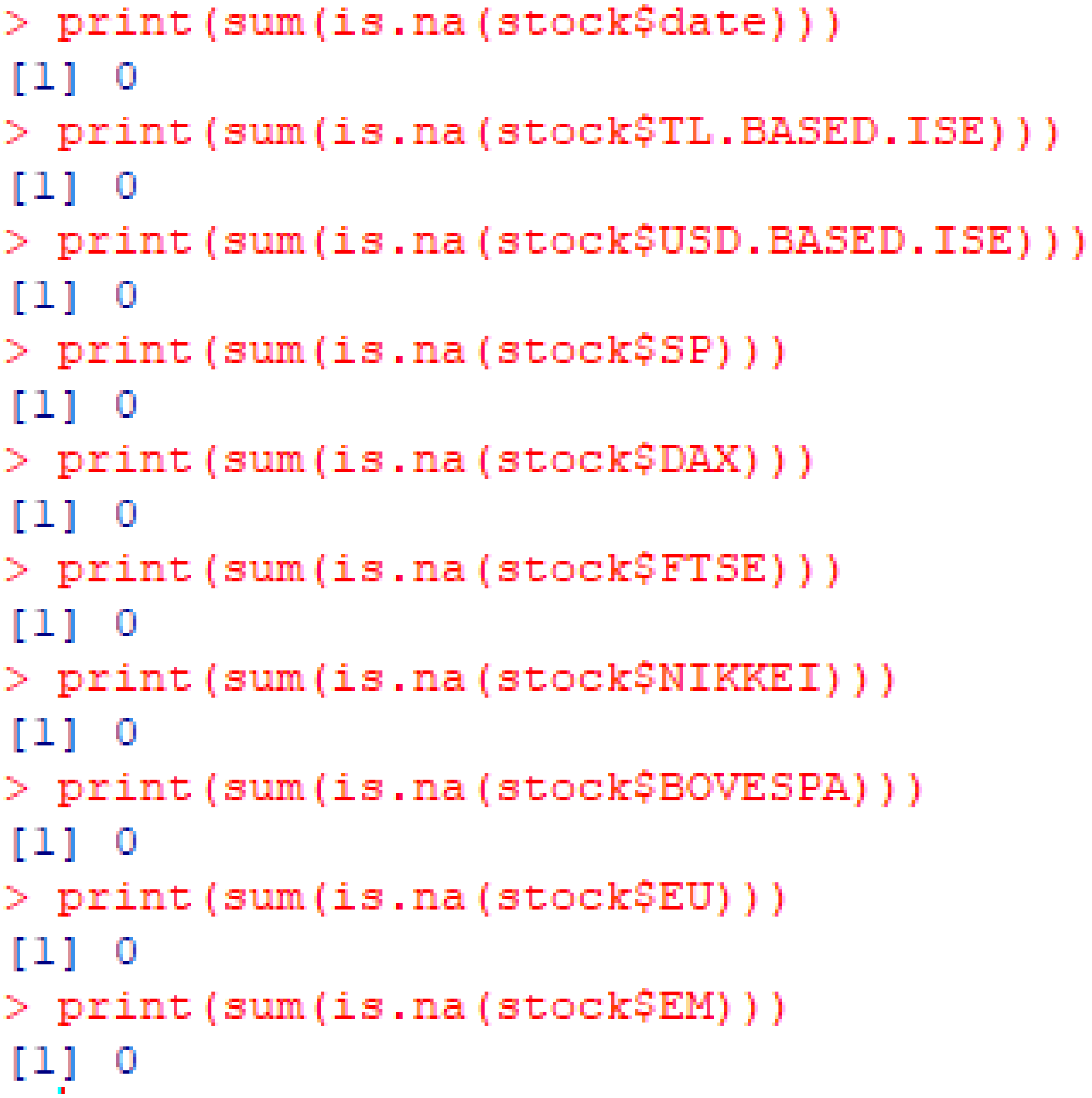
```
[241] "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit" "Loss"   "Profit
[249] "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"
[257] "Loss"   "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Loss"
[265] "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Loss"
[273] "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Loss"
[281] "Profit" "Profit" "Profit" "Loss"   "Profit" "Loss"   "Loss"   "Loss"
[289] "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit
[297] "Loss"   "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Profit" "Loss"
[305] "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Profit
[313] "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Loss"   "Profit
[321] "Loss"   "Loss"   "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Loss"
[329] "Loss"   "Profit" "Loss"   "Loss"   "Profit" "Profit" "Loss"   "Loss"
[337] "Loss"   "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"
[345] "Profit" "Profit" "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Profit
[353] "Profit" "Loss"   "Profit" "Profit" "Profit" "Loss"   "Loss"   "Loss"
[361] "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit
[369] "Profit" "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Loss"   "Loss"
[377] "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit
[385] "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit
[393] "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"   "Profit" "Loss"
[401] "Loss"   "Loss"   "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit
[409] "Profit" "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Loss"   "Loss"
[417] "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"
[425] "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit
[433] "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"
[441] "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Profit
[449] "Loss"   "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit" "Profit
[457] "Loss"   "Loss"   "Profit" "Loss"   "Profit" "Profit" "Loss"   "Profit
[465] "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Loss"   "Loss"   "Profit
[473] "Profit" "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit
[481] "Profit" "Loss"   "Loss"   "Loss"   "Profit" "Profit" "Loss"   "Loss"
[489] "Loss"   "Loss"   "Profit" "Profit" "Profit" "Profit" "Profit" "Profit
[497] "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"
[505] "Loss"   "Profit" "Profit" "Profit" "Loss"   "Loss"   "Profit" "Profit
[513] "Loss"   "Profit" "Loss"   "Profit" "Profit" "Loss"   "Loss"   "Profit
[521] "Profit" "Profit" "Loss"   "Profit" "Profit" "Profit" "Profit" "Loss"
[529] "Profit" "Profit" "Profit" "Profit" "Profit" "Profit" "Loss"   "Loss"
```

DATA MANIPULATION



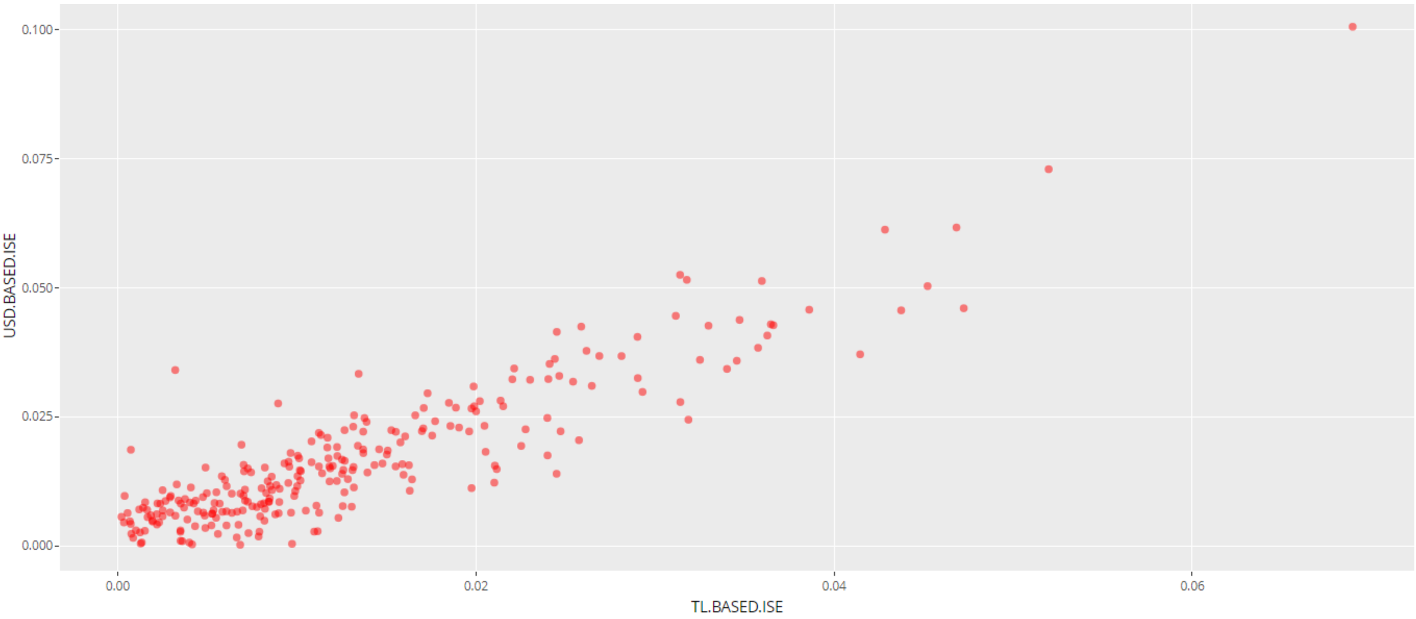
```
> print(is.na(stock))
       date TL.BASED.ISE USD.BASED.ISE    SP   DAX  FTSE NIKKEI BOVESPA    EU    EM Result
 [1,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [2,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [3,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [4,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [5,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [6,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [7,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [8,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
 [9,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[10,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[11,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[12,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[13,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[14,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[15,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[16,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[17,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[18,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[19,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[20,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[21,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[22,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[23,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[24,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[25,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[26,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[27,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[28,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[29,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[30,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[31,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[32,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[33,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[34,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[35,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
[36,] FALSE        FALSE         FALSE FALSE FALSE FALSE  FALSE   FALSE FALSE FALSE  FALSE
```

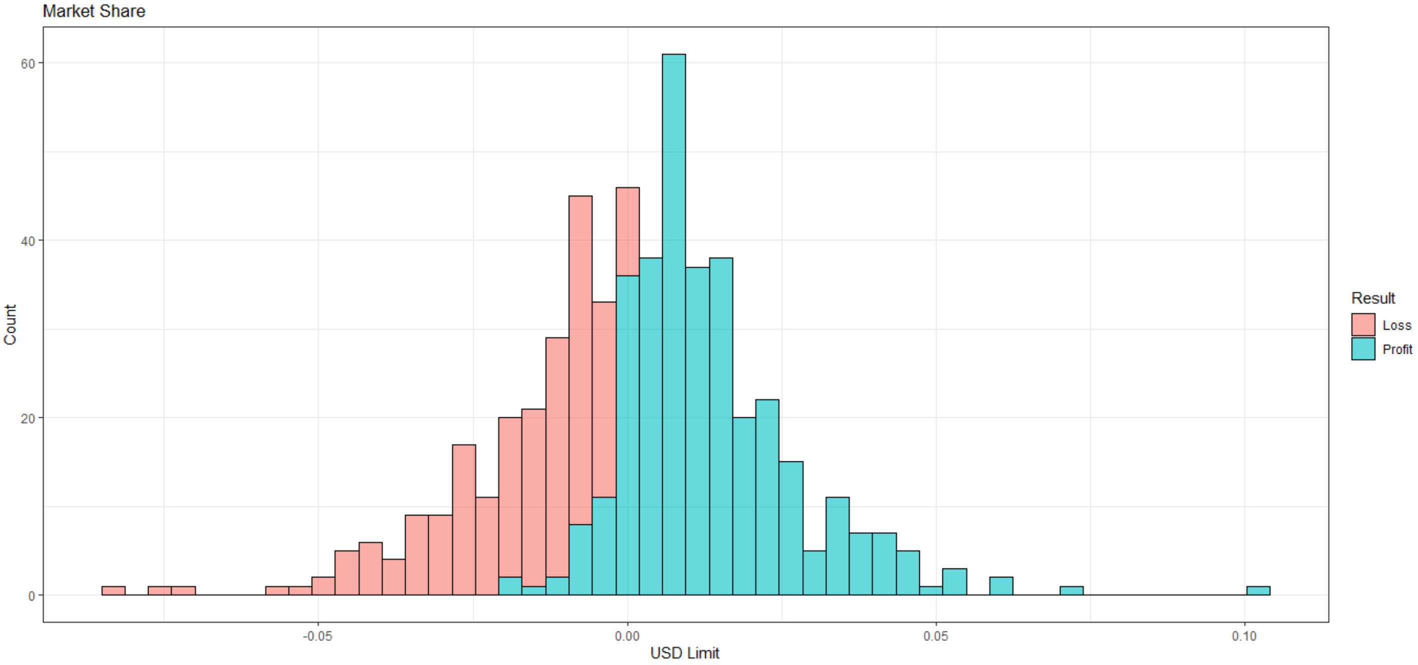
This continues for the rest of the 536 values

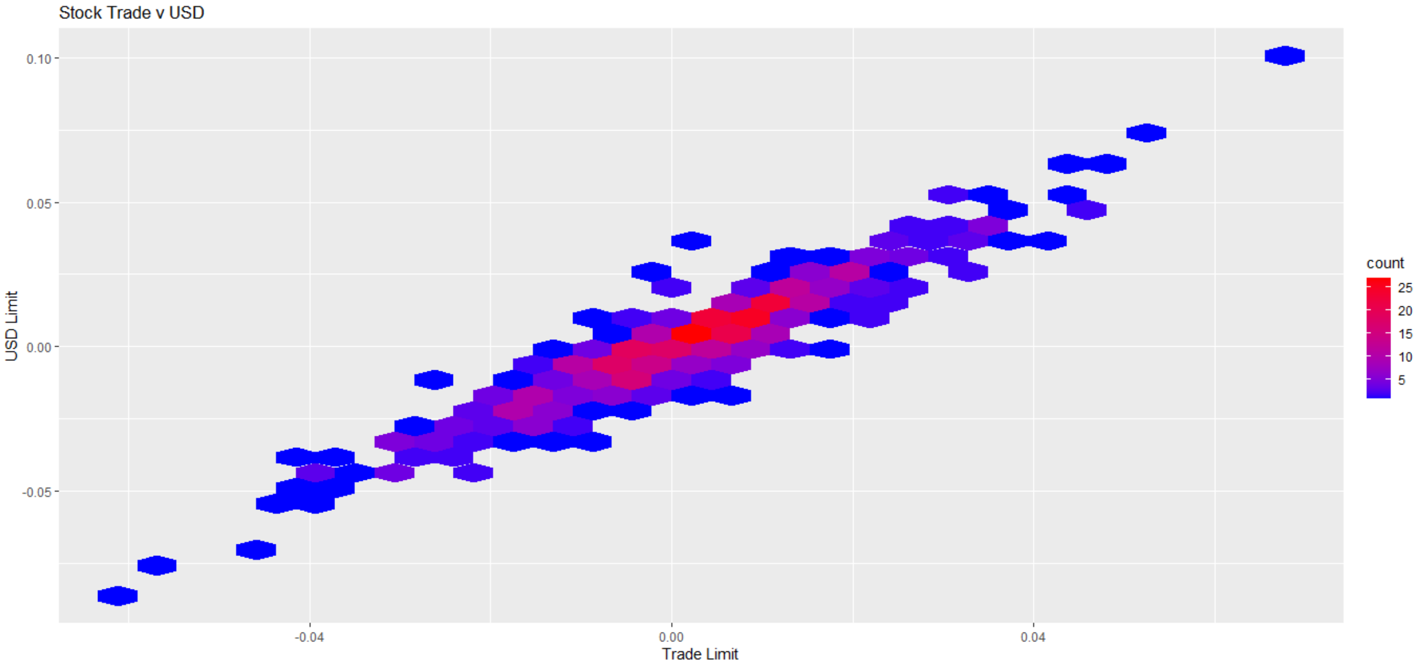


```
> print(sum(is.na(stock$date)))
[1] 0
> print(sum(is.na(stock$TL.BASED.ISE)))
[1] 0
> print(sum(is.na(stock$USD.BASED.ISE)))
[1] 0
> print(sum(is.na(stock$SP)))
[1] 0
> print(sum(is.na(stock$DAX)))
[1] 0
> print(sum(is.na(stock$FTSE)))
[1] 0
> print(sum(is.na(stock$NIKKEI)))
[1] 0
> print(sum(is.na(stock$BOVESPA)))
[1] 0
> print(sum(is.na(stock$EU)))
[1] 0
> print(sum(is.na(stock$EM)))
[1] 0
```

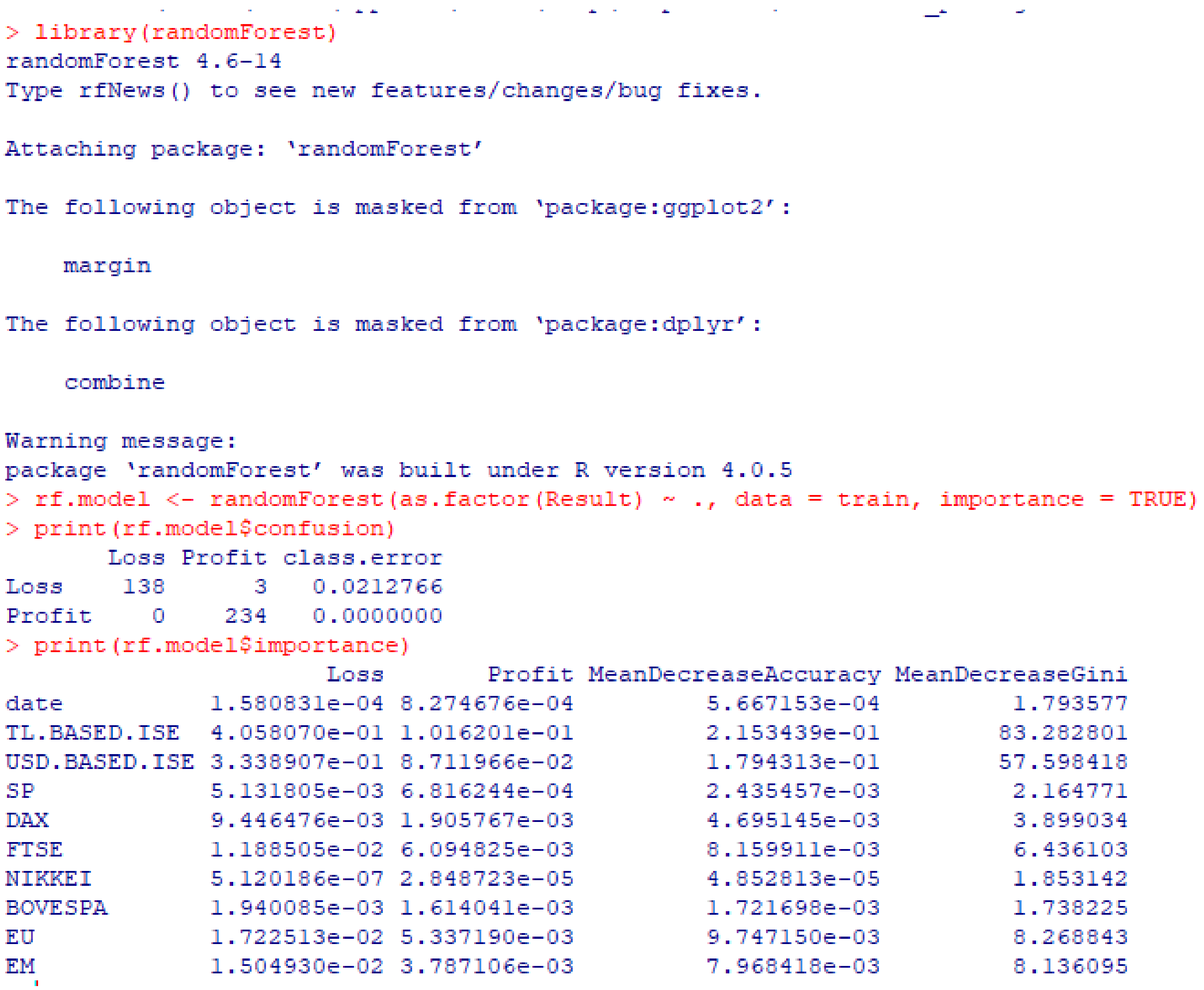
DATA VISUALIZATION


0.100
0.075
0.050
0.025
0.000
USD.BASED.ISE
0.00
0.02
0.04
0.06
TL.BASED.ISE


Market Share
60
40
20
0
Count
-0.05
0.00
0.05
0.10
USD Limit
Result
Loss
Profit

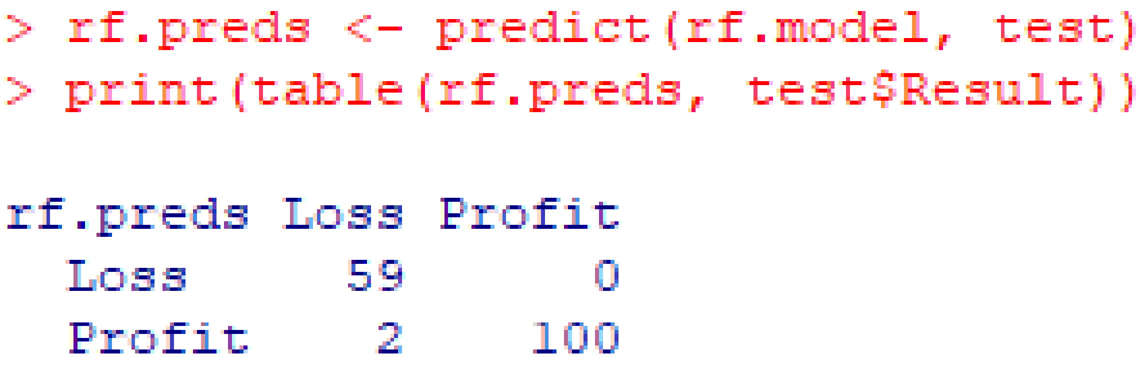

Stock Trade v USD
USD Limit
0.10
0.05
0.00
-0.05
Trade Limit
-0.04
0.00
0.04
count
25
20
15
10
5

TRAIN MODEL



```
> library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

The following object is masked from 'package:dplyr':

    combine

Warning message:
package 'randomForest' was built under R version 4.0.5
> rf.model <- randomForest(as.factor(Result) ~ ., data = train, importance = TRUE)
> print(rf.model$confusion)
       Loss Profit class.error
Loss    138      3   0.0212766
Profit    0    234   0.0000000
> print(rf.model$importance)
                      Loss       Profit MeanDecreaseAccuracy MeanDecreaseGini
date          1.580831e-04 8.274676e-04         5.667153e-04         1.793577
TL.BASED.ISE  4.058070e-01 1.016201e-01         2.153439e-01        83.282801
USD.BASED.ISE 3.338907e-01 8.711966e-02         1.794313e-01        57.598418
SP            5.131805e-03 6.816244e-04         2.435457e-03         2.164771
DAX           9.446476e-03 1.905767e-03         4.695145e-03         3.899034
FTSE          1.188505e-02 6.094825e-03         8.159911e-03         6.436103
NIKKEI        5.120186e-07 2.848723e-05         4.852813e-05         1.853142
BOVESPA       1.940085e-03 1.614041e-03         1.721698e-03         1.738225
EU            1.722513e-02 5.337190e-03         9.747150e-03         8.268843
EM            1.504930e-02 3.787106e-03         7.968418e-03         8.136095
```

PREDICTIONS



```
> rf.preds <- predict(rf.model, test)
> print(table(rf.preds, test$Result))

rf.preds Loss Profit
  Loss     59      0
  Profit    2    100
```

Result and Analysis:

Random Forest is a great algorithm, for both classification and regression problems, to produce a predictive model. Its default hyperparameters already return great results and the system is great at avoiding overfitting. Moreover, it is a pretty good indicator of the importance it assigns to your features.

From this the dataset taken we find the profit respective to various countries. We from the last confusion matrix get the conclusion of profit-profit percentage to be 100%.

We analyze and observe that random forest is the best algorithm for classifying giving the best result out of all the trees it finds, the only problem we face is the since the output would be large it takes a lot of time thus increasing the computation time.

Conclusion:

The Random Forest algorithm is one of the most popular machine learning algorithms that is used for both classification and regression. The ability to perform both tasks makes it unique, and enhances its wide-spread usage across a myriad of applications. It also assures high accuracy most of the time, making it one of the most sought-after classification algorithms. Random Forests are comprised of Decision Trees. The more trees it has, the more sophisticated the algorithm is. It selects the best result out of the votes that are pooled by the trees, making it robust.

THANK YOU!!