XGB Stock Forecast

Progetto 'Tree Based Models'

Studente: Davide Zicca

Professoressa: Anna Gottard

UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI ECONOMIA, STATISTICA E FINANZA "Giovanni Anania" - DESF

Dipartimento di Economia, Statistica e Finanza
Università della Calabria (UNICAL)

15 Maggio, 2021

Motivazione

L'obiettivo di questo progetto è quello di utilizzare *Tree Based* Models per predire l'andamento (Up-tick vs Down-tick) del prezzo giornaliero delle stock:

- Amazon ("AMZN")
- Apple ("AAPL") [1]

L'idea è quindi quella di assegnare valore 1 in caso di Up-movement e valore 0 per il Down-movement. Essendo un problema di classificazione binaria utilizzo la libreria, presente in *R Studio*, **xgboost**.

[1] Nelle prossime slide verrà mostrato il codice che fa riferimento allo script contenente la serie storica del prezzo delle azioni di Apple. Tuttavia, lo stesso procedimento è stato seguito per studiare anche l'andamento del prezzo azionario di Amazon.

eXtreme Gradient Boosting (**XGBoost**) model

Di importanza cruciale per questo progetto è una breve introduzione al modello oggetto di studio. L'eXtreme Gradient Boosting (**XGBoost**) model (Chen and Guestrin, 2016), è un approccio attraverso il quale nuovi modelli vengono creati (con la capacità di prevedere gli errori di modelli precedenti) e poi vengono uniti al fine di raggiungere la previsione finale.

Il Boosting è una tecnica ensemble nei quali nuovi modelli vengono aggregati per correggere gli errori dei modelli precedenti. I modelli vengono aggiunti sequenzialmente fino a che nessun miglioramento può essere raggiunto. Esso usa il *tree ensemble model* che è un insieme di alberi di classificazione e regressione (*CART models*). L'approccio ensemble viene usato perchè, in alcuni casi, un singolo CART non ha forte potere predittivo. Usando un insieme di *CART* (un *tree ensemble model*) è possibile considerare la somma delle previsioni di più alberi predittivi.

La *objective function* dell'eXtreme Gradient Boosting (**XGBoost**) model è:

eXtreme Gradient Boosting (**XGBoost**) model (cont.)

$$Obj = L + \Omega$$

Dove, L è la *loss function* che controlla il potere predittivo, e Ω è la componente di *regolarizzazione* che controlla la complessità e l'overfitting del modello. La *loss function* (L), che deve essere ottimizzata, è rappresentata dal *Root Mean Squared Error* per la regressione, *Log-loss* per la classificazione binaria, o *mlogloss* per la classificazione multi-classe. La componente di *regolarizzazione* Ω dipende dal numero di foglie e dallo score previsionale assegnato alle foglie nel *tree ensemble model*.

Viene chiamato *gradient boosting* perchè usa un algoritmo *gradient descent* per minimizzare la perdita quando si aggiungo nuovi modelli. L'algoritmo di *gradient boosting* (M. et al., 2020) supporta sia la regressione che la classificazione per i problemi di modellizzazione previsionali (Mottaghi and Farhangdoost, 2021).

Data Preparation: importazione e creazione indicatori

Sono stati collezionati i prezzi azionari delle stock di Apple dal 1° Gennaio 2015. Da questi valori sono stati calcolati alcuni tra i più utilizzati indici azionari:

- Il Relative Strength Index (RSI) è un *momentum indicator* largamente utilizzato nell'analisi tecnica e che misura la magnitudo dei più cambiamenti di prezzo per valutare fenomeni di eccessiva vendita/acquisti di azioni o altri asset.
- L'Average Directional Index (ADX) serve per determinare quando una stock è in fase di forte trading sul mercato azionario. Si basa sul calcolo del Moving Average [2] del range del prezzo, fissato un intervallo di tempo di interesse.
- Il SAR parabolico è un indicatore tecnico che determina la direzione del prezzo di un determinato asset: è usato anche come alert dell'imminente cambiamento di direzione del prezzo dell'asset.

[2] Si veda anche il *MACD* (Moving Average Convergence DIvergence) che è un altro indicatore fortemente utilizzato nell'analisi tecnica.

Tree Based Models - Progetto Xgb Stock Forecast

Davide Zicca

9/5/2021

Data Preparation

```
setwd("C:/Davide/MASTER IN DATA SCIENCE/Materiale del Master/Tree Based Models/PROGETTO/XGB stock foreca
library(quantmod)
```

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

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

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

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

```
## Loading required package: TTR
```

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

```
library(TTR)
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.0.4
```

```
library(readr)

getSymbols("AAPL", from = "2015-01-01")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "AAPL"
```

```
# AAPL
write.csv(AAPL, file = "AAPL.csv")
AAPL <- read_csv("AAPL.csv", col_types =  cols(
  X1 = col_number(),
  AAPL.Open = col_number(),
```

```
  AAPL.High = col_number(),
  AAPL.Low = col_number(),
  AAPL.Close = col_number(),
  AAPL.Volume = col_number(),
  AAPL.Adjusted = col_number()
))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
symbol = "AAPL"
fileName = paste(getwd(),"/",symbol,".csv",sep="") ;
df = as.data.frame(AAPL)
colnames(df) = c("Date","Open","High","Low","Close",  "Volume", "Adjusted")
head(df)
```

```
##   Date    Open    High     Low   Close    Volume Adjusted
## 1    1 27.8475 27.8600 26.8375 27.3325 212818400 24.81924
## 2    2 27.0725 27.1625 26.3525 26.5625 257142000 24.12005
## 3    3 26.6350 26.8575 26.1575 26.5650 263188400 24.12232
## 4    4 26.8000 27.0500 26.6750 26.9375 160423600 24.46056
## 5    5 27.3075 28.0375 27.1750 27.9725 237458000 25.40040
## 6    6 28.1675 28.3125 27.5525 28.0025 214798000 25.42763
```

```
rsi = RSI(df$Close, n=14, maType="WMA")
adx = data.frame(ADX(df[,c("High","Low","Close")]))
sar = SAR(df[,c("High","Low")], accel = c(0.02, 0.2))
trend = df$Close - sar
rsi = c(NA,head(rsi,-1))
adx$ADX = c(NA,head(adx$ADX,-1))
trend = c(NA,head(trend,-1))
# Create the target variable
price = df$Close-df$Open
class = ifelse(price > 0,1,0)
# Create a Matrix
model_df = data.frame(class,rsi,adx$ADX,trend)
model = matrix(c(class,rsi,adx$ADX,trend), nrow=length(class))
model = na.omit(model) # Drop NA
colnames(model) = c("class","rsi","adx","trend")
```

Predisposizione del modello: training e CV

```
# Split data into train and test sets
train_size = 2/3
breakpoint = nrow(model) * train_size
training_data = model[1:breakpoint,]
test_data = model[(breakpoint+1):nrow(model),]
# Split data training and test data into X and Y
X_train = training_data[,2:4] ; Y_train = training_data[,1]
class(X_train)[1]; class(Y_train)
```

```
## [1] "matrix"
```

```
## [1] "numeric"
```

Data Preparation: Creazione matrice per xgboost

Prima di procedere oltre, è stato creato un *lag* negli indicatori precedentemente calcolati per evitare il *look-ahead bias* [3].

Le *input features* sono state cosi create.

Si noti che tale modello è una semplificazione di un modello di *stock forecast*. Altri indicatori e migliori assunzioni sono necessarie per migliorare le performance predittive del modello. Come si vedrà le performance di tale modello si attestano su percentuali non soddisfacenti per basare una solida strategia di trading azionario. Le input features e la variabile target create precedentemente sono unite per creare una singola matrice: al fine di seguire la struttura matriciale imposta dal modello XGBoost. Il dataset viene poi diviso in training e test dataset. La funzione **xgboost** contenuta in R viene usata per addestrare il modello.

[3] Tale fenomeno si manifesta quando si usano dati, all'interno di una simulazione o di un'analisi, che non sarebbero altrimenti disponibili durante il periodo di interesse. Questo può portare a risultati inappropriati. Talvolta, può anche portare i risultati della simulazione verso quelli desiderati dallo studio: dando *overconfidence* verso i risultati ottenuti dal modello di forecast. Infine, il *look-ahead bias* può alterare le strategie di trading, quando queste ultime fanno troppo riferimento alle serie storiche.

```
X_test = test_data[,2:4] ; Y_test = test_data[,1]
class(X_test)[1]; class(Y_test)
```

```
## [1] "matrix"
```

```
## [1] "numeric"
```

```
# Train the xgboost model using the "xgboost" function
dtrain = xgb.DMatrix(data = X_train, label = Y_train)
xgModel = xgboost(data = dtrain, nround = 5, objective = "binary:logistic",
                  eval_metric= "error")
```

```
## [1]  train-error:0.374761
## [2]  train-error:0.301147
## [3]  train-error:0.267686
## [4]  train-error:0.239006
## [5]  train-error:0.224665
```

```
# Using cross validation
dtrain = xgb.DMatrix(data = X_train, label = Y_train)
cv = xgb.cv(data = dtrain, nround = 10, nfold = 5, objective = "binary:logistic",
            eval_metric= "error")
```

```
## [1]  train-error:0.331741+0.018158   test-error:0.523864+0.023829
## [2]  train-error:0.292782+0.013763   test-error:0.515284+0.012411
## [3]  train-error:0.272709+0.014816   test-error:0.519121+0.013701
## [4]  train-error:0.254546+0.016846   test-error:0.524858+0.010658
## [5]  train-error:0.241640+0.018721   test-error:0.531556+0.009635
## [6]  train-error:0.227778+0.023305   test-error:0.523887+0.021587
## [7]  train-error:0.220848+0.021945   test-error:0.517188+0.026885
## [8]  train-error:0.210809+0.018373   test-error:0.520078+0.028453
## [9]  train-error:0.198857+0.019596   test-error:0.504780+0.037977
## [10] train-error:0.195271+0.019460   test-error:0.516250+0.038586
```

Forecast, Performance del Modello e Visualizzazione

```
# Make the predictions on the test data
preds = predict(xgModel, X_test)
# Determine the size of the prediction vector
print(length(preds))
```

```
## [1] 523
```

```
# Limit display of predictions to the first 6
print(head(preds))
```

```
## [1] 0.5860648 0.4273933 0.5050582 0.5050582 0.5050582 0.5050582
```

```
prediction = as.numeric(preds > 0.5)
print(head(prediction))
```

```
## [1] 1 0 1 1 1 1
```

```
# Measuring model performance
error_value = mean(as.numeric(preds > 0.5) != Y_test)
print(paste("test-error=", error_value))
```

```
## [1] "test-error= 0.466539196940727"
```

Modello: Procedura

Tale funzione mostra il *gradient* e *second order gradient*. L'output è l'errore di classificazione sul training dataset.

In aggiunta viene usata la funzione di *cross-validation* attraverso il comando: *xgb.cv*. In questo caso, il campione originale è partizionato randomicamente in *nfold* sub-campioni di ugual misura. Tra i *nfold* sub-campioni, un unico sub-campione viene mantenuto come *validation data* per testare il modello, e i rimanenti (*nfold - 1*) sub-campioni vengono usati come *training data*. Il processo di cross-validation viene ripetuto *nrounds* volte, con ognuno dei *nfold* sub-campioni usati uno alla volta come *validation data*. La funzione **xgb.cv** mostra una *data.table* contenente i risultati di cross validation.

Modello: Interpretazioni risultati

Al fine di fare previsioni sui dati non osservati (ovvero sul test dataset), applichiamo il modello fittato sul test dataset. Al fine di leggere i dati ottenuti per la nostra classificazione binaria (0= Down-movement; 1= Up-movement) effettuiamo una semplice trasformazione. Compariamo il risultato ottenuto con una *threshold* fissata pari a 0.5. Come metrica per testare le performance del nostro modello, viene usato l'errore medio. Compariamo lo score ottenuto con la *threshold* pari a 0.5.

Regola decisionale: se lo score è più piccolo di 0.5, allora otteniamo un valore pari a 0. Se tale valore non è uguale al valore effettivo dal test dataset, allora è interpretato come risultato erroneo. Tutte le previsioni sono state comparate con i rispettivi valori del *Y test* (con il rispettivo errore medio). L'alternativa è quella di creare una matrice di confusione per misurare le performance del modello. Le *important features* del modello sono estratte con la funzione **xgb.importance**.

Bibliografia

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: a Scalable Tree Boosting System". In: *CoRR* abs/1603.02754. arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

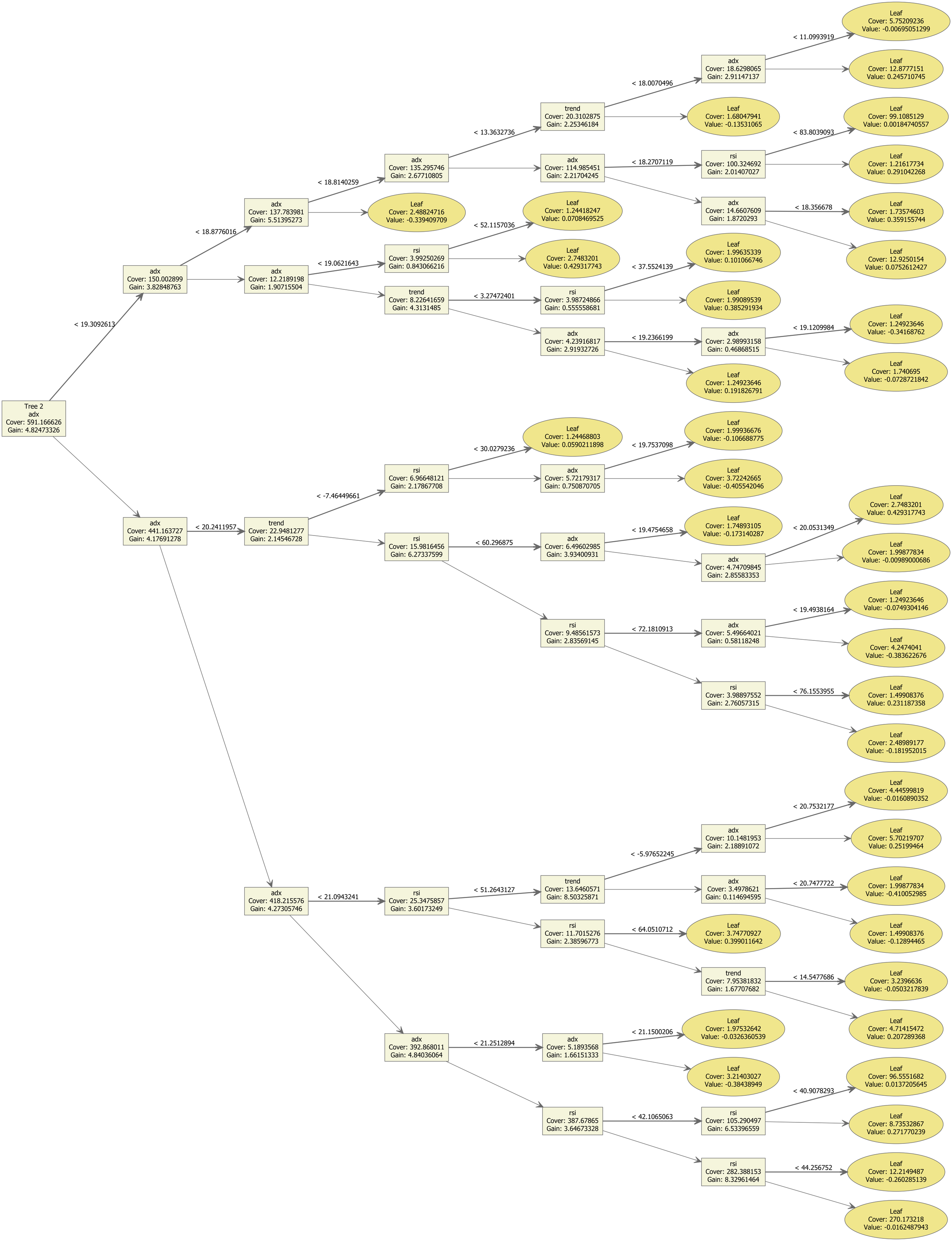
M., Nabipour et al. (2020). "Deep Learning for Stock Market Prediction". In: *Entropy* 22. URL: http://dx.doi.org/10.3390/e22080840.

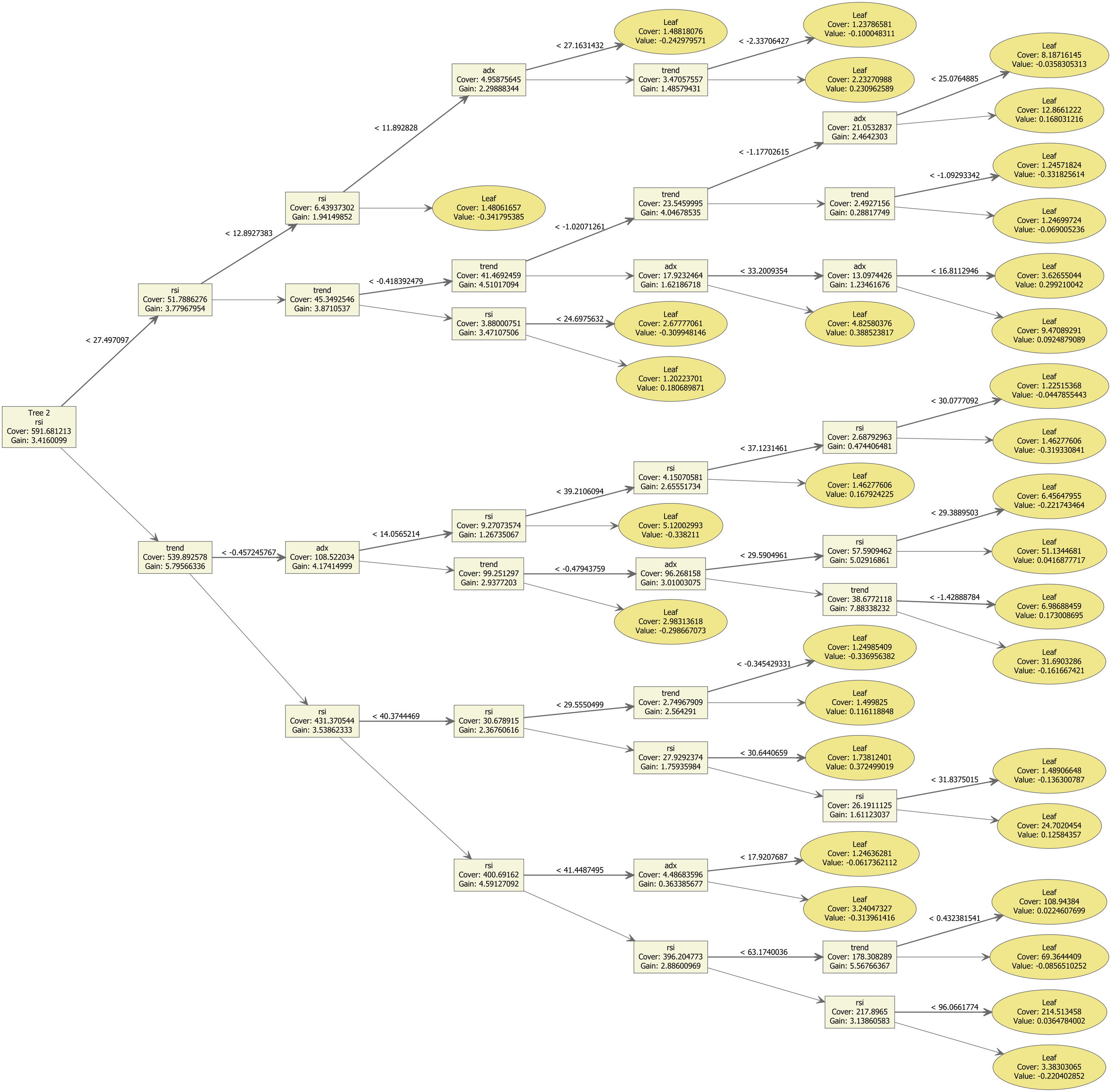
Mottaghi, Navid and Sara Farhangdoost (2021). "Stock Price Forecasting in Presence of Covid-19 Pandemic and Evaluating Performances of Machine Learning Models for Time-Series Forecasting". In: URL: https://arxiv.org/abs/2105.02785.

```r
# View feature importance from the learnt model
importance_matrix = xgb.importance(model = xgModel)
print(importance_matrix)
```

```
##    Feature      Gain     Cover Frequency
## 1:     rsi 0.3970108 0.3238952 0.3860759
## 2:   trend 0.3545263 0.4393252 0.3481013
## 3:     adx 0.2484628 0.2367796 0.2658228
```

```r
# View the trees from a model
#xgb.plot.tree(model = xgModel)
# View only the first tree in the XGBoost model
#xgb.plot.tree(model = xgModel, trees = 1)
#xgb.plot.tree(model = xgModel, trees = 3, render = F, plot_height = 2000, plot_width = 1500)
```


Tree 2
adx
Cover: 591.166626
Gain: 4.82473326
< 19.3092613
adx
Cover: 150.002899
Gain: 3.82848763
< 18.8776016
adx
Cover: 137.783981
Gain: 5.51395273
< 18.8140259
adx
Cover: 135.295746
Gain: 2.67710805
< 13.3632736
trend
Cover: 20.3102875
Gain: 2.25346184
< 18.0070496
adx
Cover: 18.6298065
Gain: 2.91147137
< 11.0993919
Leaf
Cover: 5.75209236
Value: -0.00695051299
Leaf
Cover: 12.8777151
Value: 0.245710745
Leaf
Cover: 1.68047941
Value: -0.13531065
adx
Cover: 114.985451
Gain: 2.21704245
< 18.2707119
rsi
Cover: 100.324692
Gain: 2.01407027
< 83.8039093
Leaf
Cover: 99.1085129
Value: 0.00184740557
Leaf
Cover: 1.21617734
Value: 0.291042268
adx
Cover: 14.6607609
Gain: 1.8720293
< 18.356678
Leaf
Cover: 1.73574603
Value: 0.359155744
Leaf
Cover: 12.9250154
Value: 0.0752612427
Leaf
Cover: 2.48824716
Value: -0.339409709
adx
Cover: 12.2189198
Gain: 1.90715504
< 19.0621643
rsi
Cover: 3.99250269
Gain: 0.843066216
< 52.1157036
Leaf
Cover: 1.24418247
Value: 0.0708469525
Leaf
Cover: 2.7483201
Value: 0.429317743
trend
Cover: 8.22641659
Gain: 4.3131485
< 3.27472401
rsi
Cover: 3.98724866
Gain: 0.555558681
< 37.5524139
Leaf
Cover: 1.99635339
Value: 0.101066746
Leaf
Cover: 1.99089539
Value: 0.385291934
adx
Cover: 4.23916817
Gain: 2.91932726
< 19.2366199
adx
Cover: 2.98993158
Gain: 0.46868515
< 19.1209984
Leaf
Cover: 1.24923646
Value: -0.34168762
Leaf
Cover: 1.740695
Value: -0.0728721842
Leaf
Cover: 1.24923646
Value: 0.191826791
adx
Cover: 441.163727
Gain: 4.17691278
< 20.2411957
trend
Cover: 22.9481277
Gain: 2.14546728
< -7.46449661
rsi
Cover: 6.96648121
Gain: 2.17867708
< 30.0279236
Leaf
Cover: 1.24468803
Value: 0.0590211898
adx
Cover: 5.72179317
Gain: 0.750870705
< 19.7537098
Leaf
Cover: 1.99936676
Value: -0.106688775
Leaf
Cover: 3.72242665
Value: -0.405542046
rsi
Cover: 15.9816456
Gain: 6.27337599
< 60.296875
adx
Cover: 6.49602985
Gain: 3.93400931
< 19.4754658
Leaf
Cover: 1.74893105
Value: -0.173140287
adx
Cover: 4.74709845
Gain: 2.85583353
< 20.0531349
Leaf
Cover: 2.7483201
Value: 0.429317743
Leaf
Cover: 1.99877834
Value: -0.00989000686
rsi
Cover: 9.48561573
Gain: 2.83569145
< 72.1810913
adx
Cover: 5.49664021
Gain: 0.58118248
< 19.4938164
Leaf
Cover: 1.24923646
Value: -0.0749304146
Leaf
Cover: 4.2474041
Value: -0.383622676
rsi
Cover: 3.98897552
Gain: 2.76057315
< 76.1553955
Leaf
Cover: 1.49908376
Value: 0.231187358
Leaf
Cover: 2.48989177
Value: -0.181952015
adx
Cover: 418.215576
Gain: 4.27305746
< 21.0943241
rsi
Cover: 25.3475857
Gain: 3.60173249
< 51.2643127
trend
Cover: 13.6460571
Gain: 8.50325871
< -5.97652245
adx
Cover: 10.1481953
Gain: 2.18891072
< 20.7532177
Leaf
Cover: 4.44599819
Value: -0.0160890352
Leaf
Cover: 5.70219707
Value: 0.25199464
adx
Cover: 3.4978621
Gain: 0.114694595
< 20.7477722
Leaf
Cover: 1.99877834
Value: -0.410052985
Leaf
Cover: 1.49908376
Value: -0.12894465
rsi
Cover: 11.7015276
Gain: 2.38596773
< 64.0510712
Leaf
Cover: 3.74770927
Value: 0.399011642
trend
Cover: 7.95381832
Gain: 1.67707682
< 14.5477686
Leaf
Cover: 3.2396636
Value: -0.0503217839
Leaf
Cover: 4.71415472
Value: 0.207289368
adx
Cover: 392.868011
Gain: 4.84036064
< 21.2512894
adx
Cover: 5.1893568
Gain: 1.66151333
< 21.1500206
Leaf
Cover: 1.97532642
Value: -0.0326360539
Leaf
Cover: 3.21403027
Value: -0.38438949
rsi
Cover: 387.67865
Gain: 3.64673328
< 42.1065063
rsi
Cover: 105.290497
Gain: 6.53396559
< 40.9078293
Leaf
Cover: 96.5551682
Value: 0.0137205645
Leaf
Cover: 8.73532867
Value: 0.271770239
rsi
Cover: 282.388153
Gain: 8.32961464
< 44.256752
Leaf
Cover: 12.2149487
Value: -0.260285139
Leaf
Cover: 270.173218
Value: -0.0162487943


Tree 2
rsi
Cover: 591.681213
Gain: 3.4160099
< 27.497097
rsi
Cover: 51.7886276
Gain: 3.77967954
< 12.8927383
rsi
Cover: 6.43937302
Gain: 1.94149852
< 11.892828
adx
Cover: 4.95875645
Gain: 2.29888344
< 27.1631432
Leaf
Cover: 1.48818076
Value: -0.242979571
trend
Cover: 3.47057557
Gain: 1.48579431
< -2.33706427
Leaf
Cover: 1.23786581
Value: -0.100048311
Leaf
Cover: 2.23270988
Value: 0.230962589
Leaf
Cover: 1.48061657
Value: -0.341795385
trend
Cover: 45.3492546
Gain: 3.8710537
< -0.418392479
trend
Cover: 41.4692459
Gain: 4.51017094
< -1.02071261
trend
Cover: 23.5459995
Gain: 4.04678535
< -1.17702615
adx
Cover: 21.0532837
Gain: 2.4642303
< 25.0764885
Leaf
Cover: 8.18716145
Value: -0.0358305313
Leaf
Cover: 12.8661222
Value: 0.168031216
trend
Cover: 2.4927156
Gain: 0.28817749
< -1.09293342
Leaf
Cover: 1.24571824
Value: -0.331825614
Leaf
Cover: 1.24699724
Value: -0.069005236
adx
Cover: 17.9232464
Gain: 1.62186718
< 33.2009354
adx
Cover: 13.0974426
Gain: 1.23461676
< 16.8112946
Leaf
Cover: 3.62655044
Value: 0.299210042
Leaf
Cover: 4.82580376
Value: 0.388523817
Leaf
Cover: 9.47089291
Value: 0.0924879089
rsi
Cover: 3.88000751
Gain: 3.47107506
< 24.6975632
Leaf
Cover: 2.67777061
Value: -0.309948146
Leaf
Cover: 1.20223701
Value: 0.180689871
trend
Cover: 539.892578
Gain: 5.79566336
< -0.457245767
adx
Cover: 108.522034
Gain: 4.17414999
< 14.0565214
rsi
Cover: 9.27073574
Gain: 1.26735067
< 39.2106094
rsi
Cover: 4.15070581
Gain: 2.65551734
< 37.1231461
rsi
Cover: 2.68792963
Gain: 0.474406481
< 30.0777092
Leaf
Cover: 1.22515368
Value: -0.0447855443
Leaf
Cover: 1.46277606
Value: -0.319330841
Leaf
Cover: 1.46277606
Value: 0.167924225
Leaf
Cover: 5.12002993
Value: -0.338211
trend
Cover: 99.251297
Gain: 2.9377203
< -0.47943759
adx
Cover: 96.268158
Gain: 3.01003075
< 29.5904961
rsi
Cover: 57.5909462
Gain: 5.02916861
< 29.3889503
Leaf
Cover: 6.45647955
Value: -0.221743464
Leaf
Cover: 51.1344681
Value: 0.0416877717
trend
Cover: 38.6772118
Gain: 7.88338232
< -1.42888784
Leaf
Cover: 6.98688459
Value: 0.173008695
Leaf
Cover: 31.6903286
Value: -0.161667421
Leaf
Cover: 2.98313618
Value: -0.298667073
rsi
Cover: 431.370544
Gain: 3.53862333
< 40.3744469
rsi
Cover: 30.678915
Gain: 2.36760616
< 29.5550499
trend
Cover: 2.74967909
Gain: 2.564291
< -0.345429331
Leaf
Cover: 1.24985409
Value: -0.336956382
Leaf
Cover: 1.499825
Value: 0.116118848
rsi
Cover: 27.9292374
Gain: 1.75935984
< 30.6440659
Leaf
Cover: 1.73812401
Value: 0.372499019
rsi
Cover: 26.1911125
Gain: 1.61123037
< 31.8375015
Leaf
Cover: 1.48906648
Value: -0.136300787
Leaf
Cover: 24.7020454
Value: 0.12584357
rsi
Cover: 400.69162
Gain: 4.59127092
< 41.4487495
adx
Cover: 4.48683596
Gain: 0.363385677
< 17.9207687
Leaf
Cover: 1.24636281
Value: -0.0617362112
Leaf
Cover: 3.24047327
Value: -0.313961416
rsi
Cover: 396.204773
Gain: 2.88600969
< 63.1740036
trend
Cover: 178.308289
Gain: 5.56766367
< 0.432381541
Leaf
Cover: 108.94384
Value: 0.0224607699
Leaf
Cover: 69.3644409
Value: -0.0856510252
rsi
Cover: 217.8965
Gain: 3.13860583
< 96.0661774
Leaf
Cover: 214.513458
Value: 0.0364784002
Leaf
Cover: 3.38303065
Value: -0.220402852

GRAZIE PER L'ATTENZIONE

```r
table(prediction)
```

```
## prediction
##   0   1
## 203 320
```