Stock trading - Algos evaluated via Accuracy metrics

Davide Zicca

15/5/2021

Data Preparation

```
setwd("C:/Davide/MASTER IN DATA SCIENCE/Materiale del Master/Tree Based Models/PROGETTO/Predictive-Model
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(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(FSelector)
```

```
## Warning: package 'FSelector' was built under R version 4.0.5
```

```r
## Use set.seed function to ensure the results are repeatable
set.seed(5)

## Read the stock and index data
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"
```

```r
# AAPL
write.csv(AAPL, file = "AAPL.csv")

getSymbols("^GSPC", from = "2015-01-01")
```

```
## [1] "^GSPC"
```

```r
# SP500
write.csv(GSPC, file = "SP500.csv")
df_stock = read.csv("AAPL.csv")
df_index = read.csv("SP500.csv")

## Compute the price change for the stock and classify as UP/DOWN
price = df_stock$AAPL.Close-df_stock$AAPL.Open
class = ifelse(price > 0,"UP","DOWN")

## Compute the various technical indicators that will be used
# Force Index Indicator
forceindex = (df_stock$AAPL.Close - df_stock$AAPL.Open) * df_stock$AAPL.Volume ; forceindex = c(NA,head

# Buy & Sell signal Indicators (Williams R% and RSI)
WillR5  = WPR(df_stock[,c("AAPL.High","AAPL.Low","AAPL.Close")], n = 5) ; WillR5 = c(NA,head(WillR5,-1))
WillR10 = WPR(df_stock[,c("AAPL.High","AAPL.Low","AAPL.Close")], n = 10) ; WillR10 = c(NA,head(WillR10,
WillR15 = WPR(df_stock[,c("AAPL.High","AAPL.Low","AAPL.Close")], n = 15) ; WillR15 = c(NA,head(WillR15,

RSI5  = RSI(df_stock$AAPL.Close, n = 5,maType="WMA") ;RSI5 = c(NA,head(RSI5,-1)) ;
RSI10 = RSI(df_stock$AAPL.Close, n = 10,maType="WMA") ;RSI10 = c(NA,head(RSI10,-1)) ;
RSI15 = RSI(df_stock$AAPL.Close, n = 15,maType="WMA") ;RSI15 = c(NA,head(RSI15,-1)) ;

# Price change Indicators (ROC and Momentum)
ROC5 = ROC(df_stock$AAPL.Close, n = 5,type ="discrete")*100 ; ROC5 = c(NA,head(ROC5,-1)) ;
ROC10 = ROC(df_stock$AAPL.Close, n = 10,type ="discrete")*100 ; ROC10 = c(NA,head(ROC10,-1)) ;

MOM5 = momentum(df_stock$AAPL.Close, n = 5, na.pad = TRUE) ; MOM5 = c(NA,head(MOM5,-1)) ;
MOM10 = momentum(df_stock$AAPL.Close, n = 10, na.pad = TRUE) ; MOM10 = c(NA,head(MOM10,-1)) ;

MOM5Indx = momentum(df_index$GSPC.Close, n = 5, na.pad = TRUE) ; MOM5Indx = c(NA,head(MOM5Indx,-1)) ;
MOM10Indx = momentum(df_index$GSPC.Close, n = 10, na.pad = TRUE); MOM10Indx = c(NA,head(MOM10Indx,-1))
```

```
# Volatility signal Indicator (ATR)
ATR5 = ATR(df_stock[,c("AAPL.High","AAPL.Low","AAPL.Close")], n = 5, maType="WMA")[,1] ; ATR5 = c(NA,hea
ATR10 = ATR(df_stock[,c("AAPL.High","AAPL.Low","AAPL.Close")], n = 10, maType="WMA")[,1]; ATR10 = c(NA,

ATR5Indx = ATR(df_index[,c("GSPC.High","GSPC.Low","GSPC.Close")], n = 5, maType="WMA")[,1]; ATR5Indx =
ATR10Indx = ATR(df_index[,c("GSPC.High","GSPC.Low","GSPC.Close")], n = 10, maType="WMA")[,1]; ATR10Indx

## Combining all the Indicators and the Class into one dataframe
dataset = data.frame(class,forceindex,WillR5,WillR10,WillR15,RSI5,RSI10,RSI15,ROC5,
                     ROC10,MOM5,MOM10,ATR5,ATR10,MOM5Indx,MOM10Indx,ATR5Indx,ATR10Indx)
dataset = na.omit(dataset)

## Understanding the dataset using descriptive statistics
print(head(dataset),5)
```

```
##    class forceindex     WillR5    WillR10    WillR15      RSI5    RSI10
## 17  DOWN  -35593378 0.16030534 0.13755453 0.12949635 100.00000 72.64431
## 18  DOWN -313465336 0.85714229 0.56986875 0.53648488  30.64797 46.20802
## 19    UP -339826872 0.30913166 0.21749274 0.21749274  68.97766 68.96093
## 20  DOWN  217845912 0.02854369 0.02072937 0.02072937  78.52883 76.98789
## 21    UP -103844085 0.25888751 0.19189157 0.19189157  66.49872 69.48670
## 22    UP   36388176 0.12488642 0.10148178 0.09256781  73.61045 72.21792
##       RSI15      ROC5      ROC10     MOM5     MOM10     ATR5    ATR10  MOM5Indx
## 17 66.50529 6.7081840  0.9731238 1.777501  0.272499 0.389999 0.389999  37.67004
## 18 48.59097 0.3863135 -0.1006865 0.105000 -0.027500 1.017500 1.017500   7.00000
## 19 65.13349 5.2578656  4.6180330 1.439998  1.272499 2.245001 2.245001 -29.95996
## 20 71.45192 5.7829181  8.2877920 1.625000  2.274999 0.970002 0.970002 -41.89990
## 21 65.59853 3.6997697  9.6798390 1.045000  2.585001 0.787500 0.787500 -56.83008
## 22 69.02365 4.8894748 11.9256538 1.382499  3.160000 0.772500 0.772500 -36.24011
##     MOM10Indx ATR5Indx ATR10Indx
## 17  12.280029 16.65015  16.65015
## 18   1.290039 37.18005  37.18005
## 19 -20.869995 41.00000  41.00000
## 20   9.979980 35.45996  35.45996
## 21   2.319946 29.93994  29.93994
## 22   1.429932 40.76001  40.76001
```

```
dim(dataset)
```

```
## [1] 1587   18
```

```
y = dataset$class
cbind(freq=table(y), percentage=prop.table(table(y))*100)
```

```
##      freq percentage
## DOWN  743    46.8179
## UP    844    53.1821
```

```
summary(dataset)
```

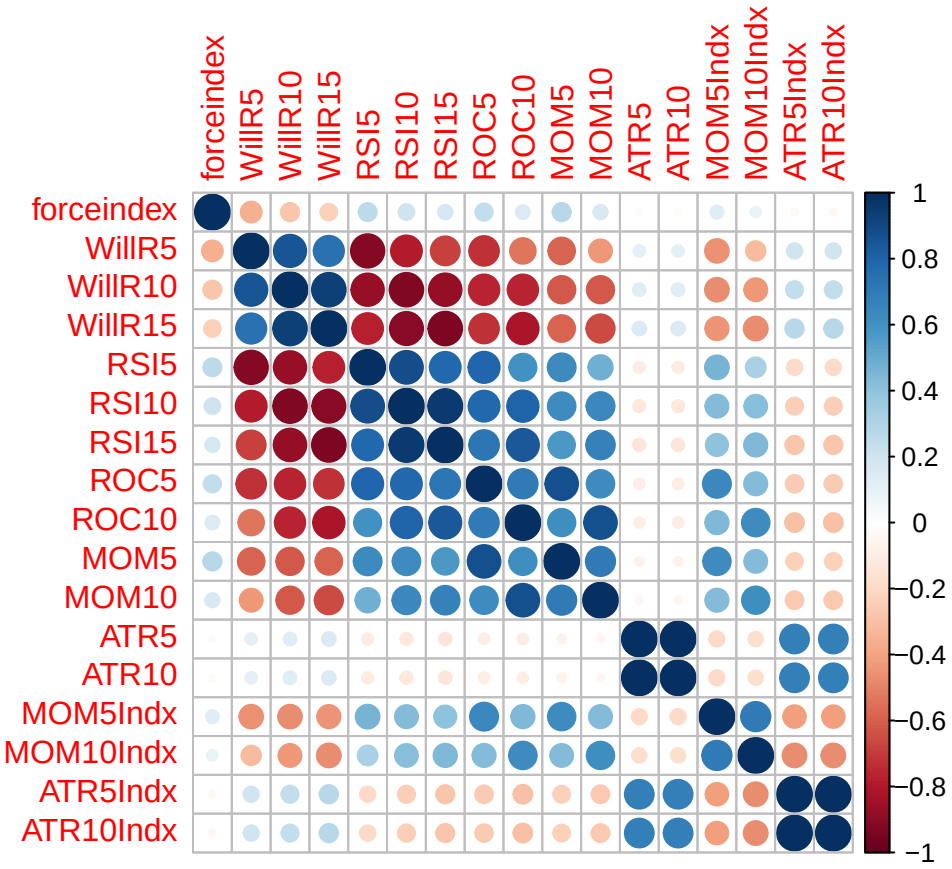
```
##     class             forceindex            WillR5           WillR10
##  Length:1587        Min.   :-1.802e+09   Min.   :0.0000   Min.   :0.0000
##  Class :character   1st Qu.:-3.329e+07   1st Qu.:0.1468   1st Qu.:0.1202
##  Mode  :character   Median : 3.807e+06   Median :0.3816   Median :0.3381
##                     Mean   :-1.106e+06   Mean   :0.4256   Mean   :0.4040
##                     3rd Qu.: 4.310e+07   3rd Qu.:0.6943   3rd Qu.:0.6784
```

```
##                     Max.   : 1.727e+09   Max.   :1.0000   Max.   :1.0000
##     WillR15            RSI5            RSI10            RSI15
##  Min.   :0.0000   Min.   :  0.00   Min.   : 0.4707   Min.   : 3.328
##  1st Qu.:0.1123   1st Qu.: 33.06   1st Qu.:40.3996   1st Qu.:42.209
##  Median :0.3121   Median : 58.81   Median :57.1429   Median :57.055
##  Mean   :0.3889   Mean   : 55.92   Mean   :56.2790   Mean   :56.341
##  3rd Qu.:0.6574   3rd Qu.: 80.67   3rd Qu.:73.9680   3rd Qu.:71.802
##  Max.   :1.0000   Max.   :100.00   Max.   :99.5530   Max.   :96.746
##       ROC5              ROC10              MOM5              MOM10
##  Min.   :-17.5307   Min.   :-20.686   Min.   :-17.9100   Min.   :-22.0500
##  1st Qu.: -1.5559   1st Qu.: -2.007   1st Qu.: -0.6225   1st Qu.: -0.7800
##  Median :  0.7051   Median :  1.491   Median :  0.2725   Median :  0.6050
##  Mean   :  0.5551   Mean   :  1.126   Mean   :  0.3108   Mean   :  0.6339
##  3rd Qu.:  2.7161   3rd Qu.:  4.438   3rd Qu.:  1.1975   3rd Qu.:  1.9125
##  Max.   : 18.4141   Max.   : 22.680   Max.   : 17.7125   Max.   : 21.0575
##       ATR5              ATR10            MOM5Indx          MOM10Indx
##  Min.   : 0.1450   Min.   : 0.1450   Min.   :-543.300   Min.   :-732.02
##  1st Qu.: 0.4600   1st Qu.: 0.4600   1st Qu.: -14.915   1st Qu.: -14.76
##  Median : 0.7675   Median : 0.7675   Median :  10.350   Median :  18.67
##  Mean   : 1.2895   Mean   : 1.2895   Mean   :   6.654   Mean   :  13.46
##  3rd Qu.: 1.5375   3rd Qu.: 1.5375   3rd Qu.:  36.065   3rd Qu.:  54.84
##  Max.   :12.8100   Max.   :12.8100   Max.   : 389.250   Max.   : 426.28
##     ATR5Indx        ATR10Indx
##  Min.   :  4.48   Min.   :  4.48
##  1st Qu.: 14.38   1st Qu.: 14.38
##  Median : 23.33   Median : 23.33
##  Mean   : 31.89   Mean   : 31.89
##  3rd Qu.: 38.22   3rd Qu.: 38.22
##  Max.   :330.08   Max.   :330.08
```

```
## Visualizing the dataset using a correlation matrix
correlations = cor(dataset[,c(2:18)])
print(head(correlations))
```

```
##            forceindex     WillR5    WillR10    WillR15       RSI5      RSI10
## forceindex  1.0000000 -0.3585937 -0.2748742 -0.2307385  0.2667329  0.2055023
## WillR5     -0.3585937  1.0000000  0.8552643  0.7490779 -0.9153482 -0.7833994
## WillR10    -0.2748742  0.8552643  1.0000000  0.9308607 -0.8640819 -0.9280065
## WillR15    -0.2307385  0.7490779  0.9308607  1.0000000 -0.7794132 -0.9022314
## RSI5        0.2667329 -0.9153482 -0.8640819 -0.7794132  1.0000000  0.8868554
## RSI10       0.2055023 -0.7833994 -0.9280065 -0.9022314  0.8868554  1.0000000
##                RSI15       ROC5      ROC10       MOM5      MOM10        ATR5
## forceindex  0.1713309  0.2464481  0.1585779  0.2739862  0.1669174 -0.02729178
## WillR5     -0.6859790 -0.7219836 -0.5387852 -0.5834027 -0.4319294  0.11179286
## WillR10    -0.8730977 -0.7644926 -0.7602627 -0.6161701 -0.6147023  0.13496171
## WillR15    -0.9302693 -0.7241512 -0.8122093 -0.5877809 -0.6584014  0.15416496
## RSI5        0.7831675  0.7976217  0.6045578  0.6372632  0.4866980 -0.10669103
## RSI10       0.9551683  0.7861393  0.8081152  0.6280628  0.6458834 -0.12195021
##                 ATR10   MOM5Indx   MOM10Indx    ATR5Indx   ATR10Indx
## forceindex -0.02729178  0.1397387  0.09353774 -0.03905905 -0.03905905
## WillR5      0.11179286 -0.4593531 -0.31006327  0.19455489  0.19455489
## WillR10     0.13496171 -0.4626066 -0.43273161  0.24057459  0.24057459
## WillR15     0.15416496 -0.4444042 -0.46900501  0.27813764  0.27813764
## RSI5       -0.10669103  0.4660960  0.32481653 -0.19810726 -0.19810726
## RSI10      -0.12195021  0.4349677  0.42812620 -0.24267894 -0.24267894
```

```
corrplot(correlations, method="circle")
```



Predisposizione del modello

```
## Selecting features using the random.forest.importance function from the FSelector package
set.seed(5)
weights = random.forest.importance(class~., dataset, importance.type = 1)
print(weights)
```

```
##             attr_importance
## forceindex       6.81475865
## WillR5           3.69021642
## WillR10          0.69289872
## WillR15         -1.62066144
## RSI5             4.12714497
## RSI10            1.82997997
## RSI15            0.03738077
## ROC5             2.48447192
## ROC10            1.93138401
## MOM5             2.96725327
## MOM10            2.57323498
## ATR5             1.73286061
## ATR10            1.06291985
## MOM5Indx        -1.71839859
## MOM10Indx       -3.84739395
```

```
## ATR5Indx       -0.84697956
## ATR10Indx       0.01175714
```

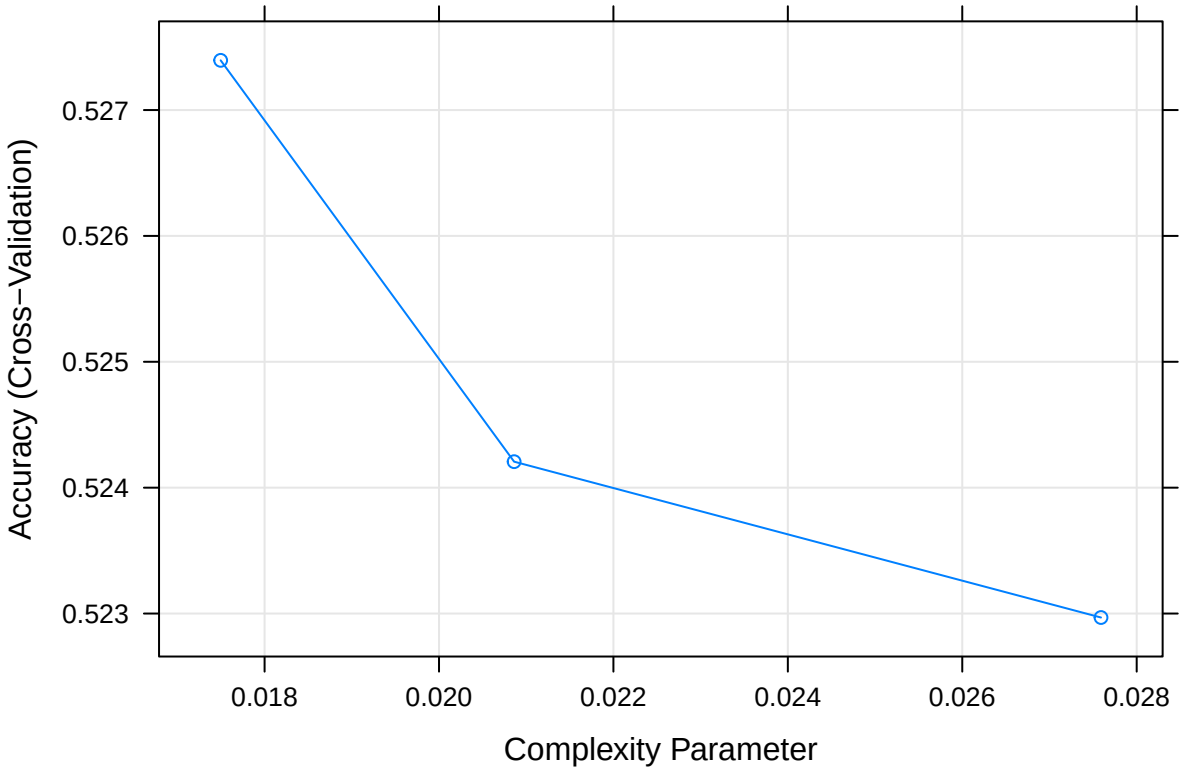
```
set.seed(5)
subset = cutoff.k(weights, 10)
print(subset)
```

```
##  [1] "forceindex" "RSI5"       "WillR5"     "MOM5"       "MOM10"
##  [6] "ROC5"       "ROC10"      "RSI10"      "ATR5"       "ATR10"
```

```
## Creating a dataframe using the selected features
dataset_rf = data.frame(class,forceindex,WillR5,WillR10,RSI5,RSI10,RSI15,ROC5,ROC10,MOM5,MOM10Indx)
dataset_rf = na.omit(dataset_rf)

# Resampling method used - 10-fold cross validation
# with "Accuracy" as the model evaluation metric.
trainControl = trainControl(method="cv", number=10)
metric = "Accuracy"
```

Fit dei modelli

```
## Trying four different Classification algorithms
# k-Nearest Neighbors (KNN)
set.seed(5)


fit.knn = train(class~., data=dataset_rf, method="knn",
                metric=metric, preProc=c("range"),trControl=trainControl)

# Classification and Regression Trees (CART)
set.seed(5)
fit.cart = train(class~., data=dataset_rf, method="rpart",
                 metric=metric,preProc=c("range"),trControl=trainControl)
plot(fit.cart)
```




```
# Naive Bayes (NB)
set.seed(5)
fit.nb = train(class~., data=dataset_rf, method="nb",
               metric=metric, preProc=c("range"),trControl=trainControl)
```

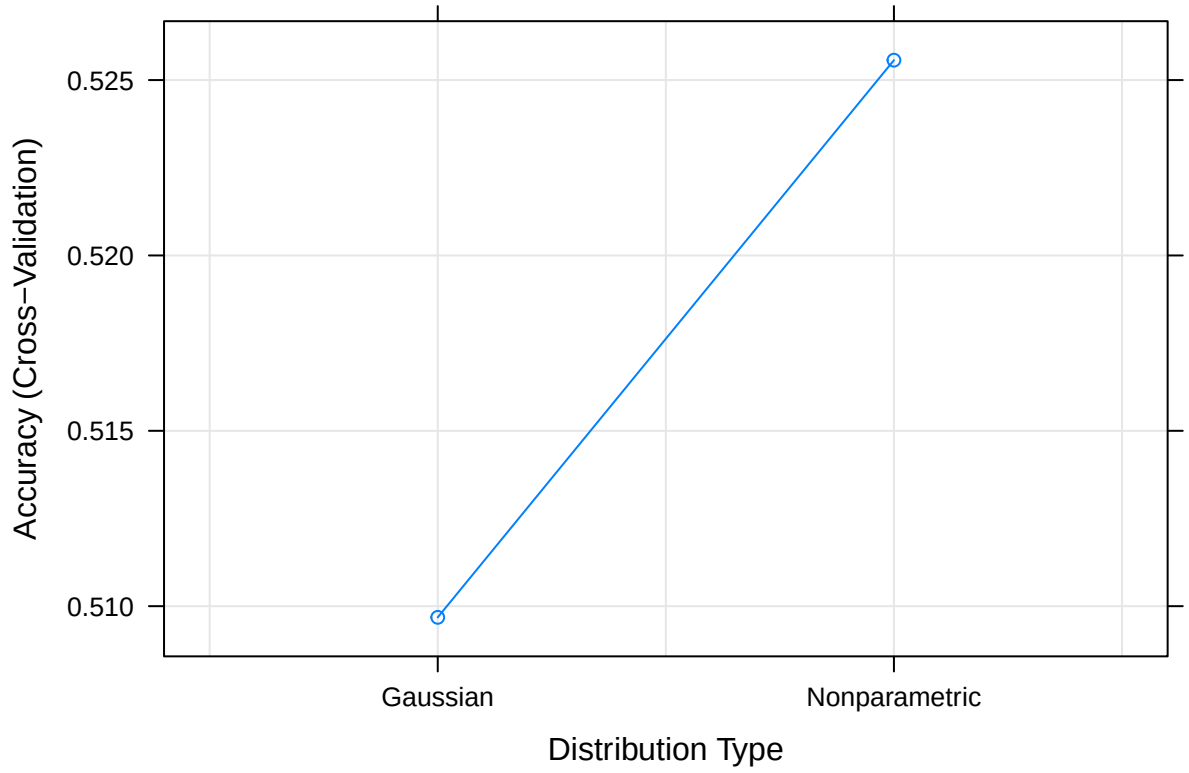
```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 135

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 139

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 141

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 15

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 15

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 130

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 131

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 145

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
```

```
## observation 124
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 136
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 148
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 136
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 138
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 139
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 148
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 124
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 125
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 139
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 123
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 131
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 123
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 130
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 131
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 144
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 132
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 144
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 134
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 147
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 123
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 123
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 136
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 137
```

```
plot(fit.nb)
```




```
# Support Vector Machine with Radial Basis Function (SVM)
set.seed(5)
fit.svm = train(class~., data=dataset_rf, method="svmRadial",
                metric=metric,preProc=c("range"),trControl=trainControl)
```

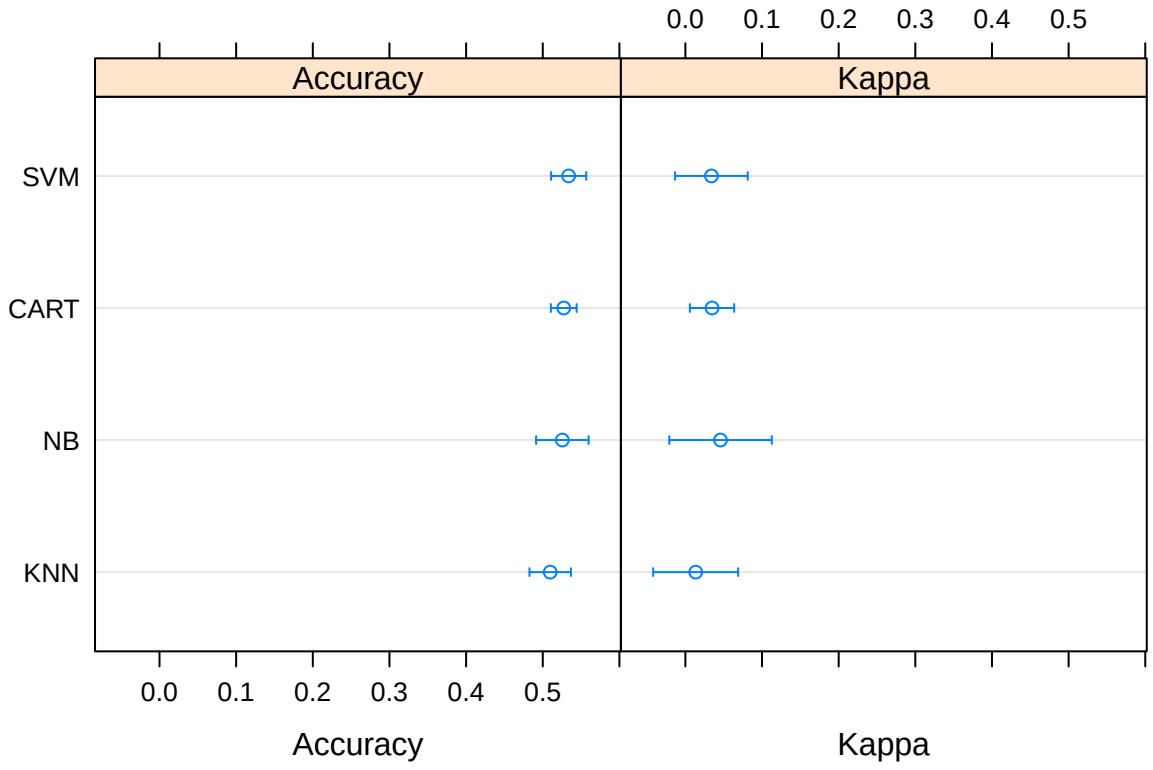
Valutazione e tuning del KNN model

```
## Evaluating the algorithms using the "Accuracy" metric
results = resamples(list(KNN=fit.knn,CART=fit.cart, NB=fit.nb, SVM=fit.svm))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: KNN, CART, NB, SVM
## Number of resamples: 10
##
```

```
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## KNN  0.4750000 0.4810127 0.4953029 0.5097369 0.5338249 0.5812500    0
## CART 0.4873418 0.5188093 0.5283019 0.5273947 0.5362935 0.5632911    0
## NB   0.4562500 0.4921384 0.5284810 0.5255672 0.5489122 0.6163522    0
## SVM  0.4750000 0.5197437 0.5284810 0.5337910 0.5513693 0.5822785    0
##
## Kappa
##             Min.       1st Qu.      Median       Mean    3rd Qu.       Max. NA's
## KNN  -0.05910165 -0.042990271 -0.01727863 0.01344474 0.06039262 0.16640747    0
## CART -0.03193033  0.005776115  0.02899271 0.03481175 0.06807296 0.09466866    0
## NB   -0.08580343 -0.028387035  0.05734996 0.04584485 0.08948689 0.22833957    0
## SVM  -0.07951807  0.005236906  0.02289245 0.03397246 0.06797023 0.13903567    0
```

```
dotplot(results)
```




```
## Tuning the shortlisted algorithm (KNN algorithm)
set.seed(5)
grid = expand.grid(.k=seq(1,10,by=1))
fit.knn = train(class~., data=dataset_rf, method="knn", metric=metric, tuneGrid=grid,
                preProc=c("range"), trControl=trainControl)
print(fit.knn)
```

```
## k-Nearest Neighbors
##
## 1587 samples
##   10 predictor
```

```
##    2 classes: 'DOWN', 'UP'
##
## Pre-processing: re-scaling to [0, 1] (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1429, 1429, 1429, 1428, 1427, 1429, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.4940090  -0.015477791
##    2  0.5034989   0.004979248
##    3  0.5040445   0.002826038
##    4  0.5091120   0.012449317
##    5  0.5059552   0.005017050
##    6  0.5235972   0.041587049
##    7  0.5097369   0.013444742
##    8  0.5154129   0.026839372
##    9  0.5084590   0.009327204
##   10  0.5096932   0.013776120
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 6.
```