ATTACCHI TERRORISTICI: MODELLI CLASSIFICATIVI

Bonavoglia Marco, Davoli Sofia, Dargenio Elisabetta

IL DATASET

Il dataset globalterrorism.csv è formato da 129356 osservazioni su 39 variabili. Ogni riga corrisponde ad un attacco terroristico, di cui sono fornite informazioni quali ad esempio la data, la modalità d'esecuzione e la zona geografica. La variabile scelta come target è "iskilled" è stata ricavata a partire dalla variabile "nkill", corrispondente al numero di vittime causate dall'attacco. Nel caso in cui ci sia stata almeno una vittima, la variabile "iskilled" assume valore "yes", altrimenti "no".

```
a$iskilled=ifelse(a$nkill>0,'yes','no')
prop.table(table(a$iskilled))
```

```
       no       yes
0.5232382 0.4767618
```

La variabile target ha una probabilità a priori dell'evento yes pari a 47.68%.

MISSING VALUES

Osservando il dataset si nota una grande quantità di dati mancanti, motivo per cui riteniamo necessario studiare i missing data di ogni variabile e valutare se togliere alcune di queste.

```
sapply(a, function(x)(sum(is.na(x))))
```

```
          iyear          imonth            iday         country      region_txt
              0               0               0               0               0
           city     specificity       doubtterr        multiple         success
            440               4           12116               0               0
        suicide  attacktype1_txt  targtype1_txt targsubtype1_txt           corp1
              0               0               0            7551           35486
        target1      natlty1_txt           gname          motive   guncertainyes
            537            1041               0           92181             270
     individual          nperps        nperpcap         claimed   weaptype1_txt
              0          110994           61536           58394               0
    weapsubtype1 weapsubtype1_txt      weapdetail           nkill          nwound
          13785           13785           39532               0            4357
       property   propextent_txt       ishostkid          ransom        dbsource
          14382           86355             234           67625            4457
        INT_LOG         INT_IDEO        INT_MISC         INT_ANY        iskilled
          75891           76011            4895           68868               0
```

Togliamo le variabili con una proporzione di missing data superiore al 20%, ovvero corp1, motive, nperps, nperpcap, claimed, weapdetail, propextent_txt, ransom, INT_LOG, INT_IDEO, INT_ANY. Inoltre, togliamo anche nkill, la variabile dalla quale è stata ricavata la risposta iskilled, e weapsubtype1, ridondante in quanto già presente weapsubtype1_txt.

Eliminate tutte queste variabili, il nostro dataset ha ora 129356 osservazioni su 27 variabili; vista l'alta numerosità delle unità possiamo eliminare le osservazioni non complete.

```
b=na.omit(a)
```

Il nuovo dataset ha 81359 osservazioni complete.

Vista la natura delle covariate – tutte categoriali eccetto una, nwound – non è necessario verificare se vi è collinearità tra variabili mediante la matrice di correlazione.

NZV

Passiamo a studiare la near zero variance sul dataset b.

```
nzv = nearZeroVar(b, saveMetrics = TRUE)
```

	freqRatio	percentUnique	zeroVar	nzv
iyear	1.091680	0.056539535	FALSE	FALSE
imonth	1.000680	0.015978564	FALSE	FALSE
iday	1.049209	0.039331850	FALSE	FALSE
country	1.364800	0.237220221	FALSE	FALSE
region_txt	1.005263	0.014749444	FALSE	FALSE
city	1.157789	23.364348136	FALSE	FALSE
specificity	11.074043	0.006145602	FALSE	FALSE
doubtterr	4.815511	0.002458241	FALSE	FALSE
multiple	7.921921	0.002458241	FALSE	FALSE
success	8.756446	0.002458241	FALSE	FALSE
suicide	32.689027	0.002458241	FALSE	TRUE
attacktype1_txt	2.216664	0.011062083	FALSE	FALSE
targtype1_txt	1.306107	0.025811527	FALSE	FALSE
targsubtype1_txt	1.073171	0.135203235	FALSE	FALSE
target1	1.484118	54.636856402	FALSE	FALSE
natlty1_txt	1.375448	0.243365823	FALSE	FALSE
gname	12.741133	2.724959746	FALSE	FALSE
guncertainyes	12.111845	0.002458241	FALSE	FALSE
individual	422.744792	0.002458241	FALSE	TRUE
weaptype1_txt	1.574332	0.006145602	FALSE	FALSE
weapsubtype1_txt	1.736886	0.035644489	FALSE	FALSE
nwound	6.220951	0.235991101	FALSE	FALSE
property	1.650044	0.002458241	FALSE	FALSE
ishostkid	26.283367	0.002458241	FALSE	TRUE
dbsource	1.380018	0.028269767	FALSE	FALSE
INT_MISC	8.408928	0.002458241	FALSE	FALSE
iskilled	1.125366	0.002458241	FALSE	FALSE

Vi sono tre variabili con nzv, ovvero suicide, individual e ishostkid, che vengono rimosse dal dataset. Tra le 24 variabili rimanenti ce ne sono 7 con un numero elevato (>40) di livelli, che decidiamo di togliere: iyear, country, city, targsubtype1_txt,natlty1_txt, target1, gname.

DESCRIZIONE VARIABILI DEL DATASET DEFINITIVO

Imonth: numero del mese (categoriale, 12 livelli)

iday: numero del giorno (categoriale, 31 livelli)

region_txt: macroregione (categoriale, 12 livelli)

specificity: specificità (categoriale, 5 livelli)

doubtterr: se vi sono dubbi sulla natura terroristica dell'attacco (dicotomica)

multiple: se è un attacco multiplo (dicotomica)

success: se l'attacco ha avuto successo (dicotomica)

attacktype1_txt: tipo di attacco (categoriale, 9 livelli)

targettype1: tipologia del target (categoriale, 22 livelli)

guncertainyes: certezza della presenza di una pistola (dicotomica)

weaptype1_txt: tipologia d'arma (categoriale, 12 livelli)

weapsubtype1_txt: sottotipologia di arma (categoriale, 29 livelli)

nwound: numero di feriti (numerica)

property: se ci sono stati danni a proprietà (dicotomica)

dbsource: fonte dei dati sull'attacco (categoriale 25 livelli)

INT_MISC: se il gruppo ha attaccato un target di una nazionalità differente dal luogo dell'attacco (dicotomica)

Iskilled: se vi sono state vittime (dicotomica, variabile target)

Vista l'alto numero di osservazioni, che potrebbe rallentare e complicare gli algoritmi in fase di modellizzazione, decidiamo di estrarre un campione casuale di 8000 unità dall'attuale dataset.

```
c=b[sample(nrow(b),8000),]
```

Vediamo come è fatto il nostro dataset c.

```
head(c)
```

```
       imonth iday               region_txt specificity doubtterr multiple success
48367      11   20             South America           1        no      yes     yes
74075       9    9 Middle East & North Africa           1        no       no     yes
117123     12   29                 South Asia           1        no       no     yes
24424      12    8 Middle East & North Africa           1        no       no     yes
115808     11   19                 South Asia           1        no       no     yes
122077      7   28 Middle East & North Africa           1        no       no     yes
         attacktype1_txt              targtype1_txt guncertainyes
48367  Bombing/Explosion             Transportation           yes
74075      Assassination        Government (General)            no
117123     Assassination        Government (General)            no
24424  Bombing/Explosion Private Citizens & Property           yes
115808     Armed Assault                     Police            no
122077 Bombing/Explosion Private Citizens & Property            no
                  weaptype1_txt                            weapsubtype1_txt nwound
48367  Explosives/Bombs/Dynamite                                    Vehicle      0
74075                  Firearms                                    Handgun      0
117123                 Firearms                            Unknown Gun Type      1
24424  Explosives/Bombs/Dynamite Projectile (rockets, mortars, RPGs, etc.)      0
115808                 Firearms                            Automatic Weapon      1
122077 Explosives/Bombs/Dynamite                                    Vehicle     17
       property                dbsource INT_MISC iskilled
48367       yes                    ISVG       no       no
74075        no                    PGIS       no      yes
117123       no                   CETIS       no      yes
24424       yes                    PGIS       no       no
115808       no START Primary Collection       no      yes
122077      yes                    ISVG       no      yes
```

SCALE E TRASFORMAZIONI

Su questo campione andiamo ad applicare all'unica covariata numerica Boxcox.

```
scaled_bc <- preProcess(c, method = c("scale", "BoxCox"))
```

```
Pre-processing:
  - ignored (16)
  - scaled (1)
```

La variabile nwound è stata dunque scalata, andiamo ad aggiungerla al nuovo dataset all_bc.

```
all_bc=predict(scaled_bc, newdata = c)
```

MODEL SELECTION

Per scegliere quali variabili tenere nel dataset finale, sul quale verranno costruiti i modelli, facciamo una model selection usando la funzione boruta.

```
boruta.train <- Boruta(iskilled~., data = all_bc, doTrace = 1)
final.boruta <- TentativeRoughFix(boruta.train)
```

```
 14 attributes confirmed important: attacktype1_txt, dbsource, doubtterr,
imonth, INT_MISC and 9 more;
 2 attributes confirmed unimportant: guncertainyes, iday;
```

Boruta suggerisce di togliere guncertainyes e iday. Il dataset rimane dunque con 15 variabili.

Prima di cominciare lo step 1 dividiamo il dataset in train set e test set.

```
set.seed(1234)
split <- createDataPartition(y=c$iskilled, p = 0.66, list = FALSE)
train <- all_bc[split,]
test <- all_bc[-split,]
```

STEP1

CLASSIFICATION TREE

Il primo modello che abbiamo costruito è il classification tree, sia con il pacchetto rpart, sia con il pacchetto caret – funzione Train -.

```
cv.ct <- rpart(iskilled ~ ., data = train, method = "class",
               cp = 0.00001, minsplit = 5, xval = 5)
pruned1 <- prune(cv.ct,
                 cp = cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
best_tree.pred <- predict(pruned1, test,type = "class")
confusionMatrix(best_tree.pred, test$iskilled, positive="yes")
```

```
          Reference
Prediction   no  yes
       no  1122  227
       yes  297 1073

Accuracy : 0.8073
Kappa : 0.6147

Sensitivity : 0.8254
Specificity : 0.7907
```

Con rpart abbiamo costruito l'albero e poi lo abbiamo potato minimizzando l'xerror utilizzando il cp. L'albero potato ha una accuracy dell'80.73% e un Kappa di 61.47%. Costruiamo ora l'albero con la funzione Train.

```
ctrl_tree <- trainControl(method = "cv", number=10 , savePredictions=T,search="
grid", summaryFunction = twoClassSummary , classProbs = TRUE)
tree <- train(iskilled~., data=train, method="rpart",
              trControl=ctrl_tree, tuneLength=10)
pred_tree=predict(tree,newdata=test)
confusionMatrix(pred_tree, test$iskilled)
```

```
   TrainROC TrainSens TrainSpec method
1 0.8377063 0.7816522 0.7967501  rpart
```

```
          Reference
Prediction   no  yes
       no  1084  203
       yes  335 1097

Accuracy : 0.8021
Kappa : 0.6052

Sensitivity : 0.7639
Specificity : 0.8438
```

L'albero generato con la funzione Train è già automaticamente potato. L'accuracy scende a 80.21% e il kappa a 60.52%, ma utilizzando la funzione Train sarà poi possibile un confronto diretto con gli altri modelli durante la fase di assessment, dunque optiamo per tenere questo secondo albero.

NAIVE BAYES

Il secondo modello creato è il Naive Bayes. Anche in questo caso si possono utilizzare due diversi pacchetti, klaR con la funzione NaiveBayes e caret con la funzione Train.

Creiamo il primo modello con naive_all1 e controlliamo se si presenta lo zero problem.

```
naive_all1 <- NaiveBayes(iskilled ~ ., data = train, usekernel = FALSE)
naive_all1$tables$guncertainyes
```

```
$success
        var
grouping         no        yes
     no  0.17272941 0.82727059
     yes 0.02567531 0.97432469
```

Abbiamo controllato tutte le table delle variabili e abbiamo osservato il presentarsi dello zero problem in alcune occasioni, riportiamo il caso di success. Per questo motivo creiamo un nuovo modello applicando la correzione di Laplace e ne valutiamo la bontà classificativa sul test.

```
naive_all4 <- NaiveBayes(iskilled ~ ., data = train, laplace = 100)
pred_test <- predict(naive_all4, test, type="class")
confusionMatrix(pred_test$class, test$iskilled)
```

```
          Reference
Prediction   no  yes
       no  1135  332
       yes  284  968

Accuracy : 0.7734
Kappa : 0.5453

Sensitivity : 0.7999
Specificity : 0.7446
```

Il Naive Bayes ha una accuracy di 77.34% e un kappa di 54.53%, un valore molto basso. Riproviamo ora con la funzione Train.

```
ctrl_nb <- trainControl(method = "cv", number=10 , savePredictions=T,search="gr
           id", summaryFunction = twoClassSummary , classProbs = TRUE)
nb <- train(iskilled~., data=train, method="nb",
      trControl=ctrl_nb, tuneLength=10)
getTrainPerf(nb)
pred_nb=predict(nb,newdata=test)
confusionMatrix(pred_nb, test$iskilled)
```

```
   TrainROC TrainSens TrainSpec method
1 0.8543635 0.1081568 0.9932806     nb
```

```
          Reference
Prediction   no  yes
       no  1120  298
       yes  328  973

Accuracy : 0.7698
Kappa : 0.5382

Sensitivity : 0.7735
Specificity : 0.7655
```

Il modello ha un'accuracy del 76.98% e un kappa del 53.82%, valori leggermente più bassi del modello ottenuto con NaiveBayes. Analogamente al classification tree, anche in questo caso preferiamo il modello creato con 'caret' poiché potrà essere più facilmente confrontato in fase di assessment e non comporta un'eccessiva perdita nelle metriche.

NEURAL NETWORK

Per poter costruire la rete neurale utilizzando il pacchetto 'nnet' è necessario che gli input siano tutti in formato numerico, motivo per cui ricodifichiamo le dicotomiche come "1; 0" e dummizziamo tutte le variabili categoriali.

```
y=ifelse(train$iskilled=="yes",1,0)
dummies <- dummyVars(imonth ~ ., data = train , fullRank = T,
           na.action = na.pass)
dummized = data.frame(predict(dummies, newdata = train))
all_train=cbind(dummized, y)
all_train$iskilled.yes=NULL
mynet <- nnet(all_train[,-114], y ,  entropy=T, size=3, decay=0.1,
         maxit=20, trace=T)
mynet.pred <- as.numeric(predict(mynet, all_train[,-114], type='class'))
confusionMatrix(mynet.pred,y)
```

```
          Reference
Prediction    0    1
         0 2360  347
         1  453 2121

Accuracy : 0.8485
Kappa : 0.6965

Sensitivity : 0.8390
Specificity : 0.8594
```

La neural network ha un'accuracy dell'84.85% e un kappa del 69.65%.

Costruiamo la rete neurale anche con il pacchetto 'caret', che non necessità della dummizzazione.

```
ctrl_nn <- trainControl(method = "cv", number=10 , savePredictions=T,search="gr
           id", summaryFunction = twoClassSummary , classProbs = TRUE)
nn <- train(iskilled~., data=train, method="nnet",
              trControl=ctrl_nn, tuneLength=10)
getTrainPerf(nn)
pred_nn=predict(nn,newdata=test)
confusionMatrix(pred_nn, test$iskilled)
```

```
          Reference
Prediction   no  yes
       no  1205  234
       yes  243 1037

Accuracy : 0.8246
Kappa : 0.6478

Sensitivity : 0.8322
Specificity : 0.8159
```

Con il pacchetto 'caret' l'accuracy è dell'82.46% e il kappa 64.78%. La funzione nnet fornisce risultati lievemente migliori ma con la funzione Train il confronto in fase di assessment sarà più diretto, dunque scegliamo quest'ultimo modello.

K NEAREST NEIGHBORS

```
ctrl_knn <- trainControl(method = "cv", number=10 , savePredictions=T,search="g
rid", summaryFunction = twoClassSummary , classProbs = TRUE)
tunegrid_knn <- expand.grid(k=c(15, 30, 45))
knn <- train(iskilled~., data=train, method="knn",
             tuneGrid=tunegrid_knn, trControl=ctrl_knn, tuneLength=10)
getTrainPerf(knn)
pred=predict(knn,newdata=test)
confusionMatrix(pred, test$iskilled)
```

```
   TrainROC TrainSens TrainSpec method
1 0.8734038 0.7465536 0.8330684    knn
```

```
          Reference
Prediction   no  yes
       no  1105  227
       yes  343 1044
```

```
Accuracy : 0.7904
Kappa : 0.5813
Sensitivity : 0.7631
Specificity : 0.8214
```

Il kNN ha un'accuracy del 79.04% e un kappa del 58.13%.

GLM

```
ctrl_glm <- trainControl(method = "cv" , number=10,
            summaryFunction = twoClassSummary , classProbs = TRUE)
glm <- train(iskilled ~ ., data = train, method = "glm",
             trControl=ctrl_glm)
getTrainPerf(glm)
predglm=predict(glm,newdata=test)
confusionMatrix(predglm, test$iskilled)
```

```
   TrainROC TrainSens TrainSpec method
1 0.7069916 0.6404457  0.713571    glm
```

```
          Reference
Prediction   no  yes
       no  1203  237
       yes  245 1034
```

```
Accuracy : 0.8227
Kappa : 0.6441
Sensitivity : 0.8308
Specificity : 0.8135
```

Glm ha un'accuracy di 82.27% e un kappa di 64.41%.

RANDOM FOREST

```
cvCtrl_rf <- trainControl(method = "cv", number=10, search="grid", classProbs =
 TRUE,
                          summaryFunction = twoClassSummary)
rfTune_rf <- train(iskilled ~ ., data = train, method = "rf",
                   tuneLength = 10,
                   trControl = cvCtrl_rf)
getTrainPerf(rfTune_rf)
pred_rf=predict(rfTune_rf,newdata=test)
confusionMatrix(pred_rf, test$iskilled)
```

```
   TrainROC TrainSens TrainSpec method
1 0.9115215 0.8350496 0.8346829     rf
```

```
          Reference
Prediction   no  yes
       no  1219  220
       yes  229 1051
```

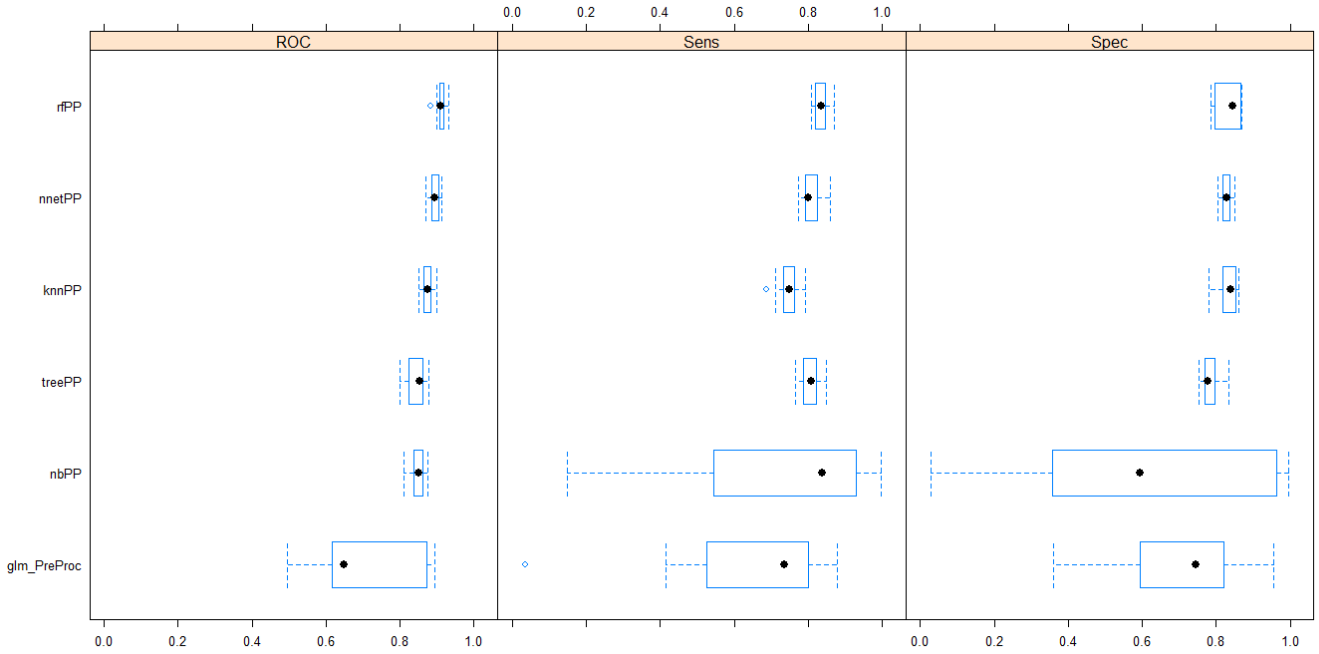
```
Accuracy : 0.8349
Kappa : 0.6685
Sensitivity : 0.8419
Specificity : 0.8269
```

La random forest dà un'accuracy dell'83.49% e un kappa del 66.85%.

STEP 2

Una volta costruiti i nostri sei modelli – glm, k Nearest Neighbors, Neural Network, Naive Bayes, Random Forest e Classification Tree – passiamo al confronto tra questi, per poter scegliere il modello più performante sul validation set e usarlo poi per la previsione di nuovi casi. L'assessment tra modelli viene fatto confrontando la ROC curve. Vediamo graficamente le differenze in quanto a ROC, Sensitivity e Specificity.

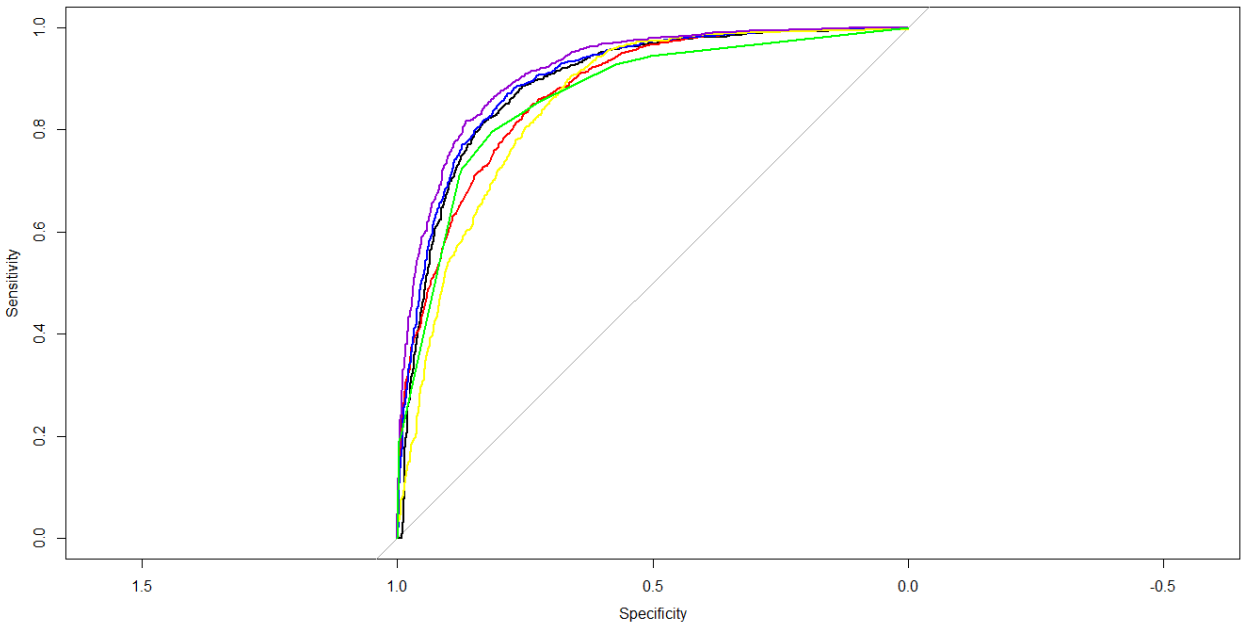
```
results <- resamples(list(glm_PreProc=glm, knnPP=knn,
                          nnetPP=nn ,nbPP=nb, rfPP=rfTune_rf, treePP=tree))
bwplot(results)
```



Osserviamo che la Random Forest ha il valore più elevato in tutte e tre le metriche. Vediamo meglio a livello numerico. Stimiamo ora le probabilità dell'evento "yes" sul test set, calcoliamo le ROC e rappresentiamole graficamente in un plot.

```
test$p1 = predict(glm       , test, "prob")[,1]
test$p2 = predict(knn         , test, "prob")[,1]
test$p3 = predict(nn    , test, "prob")[,1]
test$p4 = predict(nb     , test, "prob")[,1]
test$p5 = predict(rfTune_rf, test, "prob")[,1]
test$p6=  predict(tree, test, "prob")[,1]
r1=roc(iskilled ~ p1, data = test)
r2=roc(iskilled ~ p2, data = test)
r3=roc(iskilled ~ p3, data = test)
r4=roc(iskilled ~ p4, data = test)
r5=roc(iskilled ~ p5, data = test)
r6=roc(iskilled ~ p6, data = test)
plot(r1)
plot(r2,add=T,col="red")
plot(r3,add=T,col="blue")
plot(r4,add=T,col="yellow")
```

```
plot(r5,add=T,col="dark violet")
plot(r6,add=T,col="green")
```




```
metodo AUC
"glm"  "0.890402562909963"
"kNN"  "0.875133937692077"
"NN"   "0.899189201524879"
"NB"   "0.853416742374517"
"RF"   "0.915323939039604"
"Tree" "0.865658864773463"
```

Anche il grafico della curva ROC mostra che la Random Forest è il modello migliore, con un'area sotto la curva pari a 0.915. Decidiamo dunque di passare allo step 3 con questo modello.

STEP 3

In questo step ci occupiamo di valutare le performance classificative del modello scelto sul dataset di validation. Riprendiamo dunque le probabilità previste sul validation set e creiamo un dataset df contenente il target, la probabilità dell'evento "yes" e quella dell'evento "no".

```
predP <- predict(rfTune_rf, test ,type = "prob")
df=data.frame(cbind(test$iskilled , predP))
colnames(df)=c("iskilled","ProbNo","ProbYes")
```

Creiamo ora un ciclo che trova la matrice di confusione per ogni valore possibile della soglia, ricavandone ad ogni giro le metriche, quali Sensitivity, Specificity e Precision.

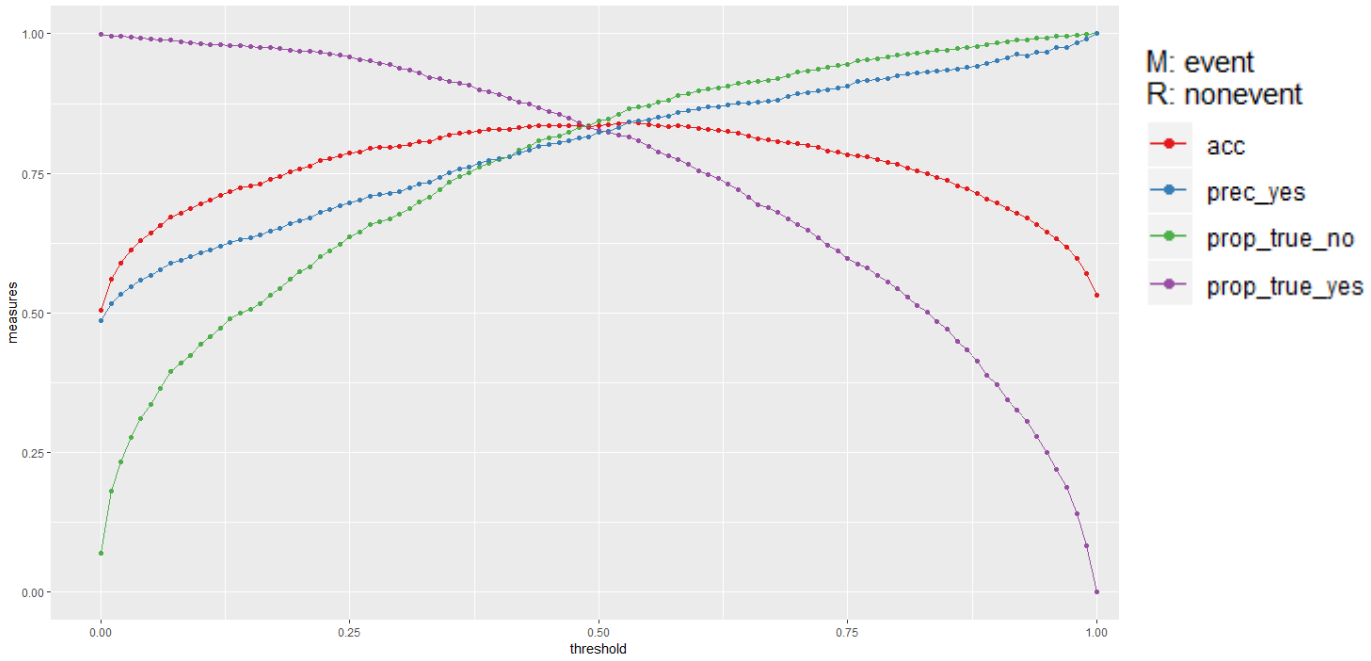
```
thresholds <- seq(from = 0, to = 1, by = 0.01)
prop_table <- data.frame(threshold = thresholds, prop_true_yes = NA,  prop_true
_no = NA, true_yes = NA,  true_no = NA ,fn_yes=NA)

for (threshold in thresholds) {
  pred <- ifelse(df$ProbYes > threshold, "yes", "no")
  pred_t <- ifelse(pred == df$iskilled, TRUE, FALSE)

  group <- data.frame(df, "pred" = pred_t) %>%
    group_by(iskilled, pred) %>%
    dplyr::summarise(n = n())

  group_yes <- filter(group, iskilled == "yes")

  true_yes=sum(filter(group_yes, pred == TRUE)$n)
  prop_yes <- sum(filter(group_yes, pred == TRUE)$n) / sum(group_yes$n)

  prop_table[prop_table$threshold == threshold, "prop_true_yes"] <- prop_yes
  prop_table[prop_table$threshold == threshold, "true_yes"] <- true_yes

  fn_yes=sum(filter(group_yes, pred == FALSE)$n)
  # true Yes predicted as No
  prop_table[prop_table$threshold == threshold, "fn_yes"] <- fn_yes


  group_no <- filter(group, iskilled == "no")

  true_no=sum(filter(group_no, pred == TRUE)$n)
  prop_no <- sum(filter(group_no, pred == TRUE)$n) / sum(group_no$n)

  prop_table[prop_table$threshold == threshold, "prop_true_no"] <- prop_no
  prop_table[prop_table$threshold == threshold, "true_no"] <- true_no

}
```

Aggiungiamo a prop_table altre misure mancanti.

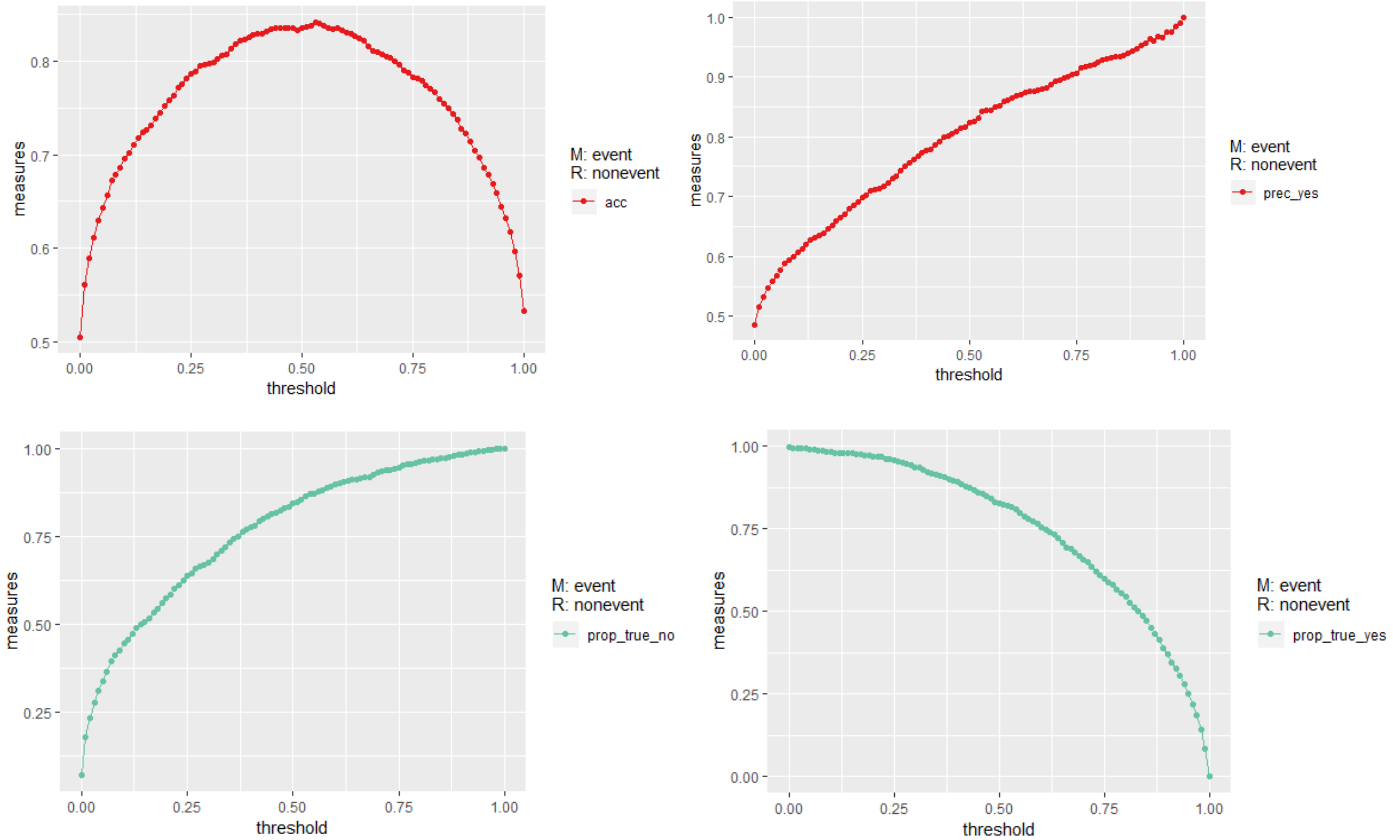
```
prop_table$n=nrow(test)
prop_table$fp_yes=nrow(test)-prop_table$true_no-prop_table$true_yes-prop_table$
                  fn_yes
prop_table$acc=(prop_table$true_no+prop_table$true_yes)/nrow(test)
prop_table$prec_yes=prop_table$true_yes/(prop_table$true_yes+prop_table$fp_yes)
```

Rappresentiamo graficamente le metriche.

```
gathered=prop_table2 %>%
  gather(x, y, prop_true_yes:prec_yes)
gathered %>%
  ggplot(aes(x = threshold, y = y, color = x)) +
  geom_point() +
  geom_line() +
  scale_color_brewer(palette = "Set1") +
  labs(y = "measures",
       color = "M: event\nR: nonevent")
```



Il grafico mostra come variano accuracy, precision, specificity e sensitivity, con le prime tre che crescono all'aumentare del valore soglia e la quarta che diminuisce. Nella scelta del valore di threshold abbiamo dato importanza ad accuracy e soprattutto precision, in quanto lo scopo del modello è principalmente quello di individuare gli attacchi terroristici che abbiano avuto vittime effettive. Il valore da noi scelto è di 0.51. Verifichiamo anche gli andamenti di accuracy e precision.



STEP 4

Studiate le performance classificative del modello con la Random Forest sul dataset di validation, creiamo un nuovo dataset di score e applichiamo il modello su questo per la previsione di nuovi casi.

```
score=test[c(1:800),-c(15:21)]
score$prob = predict(rfTune_rf, score, "prob")
probyes=score$prob[,2]
score$pred_y=ifelse(probyes>0.51, "yes","no")
head(score)
```

```
       imonth                region_txt specificity doubtterr multiple success
48367      11              South America           1        no      yes     yes
74075       9 Middle East & North Africa           1        no       no     yes
117123     12                South Asia           1        no       no     yes
122077      7 Middle East & North Africa           1        no       no     yes
49668       5             Eastern Europe           1        no       no      no
98527       9                South Asia           1        no       no     yes
         attacktype1_txt              targtype1_txt             weaptype1_txt
48367  Bombing/Explosion             Transportation Explosives/Bombs/Dynamite
74075      Assassination        Government (General)                 Firearms
117123     Assassination        Government (General)                 Firearms
122077 Bombing/Explosion Private Citizens & Property Explosives/Bombs/Dynamite
49668  Bombing/Explosion                     Police Explosives/Bombs/Dynamite
98527  Bombing/Explosion                   Business Explosives/Bombs/Dynamite
             weapsubtype1_txt     nwound property dbsource INT_MISC prob.no prob.yes
48367                 Vehicle 0.00000000      yes     ISVG       no   0.808    0.192
74075                 Handgun 0.00000000       no     PGIS       no   0.004    0.996
117123       Unknown Gun Type 0.06858171       no    CETIS       no   0.112    0.888
122077                Vehicle 1.16588909      yes     ISVG       no   0.076    0.924
49668  Unknown Explosive Type 0.00000000       no     ISVG       no   0.960    0.040
98527  Unknown Explosive Type 1.02872567      yes     PGIS       no   0.326    0.674
       pred_y
48367      no
74075     yes
117123    yes
122077    yes
49668      no
98527     yes
```