Progetto Data Mining

Kevin Capano 844018, *Sara Licaj* 846892, *Susanna Maugeri* 839365

Appello del 23 febbraio 2021

Il dataset considerato, reperito su Kaggle, contiene delle informazioni su diversi asteroidi vicini alla Terra. Si compone di 40 variabili e 4687 osservazioni.

Alcune delle variabili di interesse sono:

- Hazardous: variabile target, indica se l'asteroide è pericoloso o meno.

- Neo.Reference.ID: codice identificativo dell'asteroide.

- Name: nome dell'asteroide.

- Absolute.Magnitude: magnitudine assoluta, unità di misura della luminosità intrinseca di un oggetto celeste.

- Est.Dia.in.KM.min e Est.Dia.in.KM.max: stima minima e massima del diametro dell'asteroide in KM. Le stesse informazioni sono riportate anche in metri, miglia e piedi.

- Relative.Velocity.km.per.sec: velocità relativa dell'asteroide. La stessa informazione è riportata anche in chilometri orari e miglia orarie.

- Miss.Dist..kilometers: la stessa informazione è riportata anche in distanza lunare, unità astronimiche e miglia.

- Orbiting.Body: pianeta nella cui orbita si trova ciascun asteroide.

- Jupiter.Tisserand.Invariant: parametro di Tisserand per l'asteroide.

- Eccentricity: valore di eccentricità dell'orbita dell'asteroide.

- Semi Major Axis: valore del semiasse maggiore dell'orbita dell'asteroide.

- Orbital Period: tempo impiegato per una rivoluzione completa attorno al pianeta.

- Perihelion Distance: distanza dell'asteroide dal perielio.

- Aphelion Dist: distanza dell'asteroide dall'afelio.

Lo scopo della classificazione è quello di identificare gli asteroidi potenzialmente pericolosi, per esempio quelli che potrebbero entrare in collisione con la Terra.

Sommario

Importazione del dataset e controlli preliminari

```
library(readxl)

dataset <- read.csv("nasa.csv", sep=',')
```

Dataset è il dataset originale.

Ricodificazione del target

Hazaordous è la variabile target originale, ha come livelli True e False ed è di tipo char, tuttavia Caret richiede che il target sia di tipo factor e considera come event la label più "bassa": per questo si è deciso di etichettare l'event "asteroide pericoloso" con c0 e il non-event "asteroide non pericoloso" con c1.

```
dataset$Hazardous=ifelse(dataset$Hazardous=="False","c1","c0")
dataset$Hazardous=as.factor(dataset$Hazardous)
```

Bilanciamento del dataset

```
table(dataset$Hazardous)

##
##   c0   c1
##  755 3932

prop.table(table(dataset$Hazardous))

##
##        c0        c1
## 0.1610838 0.8389162
```

Il dataset sembra essere stato bilanciato, in quanto la percentuale di asteroidi pericolosi non è realistica perché troppo elevata. Tuttavia, nei metadati del dataset non è stato dichiarato un bilanciamento, si assume che delle priors verosimili siano 1% per gli asteroidi pericolosi e 99% per quelli non pericolosi. Si tratta di un dataset altamente sbilanciato che è stato manipolato. Teniamo conto delle true priors effettuando la correzione allo step 2, dopo aver plottato le curve ROC.

Ricerca delle magagne

```
stato=df_status(dataset, print_results=F)
head(stato%>% arrange(type))
```

```
##                    variable      type
## 1      Close.Approach.Date      character
## 2             Orbiting.Body      character
## 3 Orbit.Determination.Date      character
## 4                   Equinox      character
## 5                 Hazardous      factor
## 6      ï..Neo.Reference.ID      integer
```

L'unica variabile factor presente è il target, mentre 4 variabili sono di tipo character e riguardano le date e il nome del corpo attorno a cui orbita ciascun asteroide. Tutte le restanti variabili sono di tipo numerico.

```
head(stato%>% arrange(-p_zeros))
```

```
##               variable q_zeros p_zeros
## 1  Orbit.Uncertainity    1353   28.87
## 2 ï..Neo.Reference.ID       0    0.00
## 3                 Name       0    0.00
## 4  Absolute.Magnitude       0    0.00
## 5  Est.Dia.in.KM.min.       0    0.00
## 6  Est.Dia.in.KM.max.       0    0.00
```

Osserviamo che nel dataset solamente Orbit.Uncertainity ha degli zeri, più precisamente il 28.87% di tutti i suoi valori.

```
head(stato%>% arrange(unique))
```

```
##                        variable unique
## 1                Orbiting.Body    1
## 2                      Equinox    1
## 3                    Hazardous    2
## 4            Orbit.Uncertainity   10
## 5                     Orbit.ID  188
## 6            Absolute.Magnitude  269
```

Si osserva che Orbiting.Body ed Equinox sono variabili degeneri, quindi creano problemi di Zero Variance.

```
head(stato%>% arrange(-unique))
```

```
##                       variable unique
## 1 Relative.Velocity.km.per.sec   4687
## 2  Relative.Velocity.km.per.hr   4687
## 3               Miles.per.hour   4687
## 4     Miss.Dist..Astronomical.   4673
## 5       Miss.Dist..kilometers.   4661
## 6            Miss.Dist..lunar.   4660
```

Le uniche variabili che hanno un valore diverso per ciascuna osservazione sono quelle che riguardano la velocità. Si ipotizza che la variabile ID non rientri tra queste perché alcuni asteroidi vengono osservati più volte.

```
head(stato%>% arrange(-p_na))

##              variable      q_na    p_na
## 1    Neo.Reference.ID       0       0
## 2                Name       0       0
## 3  Absolute.Magnitude       0       0
## 4  Est.Dia.in.KM.min.       0       0
## 5  Est.Dia.in.KM.max.       0       0
## 6   Est.Dia.in.M.min.       0       0
```

Infine, non si osservano missing values da imputare.

Pulizia del dataset

Si decide di non considerare le variabili "Neo.Reference.ID", "Name" e "Orbit.ID", relative all'identificazione degli asteroidi, in quanto non utili all'analisi. Si rimuovono anche le due variabili degeneri, "Orbiting.Body" ed "Equinox", e le due variabili che contengono date, "Close.Approach.Date" e "Orbit.Determination.Date".

In questo modo le variabili del dataset sono 33 e si è risolto il problema Zero Variance, di cui soffrono quasi tutti i modelli.

STEP 1 - TUNING DEI MODELLI

Si dividono le covariate qualitative da quelle quantitative:

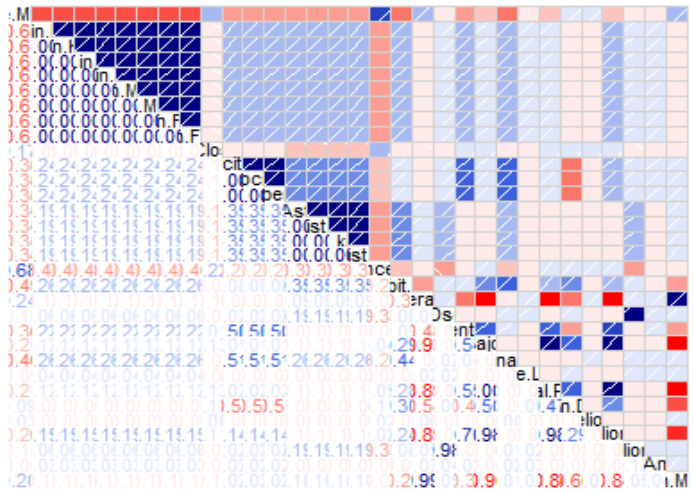
```
target <- dataset[, "Hazardous"]
str(target)

##  Factor w/ 2 levels "c0","c1": 1 2 1 2 1 2 2 2 2 1 ...

numeric <- sapply(dataset, function(x) is.numeric(x))
numericdata <-dataset[, numeric]
```

Numeric è la lista delle variabili numeriche, numericdata è un dataset che contiene solo le variabili numeriche.

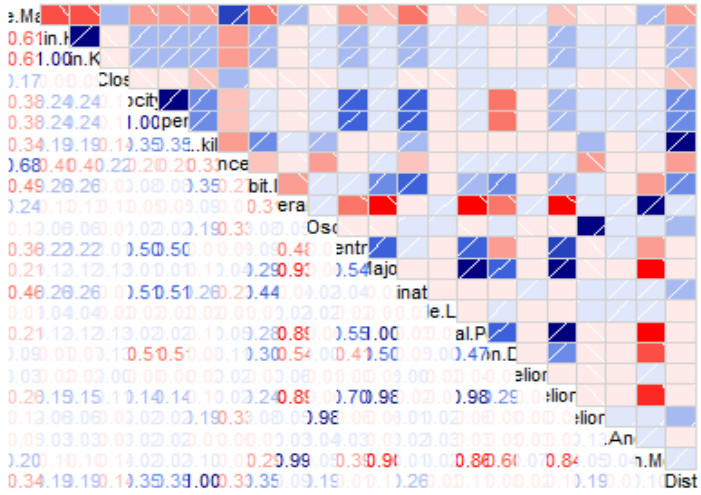
Pre-Processing

Collinearità

Si procede eliminando manualmente tutte le covariate che creano problemi di collinearità.

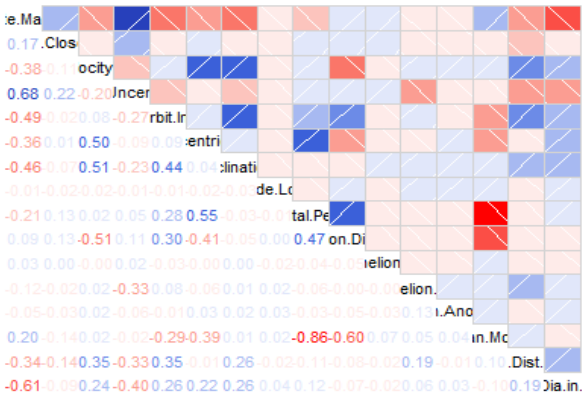


Tra alcune variabili vi è perfetta correlazione. Si tratta di variabili che esprimono le stesse quantità con unità di misura diverse, si decide di tenere solamente quelle in km. Dopo aver eliminato le variabili ridondanti, si osserva quali siano quelle ancora perfettamente correlate:



- Est.Dia.in.KM.min. e Est.Dia.in.KM.max. sono perfettamente correlate: ne teniamo una sola delle due e la chiamiamo semplicemente Est.Dia.in.KM..
- Relative.Velocity.km.per.sec e Miles.per.hour sono perfettamente correlate: teniamo solo Relative.Velocity.km.per.sec.
- Semi.Major.Axis e Orbital.Period sono perfettamente correlate: teniamo Orbital.Period.
- Semi.Major.Axis è perfettamente correlata anche con Aphelion.Dist: avendola eliminata si dovrebbe risolvere.
- Anche Orbital.Period e Aphelion.Dist sono perfettamente correlate: teniamo Orbital.Period.
- Perihelion.Time è perfettamente correlata con Epoch.Osculation: teniamo Perihelion.Time.
- Jupiter.Tisserand.Invariant è perfettamente correlata con Mean.Motion: teniamo Mean.Motion.
- Miss.Dist..kilometers. è perfettamente correlata con quella creata da noi Miss.Dist..KM.: teniamo Miss.Dist..KM. per avere una nomenclatura il più omogenea possibile.

Dopo questo processo di selezione le variabili rimaste sono 17, si osserva se persiste una correlazione eccessiva tra alcune di loro:



Ci sono ancora delle variabili correlate tra loro, ma non in modo eccessivo. Tuttavia, quando si fittano dei modelli sensibili alla collinearità, occorre inserire l'opzione di preprocessing 'corr', poiché il problema non è completamente risolto.

Zero variance and near zero variance predictors

```
##                                freqRatio percentUnique zeroVar   nzv
## Absolute.Magnitude              1.000000    5.73927886   FALSE FALSE
## Epoch.Date.Close.Approach       1.058824   16.57776830   FALSE FALSE
## Relative.Velocity.km.per.sec    1.000000   99.72263708   FALSE FALSE
## Orbit.Uncertainity              1.932857    0.21335609   FALSE FALSE
## Minimum.Orbit.Intersection      1.000000   78.47237039   FALSE FALSE
## Eccentricity                    1.000000   78.64305526   FALSE FALSE
## Inclination                     1.000000   78.64305526   FALSE FALSE
## Asc.Node.Longitude              1.000000   78.49370600   FALSE FALSE
## Orbital.Period                  1.000000   78.53637721   FALSE FALSE
## Perihelion.Distance             1.000000   78.64305526   FALSE FALSE
## Perihelion.Arg                  1.000000   78.49370600   FALSE FALSE
## Perihelion.Time                 1.166667   29.61382547   FALSE FALSE
## Mean.Anomaly                    1.000000   78.55771282   FALSE FALSE
## Mean.Motion                     1.000000   78.57904843   FALSE FALSE
## Hazardous                       5.207947    0.04267122   FALSE FALSE
## Miss.Dist..KM.                  1.000000   99.42393855   FALSE FALSE
## Est.Dia.in.KM.                  1.000000    5.73927886   FALSE FALSE
```

Non si osservano variabili con problemi di Zero Variance o Near Zero Variance.

Data partition

Si estrae il 5% dei dati di partenza nel dataset score_data, al fine di effettuare lo score su nuovi dati. La parte di dataset rimanente, su cui verranno fittati i modelli e svolte le analisi, viene rinominata nasa. La partizione avviene con stratificazione rispetto al target.

```
dim(nasa)
```

```
## [1] 4454   17
```

```
dim(score_data)

## [1] 233  17

prop.table(table(score_data$Hazardous))

##        c0        c1
## 0.1587983 0.8412017
```

A sua volta si divide il dataset nasa in train e validation, con stratificazione per il target:

```
dim(train.df)

## [1] 3119   17

dim(test.df)

## [1] 1335   17

prop.table(table(train.df$Hazardous))

##        c0        c1
## 0.1612696 0.8387304

prop.table(table(test.df$Hazardous))

##        c0        c1
## 0.1610487 0.8389513
```

Modelli

Si è deciso di tunare i modelli massimizzando la Sensitivity, ovvero ridurre al minimo gli asteroidi classificati come non pericolosi che però in realtà sono pericolosi. Per il tuning di ogni modello si è utilizzato il seme casuale 1.

```
target <- nasa[, "Hazardous"]
```

1) Naive Bayes

Nonostante l'ipotesi di indipendenza tra le esplicative all'interno delle classi del target sia poco verosimile, si decide di provare a tunare ugualmente un modello Naive Bayes.

```
ctrl =trainControl(method="cv",number=10,classProbs=T,
                   summaryFunction=twoClassSummary)
naivebayes=train(Hazardous~.,data=train.df,method="naive_bayes",
                 preProcess=c("corr"),metric="Sens",trControl=ctrl,
                 tuneLength=5,na.action=na.pass)
naivebayes

## Naive Bayes
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
```

```
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   usekernel  ROC        Sens       Spec
##   FALSE      0.9815691  0.8786275  0.9675148
##    TRUE      0.9857286  0.9363922  0.9698005
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

Il modello è stato tunato con caret e non con klaR perché, nel dataset considerato, non si presentano i due problemi che caret non gestisce efficacemente: infatti, non vi sono variabili categoriali né missing values.

Per questo modello non è sensato fare il plot dei parametri di tuning, poiché non ce ne sono. L'unica decisione da effettuare è la scelta tra un kernel gaussiano ed uno empirico. Inoltre, non è stata applicata la correzione di Laplace, perché non vi sono frequenze congiunte nulle.

Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  486   63
##         c1   17 2553
##
##                Accuracy : 0.9744
##                  95% CI : (0.9682, 0.9796)
##                   Kappa : 0.9086
##
##             Sensitivity : 0.9662
##             Specificity : 0.9759
##          Pos Pred Value : 0.8852
##          Neg Pred Value : 0.9934
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  195   38
##         c1   20 1082
##
##                Accuracy : 0.9566
```

```
##                  95% CI : (0.9442, 0.9668)
##                   Kappa : 0.8445
##
##             Sensitivity : 0.9070
##             Specificity : 0.9661
##          Pos Pred Value : 0.8369
##          Neg Pred Value : 0.9819
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Si deduce che il modello non overfitti dal fatto che la sensitivity e la specificity sui dati di train e di test non differiscano per più del 10%.

2) Analisi discriminante (LDA)

```
Control = trainControl(method="cv",number=10,classProbs=TRUE,
                       summaryFunction=twoClassSummary)
lda <- train(Hazardous~.,data=train.df,method="lda",trControl=Control,
             metric="Sens",tuneLength=5)
lda
```

```
## Linear Discriminant Analysis
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results:
##
##   ROC       Sens       Spec
##   0.966619  0.7036863  0.9552836
```

Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  356  109
##         c1  147 2507
##
##                Accuracy : 0.9179
##                  95% CI : (0.9077, 0.9273)
##                   Kappa : 0.687
##
##             Sensitivity : 0.7078
##             Specificity : 0.9583
##          Pos Pred Value : 0.7656
##          Neg Pred Value : 0.9446
##              Prevalence : 0.1613
```

```
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  144   46
##         c1   71 1074
##
##                Accuracy : 0.9124
##                  95% CI : (0.8959, 0.927)
##                   Kappa : 0.6597
##
##             Sensitivity : 0.6698
##             Specificity : 0.9589
##          Pos Pred Value : 0.7579
##          Neg Pred Value : 0.9380
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Anche in questo modello sembra che non ci sia overfitting, tuttavia la Sensitivity non è molto soddisfacente.

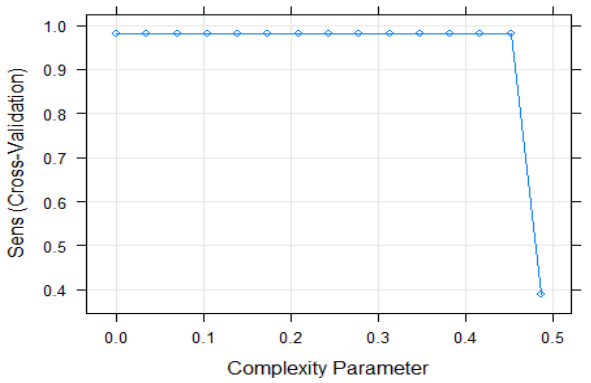
3) Albero tunato con Caret

```
cvCtrl <- trainControl(method="cv",number=10,search="grid",classProbs=TRUE,
                       summaryFunction=twoClassSummary)
tree_caret <- train(Hazardous~.,data=train.df,method="rpart",
              tuneLength=15,metric="Sens",trControl=cvCtrl)
tree_caret
```

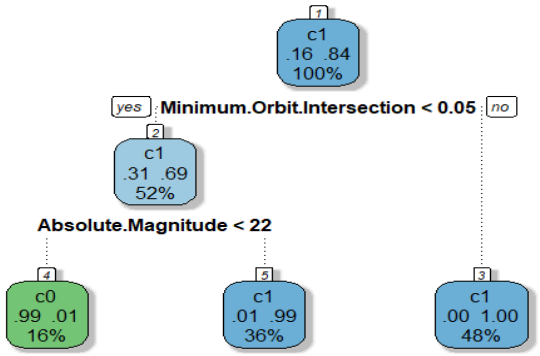
```
## CART
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens       Spec
##   0.00000000  0.9942816  0.9821569  0.9980872
##   0.03479125  0.9942816  0.9821569  0.9980872
##   0.06958250  0.9942816  0.9821569  0.9980872
##   0.10437376  0.9942816  0.9821569  0.9980872
##   0.13916501  0.9942816  0.9821569  0.9980872
##   0.17395626  0.9942816  0.9821569  0.9980872
##   0.20874751  0.9942816  0.9821569  0.9980872
##   0.24353877  0.9942816  0.9821569  0.9980872
```

```
##   0.27833002  0.9942816  0.9821569  0.9980872
##   0.31312127  0.9942816  0.9821569  0.9980872
##   0.34791252  0.9942816  0.9821569  0.9980872
##   0.38270378  0.9942816  0.9821569  0.9980872
##   0.41749503  0.9942816  0.9821569  0.9980872
##   0.45228628  0.9942816  0.9821569  0.9980872
##   0.48707753  0.6966907  0.3881176  0.9984689
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.4522863.
```

L'albero tunato è di tipo CART: prevede split binari che si basano sul massimo decremento dell'impurità misurata con l'indice di eterogeneità di Gini.



Visualizzazione grafica dell'albero decisionale:



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  495    5
##         c1    8 2611
##
##                Accuracy : 0.9958
##                  95% CI : (0.9929, 0.9978)
##                   Kappa : 0.9846

##             Sensitivity : 0.9841
##             Specificity : 0.9981
```

```
##          Pos Pred Value : 0.9900
##          Neg Pred Value : 0.9969
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

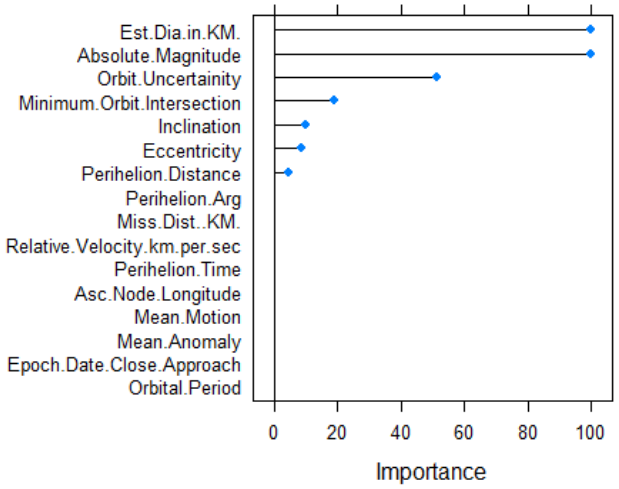
Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  211    4
##         c1    4 1116
##
##                Accuracy : 0.994
##                  95% CI : (0.9882, 0.9974)
##                   Kappa : 0.9778
##
##             Sensitivity : 0.9814
##             Specificity : 0.9964
##          Pos Pred Value : 0.9814
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

La sensitivity per il dataset di training e di validation non varia per più del 10%: il modello non overfitta.

Model selection

Estrazione delle variabili importanti:



Si possono considerare come importanti le variabili che hanno importanza superiore a 5.

```
##                              Overall
## Est.Dia.in.KM.               100.000
## Absolute.Magnitude           100.000
```

```
## Orbit.Uncertainity            51.570
## Minimum.Orbit.Intersection    18.690
## Inclination                    9.996
## Eccentricity                   8.719
## Perihelion.Distance            4.469
## Miss.Dist..KM.                 0.000
## Perihelion.Time                0.000
## Mean.Motion                    0.000
## Perihelion.Arg                 0.000
## Mean.Anomaly                   0.000
## Asc.Node.Longitude             0.000
## Orbital.Period                 0.000
## Epoch.Date.Close.Approach      0.000
## Relative.Velocity.km.per.sec   0.000
```

Il nuovo dataset selezionato è dati_tree_caret:

```
##   Absolute.Magnitude Eccentricity Est.Dia.in.KM. Inclination
## 1               21.6    0.4255491    0.127219878    6.025981
## 2               21.3    0.3516743    0.146067964   28.412996
## 3               20.3    0.3482483    0.231502122    4.237961
##   Minimum.Orbit.Intersection Orbit.Uncertainity target
## 1                  0.0252819                   5     c0
## 2                  0.1869350                   3     c1
## 3                  0.0430579                   0     c0
```

```
dim(dati_tree_caret)
```

```
## [1] 4454    7
```

Divisione di dati_tree_caret in training e test:

```
dim(dati_tree_caret_train)
```

```
## [1] 3119    7
```

```
dim(dati_tree_caret_test)
```

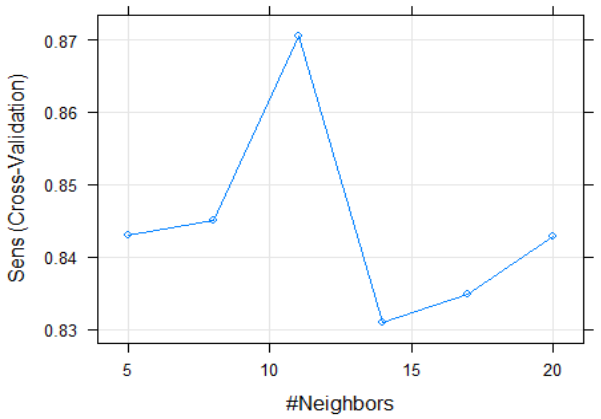
```
## [1] 1335    7
```

Le variabili selezionate in questo dataset possono essere usate per tunare modelli che richiedono model selection.

4) K-NN con variabili selezionate con Tree_caret

```
ctrl =trainControl(method="cv",number = 10,classProbs = T,
                   summaryFunction=twoClassSummary)
grid = expand.grid(k=seq(5, 20, 3))
knn_tree_caret=train(target~.,data=dati_tree_caret_train,method = "knn",
          trControl = ctrl,tuneLength=5,metric="Sens",na.action = na.pass,
          tuneGrid=grid,preProcess=c("scale", "corr"))
knn_tree_caret
```

```
## k-Nearest Neighbors
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.9801899  0.8429412  0.9732430
##    8  0.9836026  0.8449804  0.9701895
##   11  0.9849763  0.8705882  0.9751543
##   14  0.9847794  0.8309020  0.9743909
##   17  0.9856232  0.8348627  0.9763037
##   20  0.9848862  0.8428235  0.9728584
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was k = 11.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  445   48
##         c1   58 2568
##
##                Accuracy : 0.966
##                  95% CI : (0.959, 0.9721)
##                   Kappa : 0.8734
##
##             Sensitivity : 0.8847
##             Specificity : 0.9817
##          Pos Pred Value : 0.9026
##          Neg Pred Value : 0.9779
##              Prevalence : 0.1613
```

```
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  180   21
##         c1   35 1099
##
##                Accuracy : 0.9581
##                  95% CI : (0.9459, 0.9682)
##                   Kappa : 0.8406
##
##             Sensitivity : 0.8372
##             Specificity : 0.9812
##          Pos Pred Value : 0.8955
##          Neg Pred Value : 0.9691
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

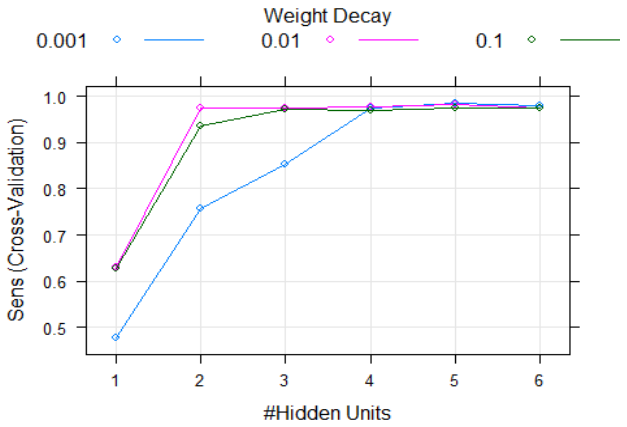
Il modello non overfitta, perché le metriche Sensitivity e Specificity su dataset di training e di test non differiscono di molto.

5) Neural Network con variabili selezionate da Tree_caret

```
ctrl = trainControl(method="cv",number=10,search="grid",classProbs=TRUE,
                    summaryFunction=twoClassSummary)
tunegrid <- expand.grid(size=c(1:6), decay=c(0.001, 0.01, 0.1))
nnet_tree_caret <- train(target~.,data=dati_tree_caret_train,method="nnet",
                   metric="Sens",tuneGrid=tunegrid,
                   preProcess=c("scale","BoxCox", "corr"),trControl=ctrl,
                   trace=F, maxit=250)
nnet_tree_caret
```

```
## Neural Network
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (6), Box-Cox transformation (5)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0.001  0.7871777  0.4767059  0.9797403
##   1     0.010  0.8600727  0.6302745  0.9782092
##   1     0.100  0.9466241  0.6281569  0.9491518
##   2     0.001  0.8936195  0.7561176  0.9923430
##   2     0.010  0.9996749  0.9761569  0.9965546
```

```
##   2     0.100  0.9950837  0.9362353  0.9923401
##   3     0.001  0.9210648  0.8521961  0.9931151
##   3     0.010  0.9987548  0.9742745  0.9969393
##   3     0.100  0.9996358  0.9722353  0.9969378
##   4     0.001  0.9996375  0.9761961  0.9965561
##   4     0.010  0.9994793  0.9781176  0.9957942
##   4     0.100  0.9996274  0.9702353  0.9969378
##   5     0.001  0.9985395  0.9861569  0.9969393
##   5     0.010  0.9997942  0.9841176  0.9961744
##   5     0.100  0.9995981  0.9741961  0.9965561
##   6     0.001  0.9985535  0.9801569  0.9961759
##   6     0.010  0.9995989  0.9761961  0.9973209
##   6     0.100  0.9995745  0.9761961  0.9965561
##
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were size = 5 and decay = 0.001.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  454   29
##         c1   49 2587
##
##                Accuracy : 0.975
##                  95% CI : (0.9689, 0.9802)
##                   Kappa : 0.906
##
##             Sensitivity : 0.9026
##             Specificity : 0.9889
##          Pos Pred Value : 0.9400
##          Neg Pred Value : 0.9814
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  187   11
##         c1   28 1109
##
##                Accuracy : 0.9708
##                  95% CI : (0.9603, 0.9791)
##                   Kappa : 0.8883
##
##             Sensitivity : 0.8698
##             Specificity : 0.9902
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9754
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

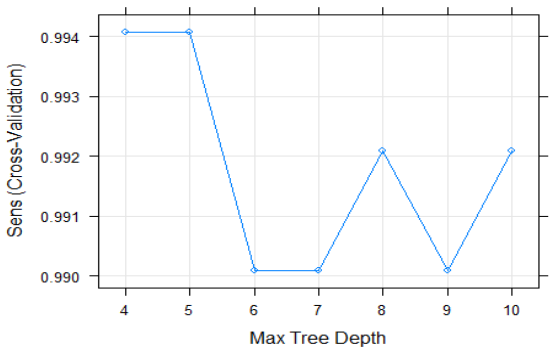
Anche questo modello non overfitta, per le stesse ragioni di quelli precedenti.

6) Gradient Boosting con variabili selezionate da Tree_Caret

```
control <- trainControl(method="cv",number=10,summaryFunction=twoClassSummary,
                        classProbs=TRUE,search="grid")
grad_boost_tree_caret <- train(target~.,data=dati_tree_caret_train,
                         method="gbm",trControl=control,metric="Sens",
                         verbose=FALSE,
                         tuneGrid=data.frame(interaction.depth=c(4:10),
                                             n.trees=150,
                                             shrinkage=0.05,
                                             n.minobsinnode=50))
grad_boost_tree_caret
```

```
## Stochastic Gradient Boosting
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  ROC        Sens       Spec
##    4                 0.9970877  0.9940784  0.9973224
##    5                 0.9992132  0.9940784  0.9973224
##    6                 0.9978886  0.9900784  0.9969393
##    7                 0.9991982  0.9900784  0.9973224
##    8                 0.9995351  0.9920784  0.9977055
##    9                 0.9993329  0.9900784  0.9973224
##   10                 0.9984495  0.9920784  0.9969393
```

```
##
## Tuning parameter 'n.trees' was held constant at a value of 150
## Tuning
##  parameter 'shrinkage' was held constant at a value of 0.05
## Tuning
##  parameter 'n.minobsinnode' was held constant at a value of 50
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  4, shrinkage = 0.05 and n.minobsinnode = 50.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  502    2
##         c1    1 2614
##
##                Accuracy : 0.999
##                  95% CI : (0.9972, 0.9998)
##                   Kappa : 0.9964
##
##             Sensitivity : 0.9980
##             Specificity : 0.9992
##          Pos Pred Value : 0.9960
##          Neg Pred Value : 0.9996
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

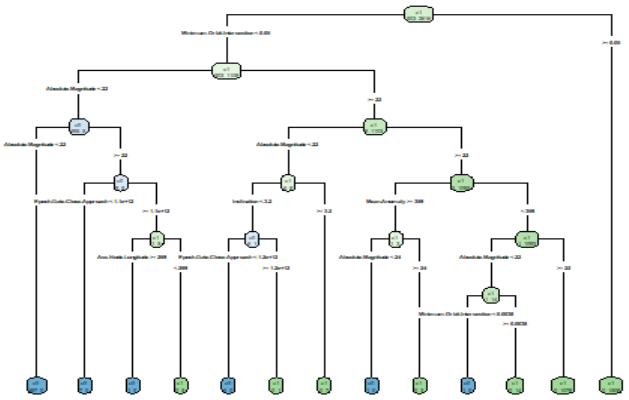
```
##           Reference
## Prediction   c0   c1
##         c0  211    2
##         c1    4 1118
##
##                Accuracy : 0.9955
##                  95% CI : (0.9902, 0.9983)
##                   Kappa : 0.9833
##
##             Sensitivity : 0.9814
```

```
##             Specificity : 0.9982
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

7) Albero tunato con rpart

```
library(rpart)
tree_rp <- rpart(Hazardous~.,data=train.df, method="class", cp=0,
                 minsplit=1)
tree_rp
```

Visualizzazione grafica dell'albero:

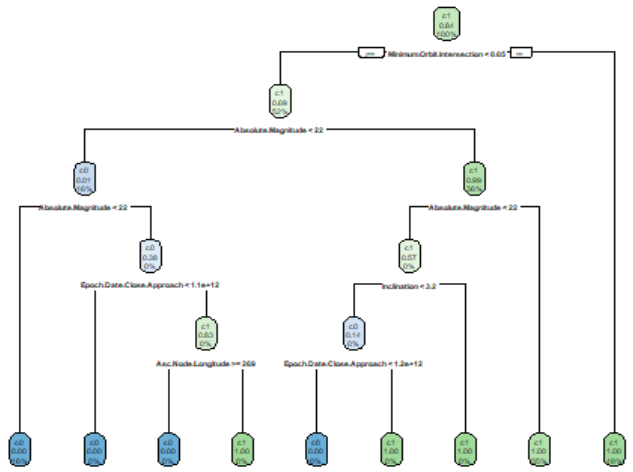


```
tree_rp$cptable
```

```
##             CP nsplit   rel error     xerror        xstd
## 1 0.4870775348      0 1.000000000 1.00000000 0.040834495
## 2 0.0049701789      2 0.025844930 0.03976143 0.008862375
## 3 0.0039761431      4 0.015904573 0.03578529 0.008410300
## 4 0.0019880716      6 0.007952286 0.03180915 0.007931863
## 5 0.0009940358      8 0.003976143 0.02186879 0.006582050
## 6 0.0000000000     12 0.000000000 0.02186879 0.006582050
```

Con questo metodo, l'albero considerato migliore è quello con cp 0.0009940358, 8 split e 9 livelli.

```
best_pruned <- prune(tree_rp, cp = 0.0009940358)
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  501    0
##         c1    2 2616
##
##                Accuracy : 0.9994
##                  95% CI : (0.9977, 0.9999)
##                   Kappa : 0.9976
##
##             Sensitivity : 0.9960
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9992
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  211    2
##         c1    4 1118
##
##                Accuracy : 0.9955
##                  95% CI : (0.9902, 0.9983)
##                   Kappa : 0.9833
##
##             Sensitivity : 0.9814
##             Specificity : 0.9982
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Il modello non overfitta, perché vi è poca differenza tra le metriche di Specificity e Sensitivity nei due dataset.

Model selection

Estrazione delle variabili importanti:

```
Importanza_var_tree_rp <- tree_rp$variable.importance
Importanza_var_tree_rp
```

```
##           Absolute.Magnitude               Est.Dia.in.KM.
##                   738.258743                   736.758743
##           Orbit.Uncertainity                  Inclination
##                   398.963302                   180.045404
##   Minimum.Orbit.Intersection Relative.Velocity.km.per.sec
##                   159.648270                    80.097109
##                 Eccentricity          Perihelion.Distance
##                    73.546797                    57.017861
##    Epoch.Date.Close.Approach           Asc.Node.Longitude
##                     8.405547                     3.910256
##                 Mean.Anomaly
##                     2.738127
```

La variabile che ha importanza maggiore è Absolute.Magnitude, si riscalano tutte le importanze rispetto a quella.

```
importanza <- tree_rp$variable.importance/738.258743*100
Importanza_var_tree_rp_100 <- as.data.frame(importanza)
Importanza_var_tree_rp_100
```

```
##                               importanza
## Absolute.Magnitude           100.0000000
## Est.Dia.in.KM.                99.7968192
## Orbit.Uncertainity            54.0411212
## Inclination                   24.3878458
## Minimum.Orbit.Intersection    21.6249752
## Relative.Velocity.km.per.sec  10.8494630
## Eccentricity                   9.9621979
## Perihelion.Distance            7.7232896
## Epoch.Date.Close.Approach      1.1385638
## Asc.Node.Longitude             0.5296593
## Mean.Anomaly                   0.3708899
```

Si considerano come importanti le variabili che hanno importanza maggiore di 10. Quelle selezionate per il dataset dati_tree_rp sono le seguenti:

```
##   Absolute.Magnitude Est.Dia.in.KM. Orbit.Uncertainity Inclination
## 1               21.6    0.127219878                  5    6.025981
## 2               21.3    0.146067964                  3   28.412996
## 3               20.3    0.231502122                  0    4.237961
##   Minimum.Orbit.Intersection Relative.Velocity.km.per.sec target
## 1                  0.0252819                     6.115834     c0
```

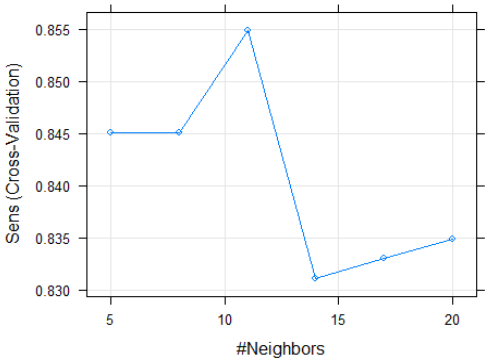
```
## 2                  0.1869350                    18.113985     c1
## 3                  0.0430579                     7.590711     c0

dim(dati_tree_rp)

## [1] 4454    7
```

Divisione di dati_tree_rp in training e test:

```
dim(dati_tree_rp_train)

## [1] 3119    7

dim(dati_tree_rp_test)

## [1] 1335    7
```

8) KNN con variabili selezionate con Tree_rp

```
ctrl =trainControl(method="cv",number=10,classProbs=T,
                   summaryFunction=twoClassSummary)
grid = expand.grid(k=seq(5, 20, 3))
knn_tree_rp=train(target~.,data=dati_tree_rp_train,method="knn",
          trControl=ctrl,tuneLength=5,metric="Sens",na.action=na.pass,
          tuneGrid=grid,preProcess=c("scale", "corr"))
knn_tree_rp

## k-Nearest Neighbors
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.9838541  0.8450196  0.9740107
##    8  0.9852653  0.8450196  0.9736319
##   11  0.9866247  0.8549020  0.9724884
##   14  0.9864878  0.8310588  0.9721082
##   17  0.9858852  0.8330588  0.9717177
##   20  0.9861757  0.8349020  0.9732430
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was k = 11.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  447   50
##         c1   56 2566
##
##                Accuracy : 0.966
##                  95% CI : (0.959, 0.9721)
##                   Kappa : 0.8738
##
##             Sensitivity : 0.8887
##             Specificity : 0.9809
##          Pos Pred Value : 0.8994
##          Neg Pred Value : 0.9786
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  183   29
##         c1   32 1091
##
##                Accuracy : 0.9543
##                  95% CI : (0.9417, 0.9649)
##                   Kappa : 0.8299
##
##             Sensitivity : 0.8512
##             Specificity : 0.9741
##          Pos Pred Value : 0.8632
##          Neg Pred Value : 0.9715
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

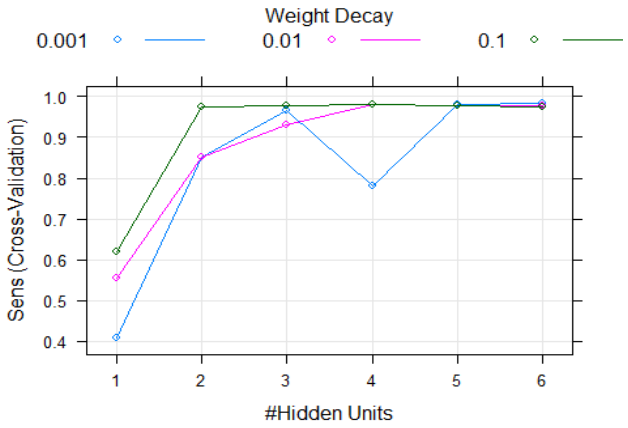
Anche questo modello non presenta overfitting.

9) Neural Network con variabili selezionate con Tree_rp

```
ctrl = trainControl(method="cv",number=10,search="grid",
                    classProbs=TRUE,summaryFunction=twoClassSummary)
tunegrid <- expand.grid(size=c(1:6), decay=c(0.001, 0.01, 0.1))
nnet_tree_rp <- train(target~.,data=dati_tree_rp_train,method="nnet",
                  metric="Sens",tuneGrid=tunegrid,
                  preProcess=c("scale","BoxCox", "corr"),
                  trControl=ctrl,trace=F,maxit=250)
nnet_tree_rp
```

```
## Neural Network
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (6), Box-Cox transformation (5)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0.001  0.7649764  0.4085098  0.9847153
##   1     0.010  0.7613909  0.5564706  0.9808882
##   1     0.100  0.9459380  0.6201176  0.9460955
##   2     0.001  0.9455655  0.8503922  0.9912082
##   2     0.010  0.9395378  0.8521569  0.9935012
##   2     0.100  0.9997268  0.9742745  0.9961715
##   3     0.001  0.9956586  0.9664706  0.9961730
##   3     0.010  0.9959825  0.9301961  0.9927335
##   3     0.100  0.9997119  0.9762745  0.9965546
##   4     0.001  0.9318586  0.7821961  0.9984704
##   4     0.010  0.9997113  0.9801961  0.9973209
##   4     0.100  0.9997185  0.9801961  0.9965561
##   5     0.001  0.9996529  0.9801961  0.9973209
##   5     0.010  0.9996138  0.9782353  0.9954111
##   5     0.100  0.9996506  0.9762353  0.9954067
##   6     0.001  0.9991165  0.9841569  0.9946448
##   6     0.010  0.9995535  0.9781961  0.9973224
##   6     0.100  0.9995977  0.9742745  0.9954067
##
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were size = 6 and decay = 0.001.
```

Il modello che la procedura indica come migliore è quello con 6 neuroni nascosti e tasso di apprendimento 0.001. Il modello con 5 neuroni nascosti e tasso di apprendimento 0.001, tuttavia, ha performance di poco inferiori ma una complessità minore, quindi potremmo tenerlo in considerazione.



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  503    2
##         c1    0 2614
##
##               Accuracy : 0.9994
##                 95% CI : (0.9977, 0.9999)
##                  Kappa : 0.9976
##
##            Sensitivity : 1.0000
##            Specificity : 0.9992
##         Pos Pred Value : 0.9960
##         Neg Pred Value : 1.0000
##             Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

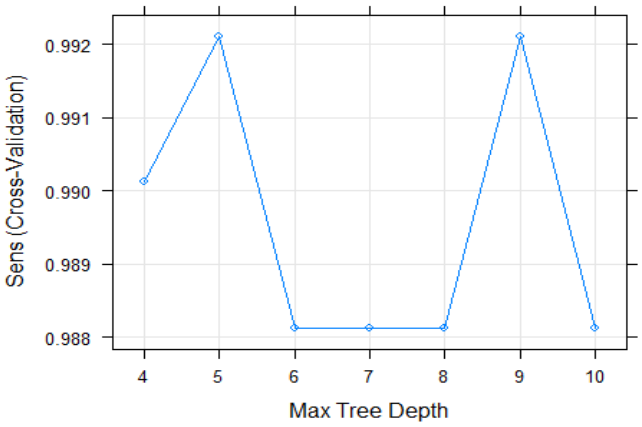
```
##           Reference
## Prediction   c0   c1
##         c0  211    3
##         c1    4 1117
##
##               Accuracy : 0.9948
##                 95% CI : (0.9892, 0.9979)
##                  Kappa : 0.9806
##
##            Sensitivity : 0.9814
##            Specificity : 0.9973
##         Pos Pred Value : 0.9860
##         Neg Pred Value : 0.9964
##             Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Il modello non overfitta, per le stesse motivazioni date in precedenza.

10) Gradient Boosting con variabili selezionate da Tree_rp

```
control <- trainControl(method="cv",number=10,summaryFunction=twoClassSummary,
                        classProbs=TRUE,search="grid")
grad_boost_tree_rp <- train(target~.,data=dati_tree_rp_train,method="gbm",
                            trControl=control,metric="Sens",verbose=FALSE,
                            tuneGrid=data.frame(interaction.depth=c(4:10),
                                                n.trees=150,
                                                shrinkage=0.05,
                                                n.minobsinnode=50))
grad_boost_tree_rp
```

```
## Stochastic Gradient Boosting
##
## 3119 samples
##    6 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  ROC        Sens       Spec
##    4                 0.9980755  0.9901176  0.9973224
##    5                 0.9997069  0.9921176  0.9973224
##    6                 0.9985993  0.9881176  0.9973224
##    7                 0.9995273  0.9881176  0.9977055
##    8                 0.9993179  0.9881176  0.9973224
##    9                 0.9995872  0.9921176  0.9973224
##   10                 0.9997668  0.9881176  0.9973224
##
## Tuning parameter 'n.trees' was held constant at a value of 150
## Tuning
##  parameter 'shrinkage' was held constant at a value of 0.05
## Tuning
##  parameter 'n.minobsinnode' was held constant at a value of 50
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  5, shrinkage = 0.05 and n.minobsinnode = 50.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  502    2
##         c1    1 2614
##
##                Accuracy : 0.999
##                  95% CI : (0.9972, 0.9998)
##                   Kappa : 0.9964
##
##             Sensitivity : 0.9980
##             Specificity : 0.9992
##          Pos Pred Value : 0.9960
##          Neg Pred Value : 0.9996
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  211    2
##         c1    4 1118
##
##                Accuracy : 0.9955
##                  95% CI : (0.9902, 0.9983)
##                   Kappa : 0.9833
##
##             Sensitivity : 0.9814
##             Specificity : 0.9982
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

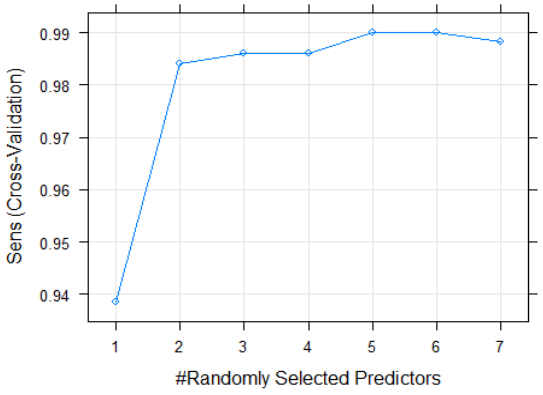
11) Random Forest

Il numero di variabili da considerare deve essere tunato intorno alla radice quadrata del numero di covariate del dataset. Nel nostro dataset le covariate sono 16, quindi verrà usato come riferimento il valore 4.

```
control <- trainControl(method="cv",number=10,search="grid",
                        summaryFunction=twoClassSummary,classProbs=TRUE)
tunegrid <- expand.grid(.mtry=c(1:7))
random_forest <- train(Hazardous~.,metric="Sens",data=train.df,
                       method="rf",tuneGrid=tunegrid,ntree=250,
                       trControl=control)
random_forest
```

```
## Random Forest
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   1     0.9992894  0.9384314  0.9996169
##   2     0.9995410  0.9841176  0.9977055
##   3     0.9997289  0.9861176  0.9977055
##   4     0.9996580  0.9861176  0.9977055
##   5     0.9994825  0.9900784  0.9977055
##   6     0.9994265  0.9901176  0.9980872
##   7     0.9997902  0.9881569  0.9977055
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

Il modello migliore è quello che considera sottoinsiemi da 6 covariate ciascuno.



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  503    0
##         c1    0 2616
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##                   Kappa : 1
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

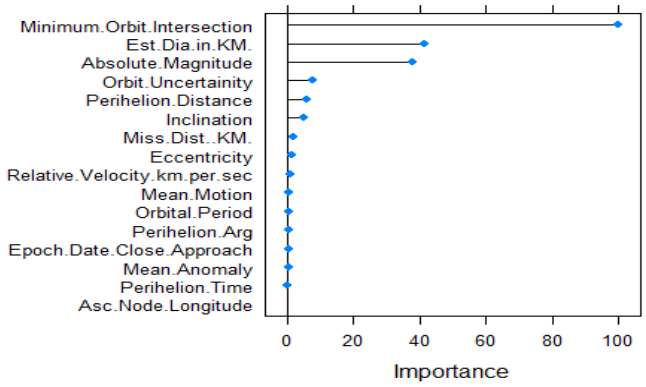
Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  212    2
##         c1    3 1118
##
##                Accuracy : 0.9963
##                  95% CI : (0.9913, 0.9988)
##                   Kappa : 0.9861
##
##             Sensitivity : 0.9860
##             Specificity : 0.9982
##          Pos Pred Value : 0.9907
##          Neg Pred Value : 0.9973
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Anche questo modello non overfitta sui dati di training.

Model selection

Estrazione delle variabili importanti:



Si considerano come importanti le variabili che hanno importanza maggiore di 4.

```
##                                 Overall
## Minimum.Orbit.Intersection   100.00000
## Est.Dia.in.KM.                41.28920
## Absolute.Magnitude            37.67462
## Orbit.Uncertainity             7.81879
## Perihelion.Distance            5.63619
## Inclination                    4.89805
## Miss.Dist..KM.                 1.91100
## Eccentricity                   1.21528
## Relative.Velocity.km.per.sec   0.76269
## Mean.Motion                    0.43015
## Orbital.Period                 0.31847
## Perihelion.Arg                 0.24040
## Epoch.Date.Close.Approach      0.18614
## Mean.Anomaly                   0.18381
## Perihelion.Time                0.05957
## Asc.Node.Longitude             0.00000
```

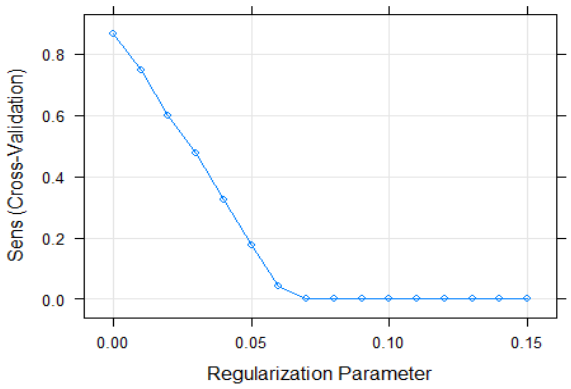
Anche se hanno valori di importanza differenti, si tratta delle stesse variabili considerate importanti dall'albero tunato con Caret.

12) Lasso

```
ctrl = trainControl(method="cv",number=10,classProbs=T,
                    summaryFunction=twoClassSummary)
grid = expand.grid(.alpha=1,.lambda=seq(0, 0.15, by = 0.01))
lasso=train(Hazardous~.,data=train.df,method="glmnet",trControl=ctrl,
            tuneLength=5,metric="Sens",preProcess="corr",
            na.action=na.pass,tuneGrid=grid)
lasso
```

```
## glmnet
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing:  (None)
```

```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   lambda  ROC        Sens         Spec
##   0.00    0.9899393  0.868470588  0.9736334
##   0.01    0.9845113  0.749333333  0.9812699
##   0.02    0.9779170  0.598274510  0.9873870
##   0.03    0.9739295  0.477176471  0.9896815
##   0.04    0.9703283  0.326196078  0.9946506
##   0.05    0.9665578  0.174941176  0.9977085
##   0.06    0.9630630  0.041725490  0.9988535
##   0.07    0.9587834  0.001960784  0.9996169
##   0.08    0.9538865  0.000000000  1.0000000
##   0.09    0.9464862  0.000000000  1.0000000
##   0.10    0.9289817  0.000000000  1.0000000
##   0.11    0.8368156  0.000000000  1.0000000
##   0.12    0.7610010  0.000000000  1.0000000
##   0.13    0.5000000  0.000000000  1.0000000
##   0.14    0.5000000  0.000000000  1.0000000
##   0.15    0.5000000  0.000000000  1.0000000
##
## Tuning parameter 'alpha' was held constant at a value of 1
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.
```



Matrice di confusione sui dati di train:

```
##          Reference
## Prediction   c0   c1
##         c0  438   64
##         c1   65 2552
##
##                Accuracy : 0.9586
##                  95% CI : (0.951, 0.9654)
##                   Kappa : 0.847
##
##             Sensitivity : 0.8708
##             Specificity : 0.9755
```

```
##          Pos Pred Value : 0.8725
##          Neg Pred Value : 0.9752
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  177   29
##         c1   38 1091
##
##                Accuracy : 0.9498
##                  95% CI : (0.9367, 0.9609)
##                   Kappa : 0.8111
##
##             Sensitivity : 0.8233
##             Specificity : 0.9741
##          Pos Pred Value : 0.8592
##          Neg Pred Value : 0.9663
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Coefficienti stimati dal modello:

```
## (Intercept)                  -316.53734592758127064371
## Absolute.Magnitude              3.49203419596397024449
## Epoch.Date.Close.Approach      -0.00000000000007643802
## Relative.Velocity.km.per.sec   -0.00090332283514547385
## Orbit.Uncertainity              0.15214747181411658605
## Minimum.Orbit.Intersection    133.93753106512326667143
## Eccentricity                    0.11061373937969694314
## Inclination                    -0.01664011309423680043
## Asc.Node.Longitude             -0.00010898182069306085
## Orbital.Period                  .
## Perihelion.Distance            -0.38115179224426914972
## Perihelion.Arg                 -0.00022325933101719731
## Perihelion.Time                 0.00009575922776293261
## Mean.Anomaly                    0.00065730010321207949
## Mean.Motion                    -0.09310311453086474176
## Miss.Dist..KM.                  0.00000000685161841732
## Est.Dia.in.KM.                 15.88201644250998789687
```

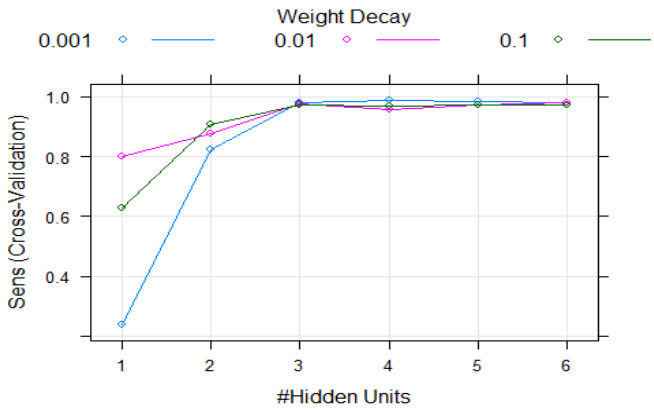
Sono stati creati due dataset, dati_lasso_train e dati_lasso_test, con le sole variabili i cui coefficienti risultano non prossimi a zero.

13) Neural Network su variabili selezionate con Lasso

```
ctrl = trainControl(method="cv",number=10,search="grid",classProbs=TRUE,
                    summaryFunction=twoClassSummary)
tunegrid <- expand.grid(size=c(1:6), decay=c(0.001, 0.01, 0.1))
```

```
nnet_lasso <- train(Hazardous~.,data=dati_lasso_train,method="nnet",
                    metric="Sens",tuneGrid=tunegrid,
                    preProcess=c("scale","BoxCox", "corr"),
                    trControl=ctrl,trace=F,maxit=200)
nnet_lasso
```

```
## Neural Network
##
## 3119 samples
##    8 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (8), Box-Cox transformation (7)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0.001  0.6200047  0.2383137  0.9908221
##   1     0.010  0.9796284  0.7988627  0.9743938
##   1     0.100  0.9460915  0.6281176  0.9468588
##   2     0.001  0.9616090  0.8243922  0.9854889
##   2     0.010  0.9229142  0.8762745  0.9954081
##   2     0.100  0.9889666  0.9062353  0.9896669
##   3     0.001  0.9983947  0.9782353  0.9946477
##   3     0.010  0.9995231  0.9761569  0.9969393
##   3     0.100  0.9996124  0.9702353  0.9965561
##   4     0.001  0.9995908  0.9881176  0.9957942
##   4     0.010  0.9970362  0.9581569  0.9931225
##   4     0.100  0.9995745  0.9662353  0.9961744
##   5     0.001  0.9969010  0.9841569  0.9961730
##   5     0.010  0.9995764  0.9722353  0.9961730
##   5     0.100  0.9995666  0.9702353  0.9961744
##   6     0.001  0.9993376  0.9802353  0.9957928
##   6     0.010  0.9995525  0.9782353  0.9969378
##   6     0.100  0.9996341  0.9722353  0.9969407
##
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were size = 4 and decay = 0.001.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  502    1
##         c1    1 2615
##
##                Accuracy : 0.9994
##                  95% CI : (0.9977, 0.9999)
##                   Kappa : 0.9976
##
##             Sensitivity : 0.9980
##             Specificity : 0.9996
##          Pos Pred Value : 0.9980
##          Neg Pred Value : 0.9996
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

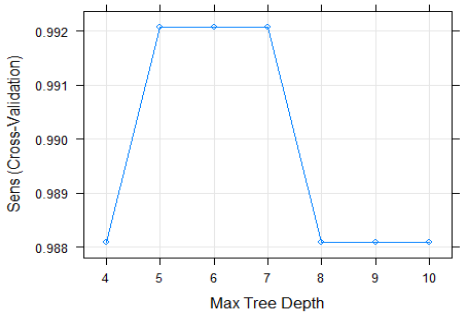
```
##           Reference
## Prediction   c0   c1
##         c0  211    3
##         c1    4 1117
##
##                Accuracy : 0.9948
##                  95% CI : (0.9892, 0.9979)
##                   Kappa : 0.9806
##
##             Sensitivity : 0.9814
##             Specificity : 0.9973
##          Pos Pred Value : 0.9860
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

14) Gradient Boosting con variabili selezionate con Lasso

```
control <- trainControl(method="cv",number=10,summaryFunction=twoClassSummary,
                        classProbs=TRUE,search="grid")
grad_boost_lasso <- train(Hazardous~.,data=dati_lasso_train,method="gbm",
                          trControl=control,metric="Sens",verbose=FALSE,
                          tuneGrid=data.frame(interaction.depth=c(4:10),
                                              n.trees=150,
                                              shrinkage=0.05,
                                              n.minobsinnode=50))
grad_boost_lasso
```

```
## Stochastic Gradient Boosting
##
## 3119 samples
```

```
##    8 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  ROC        Sens       Spec
##    4                 0.9970129  0.9880784  0.9973224
##    5                 0.9971927  0.9920784  0.9980872
##    6                 0.9982630  0.9920784  0.9977055
##    7                 0.9978436  0.9920784  0.9973224
##    8                 0.9993255  0.9880784  0.9977055
##    9                 0.9988914  0.9880784  0.9973224
##   10                 0.9997071  0.9880784  0.9977055
##
## Tuning parameter 'n.trees' was held constant at a value of 150
## Tuning
##  parameter 'shrinkage' was held constant at a value of 0.05
## Tuning
##  parameter 'n.minobsinnode' was held constant at a value of 50
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  5, shrinkage = 0.05 and n.minobsinnode = 50.
```



Matrice di confusione sui dati di train:

```
##          Reference
## Prediction   c0   c1
##         c0  502    2
##         c1    1 2614
##
##                Accuracy : 0.999
##                  95% CI : (0.9972, 0.9998)
##                   Kappa : 0.9964
##
##             Sensitivity : 0.9980
##             Specificity : 0.9992
##          Pos Pred Value : 0.9960
##          Neg Pred Value : 0.9996
##              Prevalence : 0.1613
```

```
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  211    1
##         c1    4 1119
##
##                Accuracy : 0.9963
##                  95% CI : (0.9913, 0.9988)
##                   Kappa : 0.9861
##
##             Sensitivity : 0.9814
##             Specificity : 0.9991
##          Pos Pred Value : 0.9953
##          Neg Pred Value : 0.9964
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

15) Partial Least Squares Regression (PLS)

Questo modello cerca delle combinazioni lineari delle variabili di input che massimizzino la covarianza al quadrato tra ciascuna combinazione lineare e la variabile target. Può essere usato come classificatore in sé oppure come model selector, per trovare un sottoinsieme di variabili per tunare altri modelli. La model selection si fa col valore del VIP. Il tuning verrà effettuato sia con le variabili standardizzate che con le variabili originali. Il modello che risulterà migliore tra i due verrà utilizzato per fare model selection.

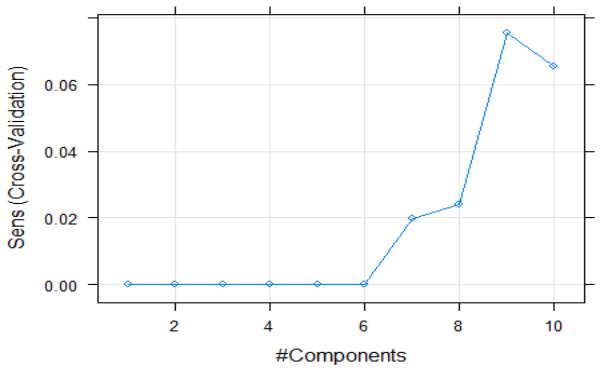
Tuning con le variabili originali:

```
Control=trainControl(method="cv",number=10,classProbs=TRUE,
                     summaryFunction=twoClassSummary,search="grid")
pls=train(Hazardous~.,data=train.df,method="pls",trControl=Control,
          metric="Sens",tuneLength=10)
pls
```

```
## Partial Least Squares
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   ncomp  ROC        Sens        Spec
##    1     0.5518414  0.00000000  1.0000000
```

```
##     2      0.5501949  0.00000000  1.0000000
##     3      0.5532777  0.00000000  1.0000000
##     4      0.5480065  0.00000000  1.0000000
##     5      0.5453332  0.00000000  1.0000000
##     6      0.6223374  0.00000000  1.0000000
##     7      0.7519493  0.01988235  0.9931210
##     8      0.7777220  0.02392157  0.9915928
##     9      0.8028322  0.07549020  0.9870054
##    10      0.8346838  0.06545098  0.9870068
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 9.
```

La Sensitivity è particolarmente bassa.



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0   40   31
##         c1  463 2585
##
##                Accuracy : 0.8416
##                  95% CI : (0.8283, 0.8543)
##                   Kappa : 0.1036
##
##             Sensitivity : 0.07952
##             Specificity : 0.98815
##          Pos Pred Value : 0.56338
##          Neg Pred Value : 0.84810
##              Prevalence : 0.16127
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0   26   18
##         c1  189 1102
##
```

```
##                Accuracy : 0.8449
##                  95% CI : (0.8244, 0.864)
##                   Kappa : 0.1545
##
##             Sensitivity : 0.12093
##             Specificity : 0.98393
##          Pos Pred Value : 0.59091
##          Neg Pred Value : 0.85360
##              Prevalence : 0.16105
##
##        'Positive' Class : c0
```

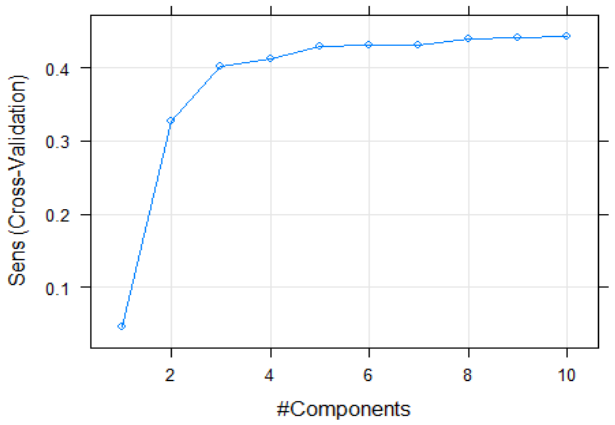
Sui dati di validation la Sensitivity risulta leggermente più elevata, tuttavia non è ancora soddisfacente.

Tuning con le variabili standardizzate:

```
Control=trainControl(method="cv",number=10,classProbs=TRUE,
                     summaryFunction=twoClassSummary,search="grid")
pls_stand=train(Hazardous~.,data=train.df,method="pls",trControl=Control,
          preProcess=c("scale", "BoxCox"), metric="Sens",tuneLength=10)
pls_stand
```

```
## Partial Least Squares
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: scaled (16), Box-Cox transformation (15)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   ncomp  ROC        Sens        Spec
##    1     0.8561535  0.04564706  0.9862405
##    2     0.9191688  0.32607843  0.9820318
##    3     0.9326614  0.40176471  0.9824150
##    4     0.9389094  0.41156863  0.9793557
##    5     0.9410057  0.42941176  0.9793557
##    6     0.9412400  0.43133333  0.9789711
##    7     0.9412226  0.43133333  0.9770583
##    8     0.9410511  0.43929412  0.9759133
##    9     0.9425103  0.44133333  0.9797359
##   10     0.9446845  0.44329412  0.9797330
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 10.
```

La Sensitivity è migliore rispetto a quella del modello precedente, anche se ancora è molto deludente.



Matrice di confusione sui dati di train:

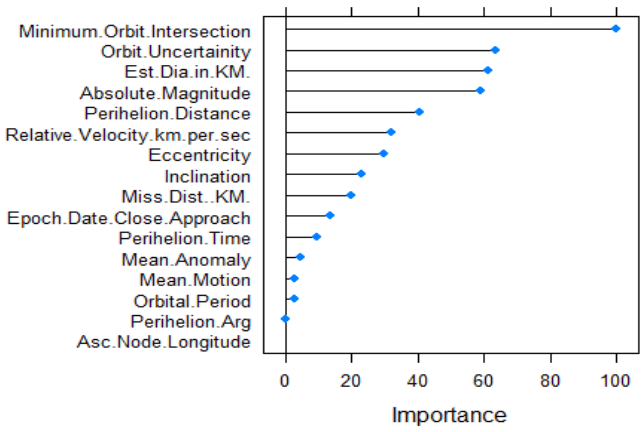
```
##            Reference
## Prediction   c0   c1
##         c0  227   53
##         c1  276 2563
##
##                Accuracy : 0.8945
##                  95% CI : (0.8832, 0.9051)
##                   Kappa : 0.525
##
##             Sensitivity : 0.45129
##             Specificity : 0.97974
##          Pos Pred Value : 0.81071
##          Neg Pred Value : 0.90278
##              Prevalence : 0.16127
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##            Reference
## Prediction   c0   c1
##         c0   93   21
##         c1  122 1099
##
##                Accuracy : 0.8929
##                  95% CI : (0.875, 0.909)
##                   Kappa : 0.5107
##
##             Sensitivity : 0.43256
##             Specificity : 0.98125
##          Pos Pred Value : 0.81579
##          Neg Pred Value : 0.90008
##              Prevalence : 0.16105
##
##        'Positive' Class : c0
```

Model selection

Si selezionano le variabili importanti dal PLS con le variabili standardizzate.



Importanza delle variabili:

```
##                               Overall
## Minimum.Orbit.Intersection   100.00000
## Orbit.Uncertainity            63.37218
## Est.Dia.in.KM.                61.19991
## Absolute.Magnitude            58.98435
## Perihelion.Distance           40.51815
## Relative.Velocity.km.per.sec  32.02143
## Eccentricity                  29.79941
## Inclination                   23.08771
## Miss.Dist..KM.                19.61752
## Epoch.Date.Close.Approach     13.29192
## Perihelion.Time                9.19605
## Mean.Anomaly                   4.55292
## Mean.Motion                    2.64614
## Orbital.Period                 2.64614
## Perihelion.Arg                 0.02474
## Asc.Node.Longitude             0.00000
```

Si considerano, per il dataset dati_pls_stand, le variabili che hanno importanza maggiore di 10.

```
##   Absolute.Magnitude Epoch.Date.Close.Approach Relative.Velocity.km.per.sec
## 1               21.6              788947200000                     6.115834
## 2               21.3              788947200000                    18.113985
## 3               20.3              789552000000                     7.590711
##   Orbit.Uncertainity Minimum.Orbit.Intersection Eccentricity Inclination
## 1                  5                  0.0252819    0.4255491    6.025981
## 2                  3                  0.1869350    0.3516743   28.412996
## 3                  0                  0.0430579    0.3482483    4.237961
##   Perihelion.Distance Miss.Dist..KM. Est.Dia.in.KM. target
## 1           0.8082589       62753535    0.127219878     c0
## 2           0.7181996       57298008    0.146067964     c1
## 3           0.9507910        7622892    0.231502122     c0
```

```
dim(dati_pls_stand)
```

```
## [1] 4454   11
```

Divisione di dati_pls_stand in training e test:

```
dim(dati_pls_stand_train)
```

```
## [1] 3119   11
```

```
dim(dati_pls_stand_test)
```

```
## [1] 1335   11
```

16) Logistico con variabili selezionate da PLS stand

```
Control = trainControl(method="cv",number=10,classProbs=TRUE,
                       summaryFunction=twoClassSummary)
glm_pls_stand <- train(target~.,data=dati_pls_stand_train,method="glm",
                  trControl=Control,metric="Sens",
                  preProcess=c("corr","nzv","BoxCox"),tuneLength=5)
```

```
glm_pls_stand
```

```
## Generalized Linear Model
##
## 3119 samples
##   10 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: Box-Cox transformation (9)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9804899  0.8029412  0.9698064
```

Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  410   77
##         c1   93 2539
##
##                Accuracy : 0.9455
##                  95% CI : (0.9369, 0.9532)
##                   Kappa : 0.7959
##
##             Sensitivity : 0.8151
##             Specificity : 0.9706
##          Pos Pred Value : 0.8419
##          Neg Pred Value : 0.9647
##              Prevalence : 0.1613
```

```
##
##        'Positive' Class : c0
```

Risultano esserci 93 asteroidi pericolosi classificati come non pericolosi.

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  159   35
##         c1   56 1085
##
##                Accuracy : 0.9318
##                  95% CI : (0.917, 0.9448)
##                   Kappa : 0.7374
##
##             Sensitivity : 0.7395
##             Specificity : 0.9688
##          Pos Pred Value : 0.8196
##          Neg Pred Value : 0.9509
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

Sul test set ci sono 56 asteroidi che potrebbero farci estinguere.

17) Rete neurale tunata sulle Componenti Principali

```
control = trainControl(method="cv", number=10, search="grid",
                       summaryFunction=twoClassSummary, classProbs=TRUE)
tunegrid <- expand.grid(size=c(1:6), decay=c(0.001, 0.01, 0.1))
nn_pca <- train(Hazardous~., data=train.df, method="nnet", metric="Sens",
                preProcess="pca", tuneGrid=tunegrid, trControl=control,
                trace=F, maxit=150)
nn_pca
```

```
## Neural Network
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: principal component signal extraction (16), centered
##  (16), scaled (16)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0.001  0.9412377  0.6642745  0.9441958
##   1     0.010  0.9409922  0.6642745  0.9430464
##   1     0.100  0.9411092  0.6741569  0.9411366
##   2     0.001  0.9882057  0.8707451  0.9744012
```

```
##   2     0.010  0.9907869  0.8927843  0.9751557
##   2     0.100  0.9862639  0.8687843  0.9755345
##   3     0.001  0.9875236  0.8787059  0.9740209
##   3     0.010  0.9921034  0.8807059  0.9774546
##   3     0.100  0.9910109  0.8807843  0.9797447
##   4     0.001  0.9912251  0.8789804  0.9778363
##   4     0.010  0.9931567  0.9047059  0.9801278
##   4     0.100  0.9925951  0.8985490  0.9770671
##   5     0.001  0.9910277  0.8946667  0.9755418
##   5     0.010  0.9916267  0.8828235  0.9774487
##   5     0.100  0.9925557  0.9045882  0.9782194
##   6     0.001  0.9886728  0.8866275  0.9743997
##   6     0.010  0.9915372  0.8907059  0.9774487
##   6     0.100  0.9922809  0.8846275  0.9805124
##
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were size = 4 and decay = 0.01.
```

Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  474   42
##         c1   29 2574
##
##                Accuracy : 0.9772
##                  95% CI : (0.9714, 0.9822)
##                   Kappa : 0.9167
##
##             Sensitivity : 0.9423
##             Specificity : 0.9839
##          Pos Pred Value : 0.9186
##          Neg Pred Value : 0.9889
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  194   20
##         c1   21 1100
##
##                Accuracy : 0.9693
##                  95% CI : (0.9586, 0.9779)
##                   Kappa : 0.8861
##
##             Sensitivity : 0.9023
##             Specificity : 0.9821
##          Pos Pred Value : 0.9065
##          Neg Pred Value : 0.9813
```

```
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

La Sensitivity non è particolarmente elevata, ma nemmeno eccessivamente bassa.

18) Naive Bayes tunato sulle Componenti Principali

```
control = trainControl(method="cv",number=10,classProbs=T,
                       summaryFunction=twoClassSummary)
nb_pca <- train(Hazardous~.,data=train.df,method="naive_bayes",
                preProcess=c("corr", "pca"),metric="Sens",
                trControl=control,tuneLength=5,na.action=na.pass)
nb_pca
```

```
## Naive Bayes
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## Pre-processing: principal component signal extraction (16), centered
##  (16), scaled (16)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   usekernel  ROC        Sens       Spec
##   FALSE      0.9330698  0.5667059  0.9621640
##    TRUE      0.9305701  0.5924706  0.9453467
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  341  116
##         c1  162 2500
##
##                Accuracy : 0.9109
##                  95% CI : (0.9003, 0.9206)
##                   Kappa : 0.6579
##
##             Sensitivity : 0.6779
##             Specificity : 0.9557
##          Pos Pred Value : 0.7462
##          Neg Pred Value : 0.9391
```

```
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  131   59
##         c1   84 1061
##                Accuracy : 0.8929
##                  95% CI : (0.875, 0.909)
##                   Kappa : 0.5841
##
##             Sensitivity : 0.60930
##             Specificity : 0.94732
##          Pos Pred Value : 0.68947
##          Neg Pred Value : 0.92664
##              Prevalence : 0.16105
##
##        'Positive' Class : c0
```

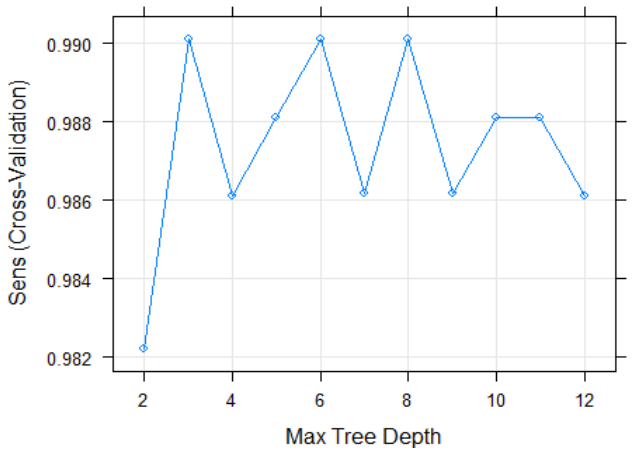
La Sensitivity sul test set non è molto soddisfacente.

19) Gradient Boosting

```
control <- trainControl(method="cv",number=10,summaryFunction=twoClassSummary,
                        classProbs=TRUE,search="grid")
grad_boost <- train(Hazardous~.,data=train.df,method="gbm",trControl=control,
                          metric="Sens",verbose=FALSE,
                          tuneGrid=data.frame(interaction.depth=c(2:12),
                                              n.trees=250,
                                              shrinkage=0.2,
                                              n.minobsinnode=50))
grad_boost
```

```
## Stochastic Gradient Boosting
##
## 3119 samples
##   16 predictor
##    2 classes: 'c0', 'c1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2807, 2808, 2807, 2806, 2807, 2808, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  ROC        Sens       Spec
##    2                 0.9976045  0.9821961  0.9980887
##    3                 0.9978513  0.9901176  0.9969407
##    4                 0.9983154  0.9861176  0.9973239
##    5                 0.9984802  0.9881176  0.9980887
##    6                 0.9983075  0.9901176  0.9977070
```

```
##    7                  0.9988164  0.9861569  0.9980872
##    8                  0.9975809  0.9901176  0.9977055
##    9                  0.9982023  0.9861569  0.9980887
##   10                  0.9984789  0.9881176  0.9977055
##   11                  0.9981085  0.9881176  0.9973224
##   12                  0.9982678  0.9861176  0.9980887
##
## Tuning parameter 'n.trees' was held constant at a value of 250
## Tuning
##  parameter 'shrinkage' was held constant at a value of 0.2
## Tuning
##  parameter 'n.minobsinnode' was held constant at a value of 50
## Sens was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 250, interaction.depth =
##  3, shrinkage = 0.2 and n.minobsinnode = 50.
```



Matrice di confusione sui dati di train:

```
##           Reference
## Prediction   c0   c1
##         c0  503    0
##         c1    0 2616
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##                   Kappa : 1
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1613
##
##        'Positive' Class : c0
```

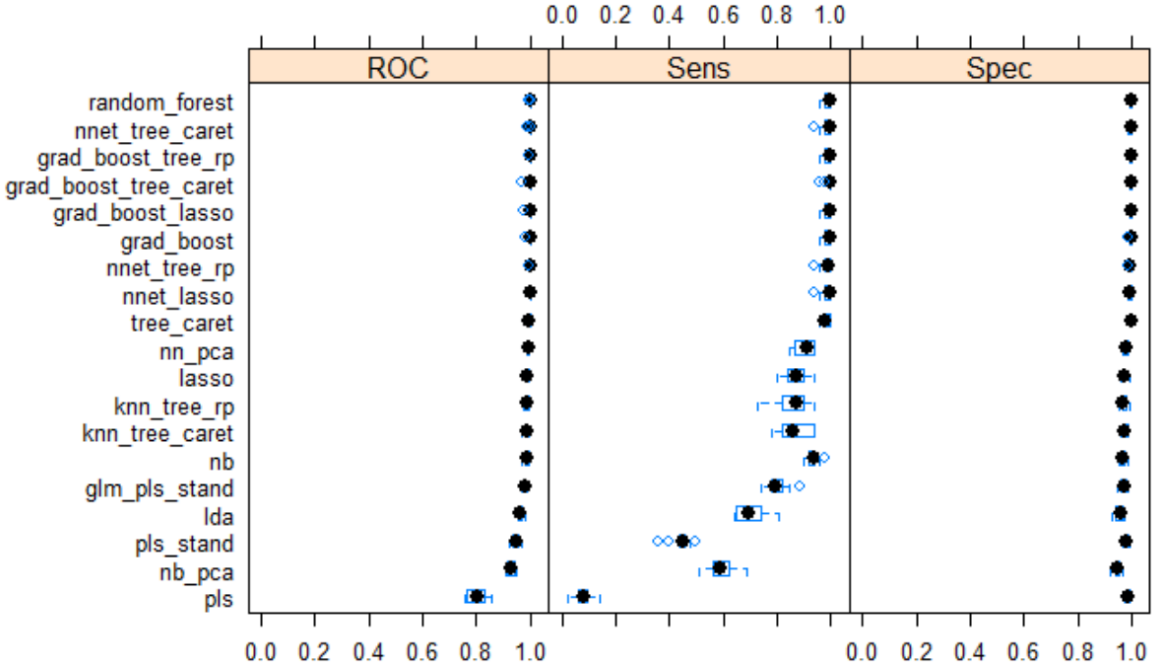
Matrice di confusione sui dati di test:

```
##           Reference
## Prediction   c0   c1
##         c0  212    2
##         c1    3 1118
##
##                Accuracy : 0.9963
##                  95% CI : (0.9913, 0.9988)
##                   Kappa : 0.9861
##
##             Sensitivity : 0.9860
##             Specificity : 0.9982
##          Pos Pred Value : 0.9907
##          Neg Pred Value : 0.9973
##              Prevalence : 0.1610
##
##        'Positive' Class : c0
```

STEP 2 - ASSESSMENT

Confronto tra le performance dei modelli

Per ogni modello si plottano le metriche di Sensitivity, Specificity e ROC:



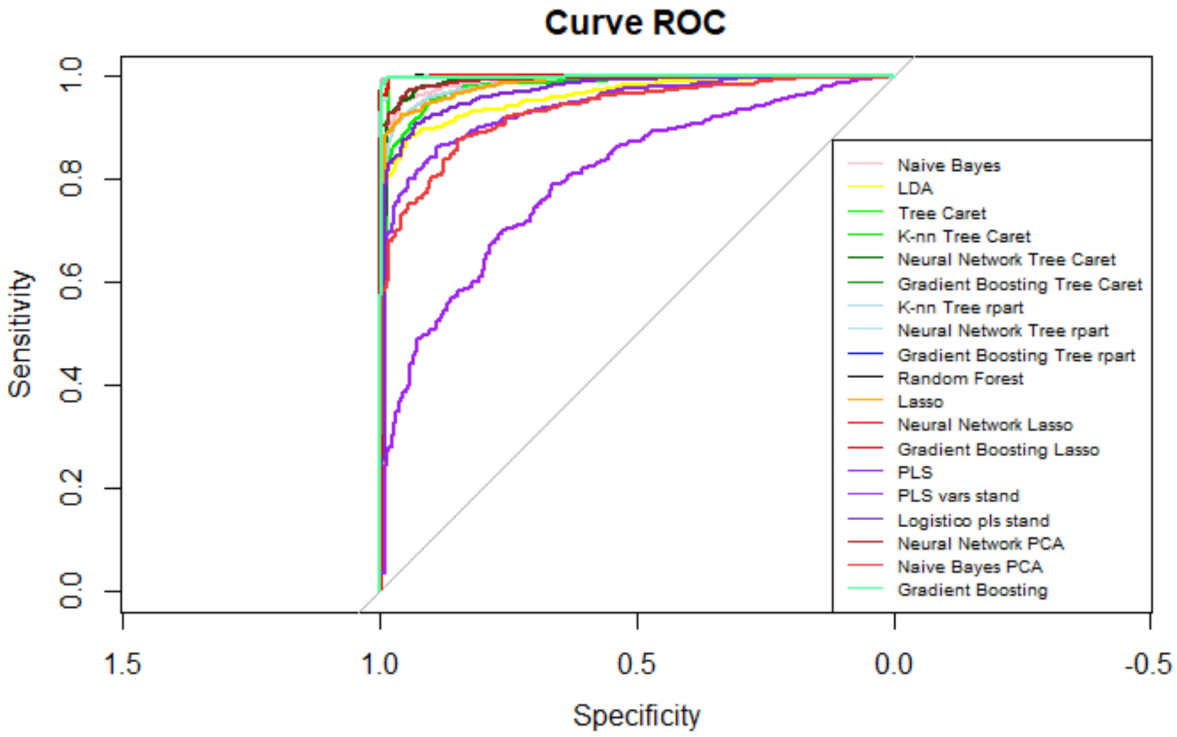
I modelli ensemble e le reti neurali sembrano avere performance migliori rispetto ai modelli base.

Curve ROC

Per ogni modello, nel dataset su cui è avvenuto il tuning, si salvano le posteriors per l'evento di interesse c0 - asteroide pericoloso - e poi si calcolano le aree sotto le curve ROC. N.B. si tratta di posteriors non ancora aggiustate con le true priors.

```
## Area under the curve naivebayes: 0.9873
## Area under the curve lda: 0.9632
## Area under the curve tree_caret: 0.9943
## Area under the curve knn_tree_caret: 0.9779
## Area under the curve nnet_tree_caret: 0.9908
## Area under the curve grad_boost_tree_caret: 0.9973
## Area under the curve knn_tree_rp: 0.9825
## Area under the curve nnet_tree_rp: 0.999
## Area under the curve grad_boost_tree_rp: 0.9974
## Area under the curve random_forest: 0.999
## Area under the curve lasso: 0.9793
## Area under the curve nnet_lasso: 0.9966
## Area under the curve grad_boost_lasso: 0.9996
## Area under the curve pls: 0.7993
## Area under the curve pls_stand: 0.9432
## Area under the curve glm_pls_stand: 0.9691
## Area under the curve nn_pca: 0.9929
## Area under the curve nb_pca: 0.931
## Area under the curve grad_boost: 0.9976
```

Plot delle curve ROC:



Dato che le curve ROC si intersecano, si plottano le curve lift, cercando il classificatore che cattura più "asteroidi pericolosi" nei primi decili, poiché le osservazioni sono ordinate per posteriors decrescente.

Correzione delle posteriors per tutti i modelli

```
prop.table(table(test.df$Hazardous))

##        c0        c1
## 0.1610487 0.8389513

true_c0 = 0.01
true_c1 = 0.99
old_c0 = 0.1610487
old_c1 = 0.8389513
```

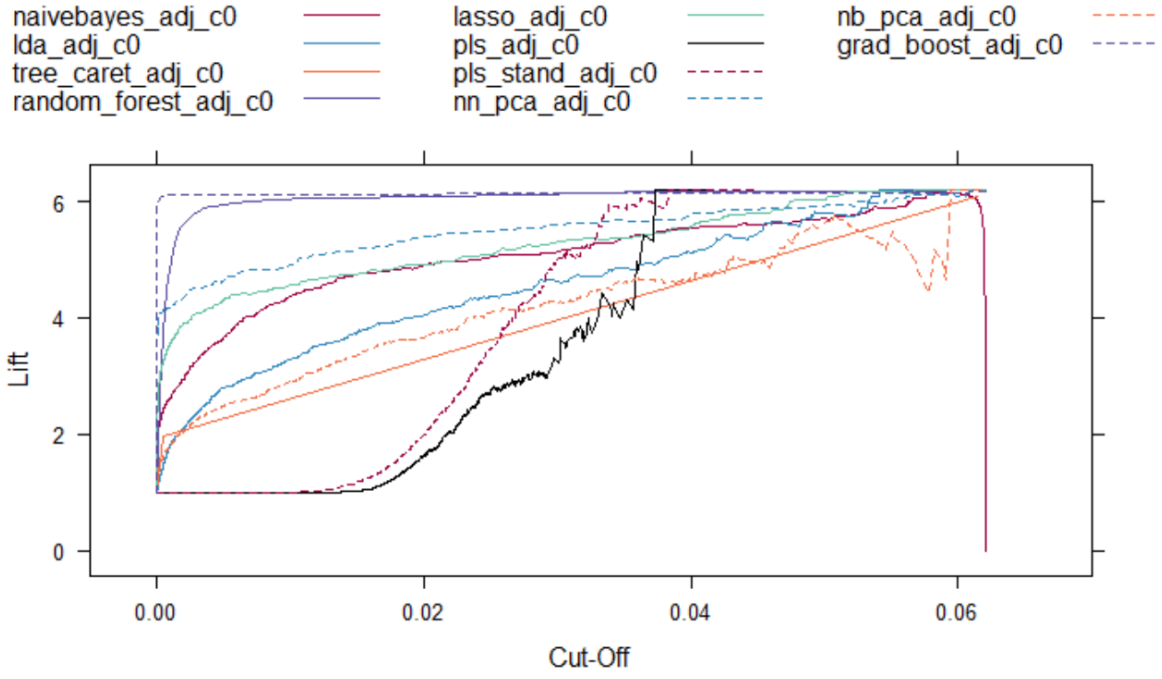
A titolo esemplificativo, si mostra come è stata effettuata la correzione delle posteriors sul primo modello, analogamente sui restanti.

```
den_nb=test.df$naivebayes_c0*(true_c0/old_c0)+test.df$naivebayes_c1*(true_c1/old_c
1)
test.df$naivebayes_adj_c0= test.df$naivebayes_c0*(true_c0/old_c0)/den_nb
test.df$naivebayes_adj_c1= 1-test.df$naivebayes_adj_c0
```

Curve LIFT

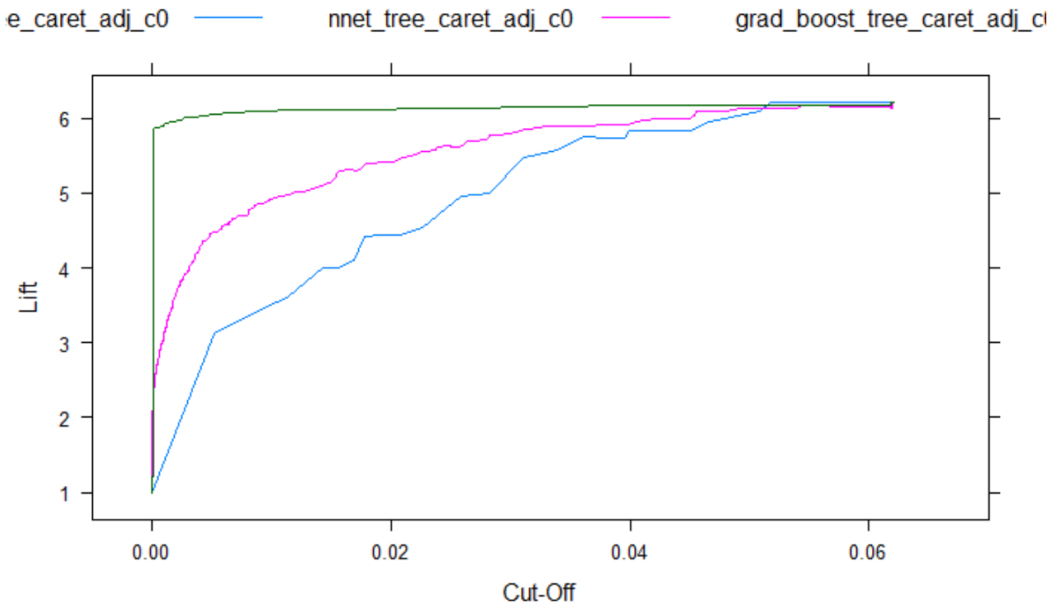
Le lift rappresentano il rapporto tra la probabilità prevista di event nel quantile di interesse e la probabilità prevista su tutto il dataset. Per il calcolo di queste curve si utilizzano le posteriors aggiustate.

Curve lift dei modelli tunati sui dati originali:



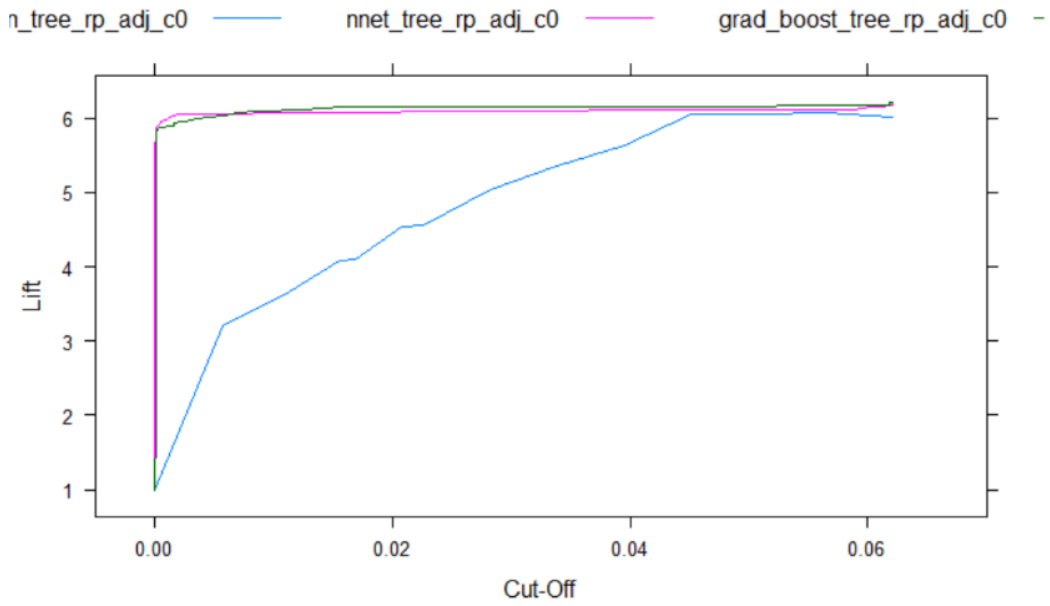
I modelli migliori sono il Random Forest e il Gradient Boosting.

Lift dei modelli tunati su variabili selezionate con un albero Caret:



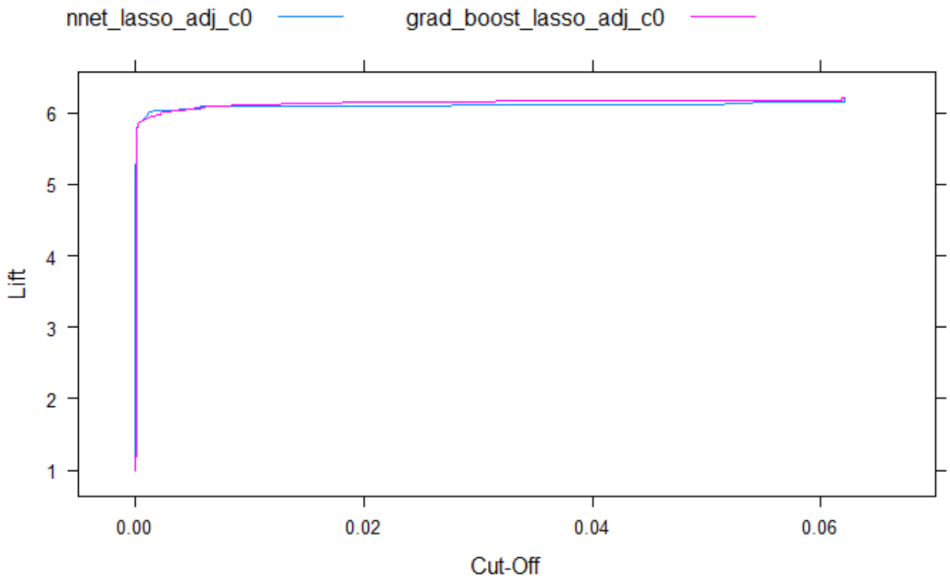
Il modello migliore è il Gradient Boosting.

Lift dei modelli tunati su variabili selezionate con un albero rpart:



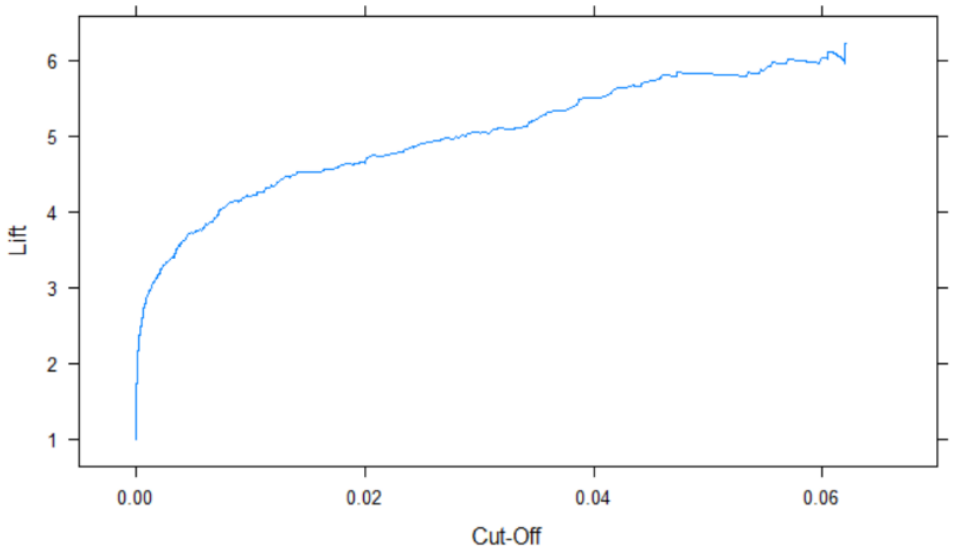
I modelli migliori sono la rete neurale e il Gradient Boosting.

Lift dei modelli tunati su variabili selezionate con un modello Lasso:



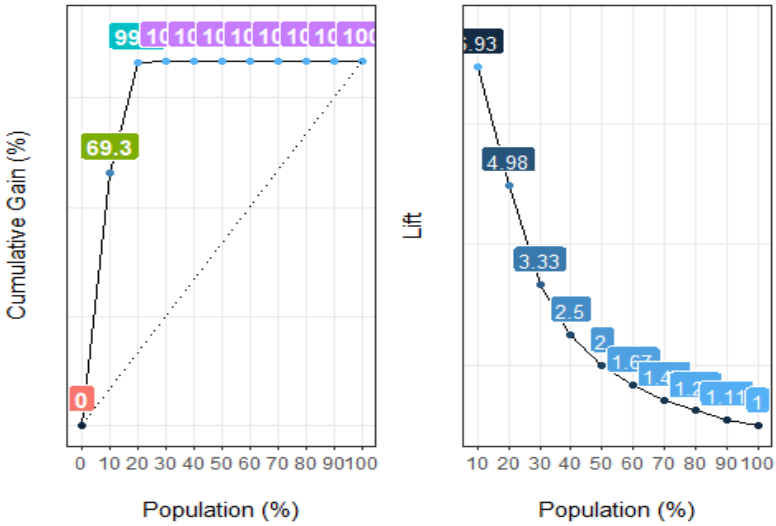
La rete neurale e Gradient Boosting hanno le curve lift quasi sovrapposte, quindi li consideriamo entrambi.

Lift del modello tunato su variabili selezionate con un modello PLS con variabili standardizzate:



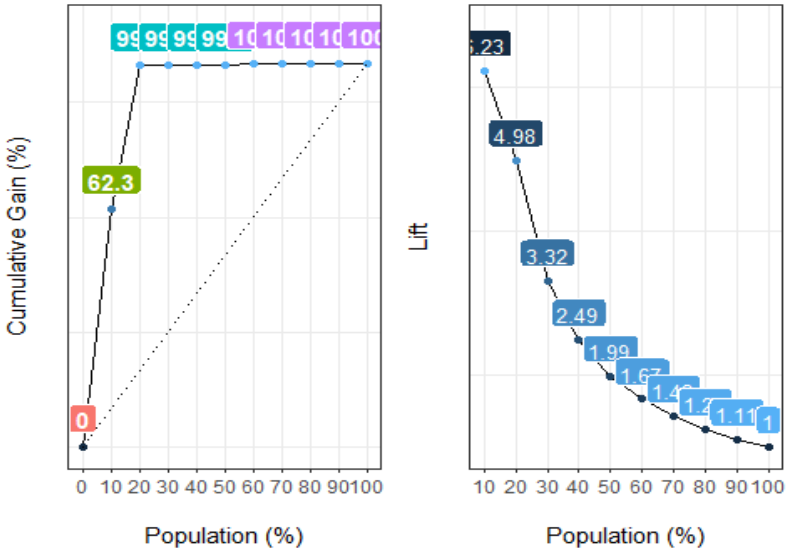
Si plottano solo le lift che sembrano vincere in ciascun grafico: il Random Forest e il Gradient Boosting per i dati originali; il Gradient Boosting per le variabili selezionate con l'albero tunato con Caret; la rete neurale e il Gradient Boosting per le variabili selezionate con l'albero tunato con rpart; la rete neurale e il Gradient Boosting per le variabili selezionate con il modello Lasso. Il logistico tunato sulle variabili selezionate con il PLS con le variabili standardizzate non presenta una buona curva lift, perciò non verrà considerato. Per ogni grafico si osserva il valore corrispondente al secondo decile.

Random Forest su dati originali:



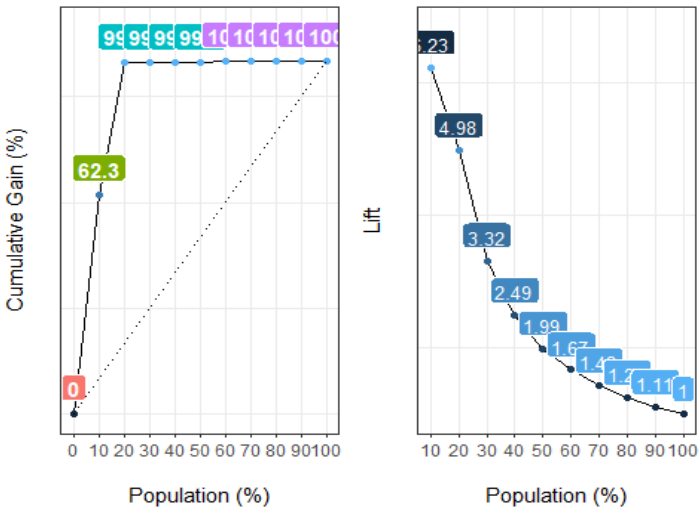

```
##    Population   Gain Lift Score.Point
## 1          10  69.30 6.93       0.988
## 2          20  99.53 4.98       0.016
## 3          30 100.00 3.33       0.004
## 4          40 100.00 2.50       0.000
## 5          50 100.00 2.00       0.000
```

Gradient Boosting su dati originali:



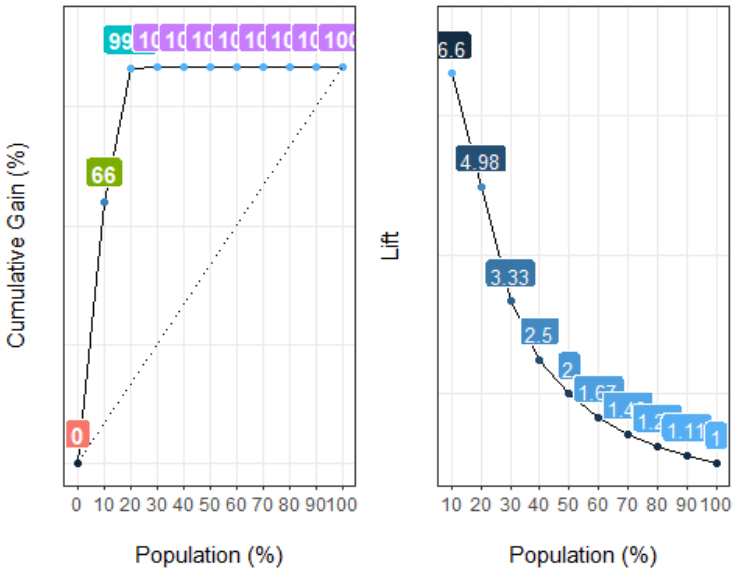

```
##    Population   Gain Lift         Score.Point
## 1          10  62.33 6.23 0.99999224917521923
## 2          20  99.53 4.98 0.00000764118580460
## 3          30  99.53 3.32 0.00000316987919478
## 4          40  99.53 2.49 0.00000177007895908
## 5          50  99.53 1.99 0.00000063982476931
```

Gradient Boosting su variabili selezionate con un albero tunato con Caret:



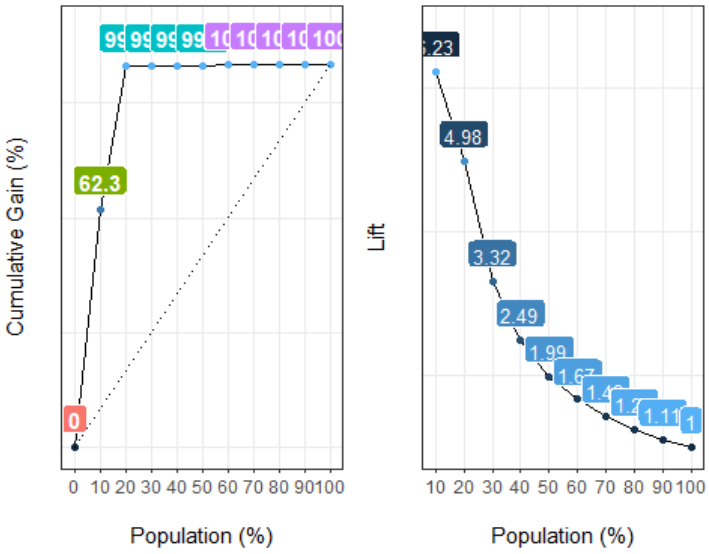

```
##    Population   Gain Lift  Score.Point
## 1          10  62.33 6.23 0.9969148406
## 2          20  99.53 4.98 0.0008948952
## 3          30  99.53 3.32 0.0006520642
## 4          40  99.53 2.49 0.0006201829
## 5          50  99.53 1.99 0.0005873985
```

Rete neurale su variabili selezionate con un albero tunato con rpart:



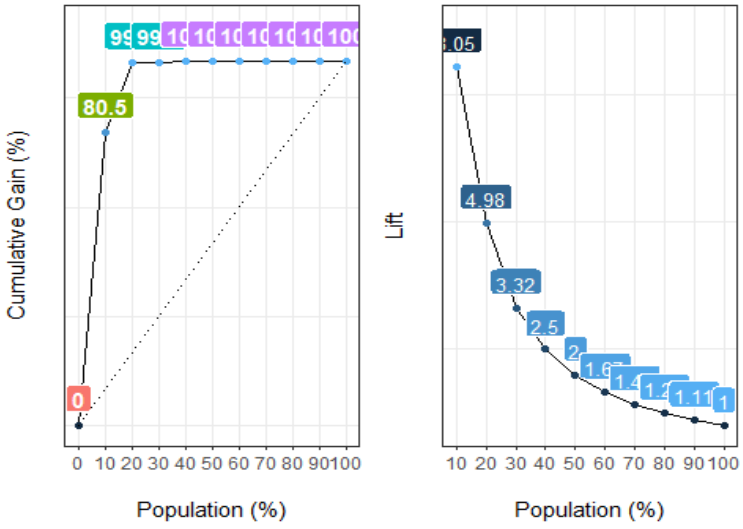

```
##    Population   Gain Lift    Score.Point
## 1          10  66.05 6.60 1.000000000000
## 2          20  99.53 4.98 0.000001172502
## 3          30 100.00 3.33 0.000000000000
## 4          40 100.00 2.50 0.000000000000
## 5          50 100.00 2.00 0.000000000000
```

Gradient boosting su variabili selezionate con un albero tunato con rpart:



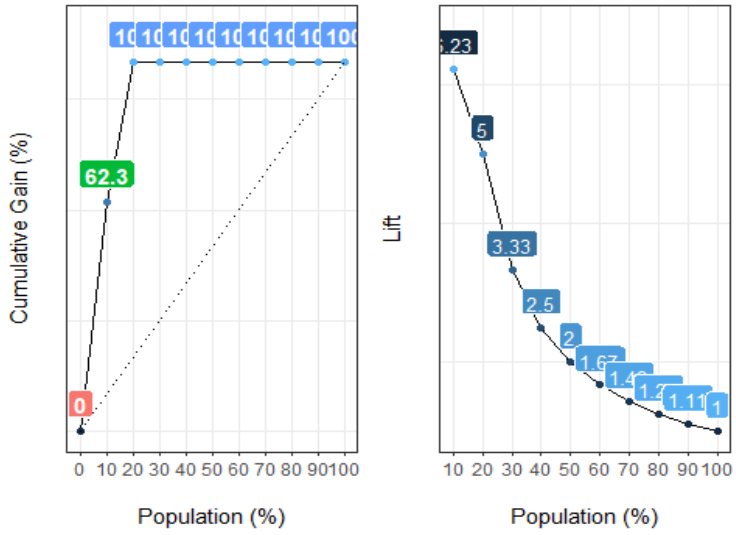

```
##    Population   Gain Lift   Score.Point
## 1          10  62.33 6.23 0.99720150984
## 2          20  99.53 4.98 0.00103569250
## 3          30  99.53 3.32 0.00060412185
## 4          40  99.53 2.49 0.00053588442
## 5          50  99.53 1.99 0.00051120150
```

Rete neurale su variabili selezionate con un modello Lasso:




```
##    Population   Gain Lift     Score.Point
## 1          10  80.47 8.05 1.0000000000000
## 2          20  99.53 4.98 0.0000006171677
## 3          30  99.53 3.32 0.0000003461203
## 4          40 100.00 2.50 0.0000000000000
## 5          50 100.00 2.00 0.0000000000000
```

Gradient Boosting su variabili selezionate con un modello Lasso:



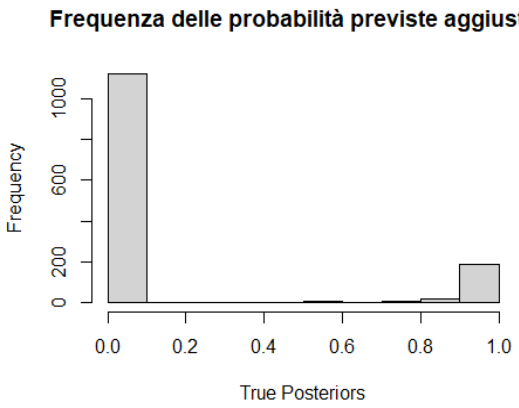

```
##    Population   Gain Lift  Score.Point
## 1          10  62.33 6.23 0.9969264644
## 2          20 100.00 5.00 0.0010010622
## 3          30 100.00 3.33 0.0007042332
## 4          40 100.00 2.50 0.0006322848
## 5          50 100.00 2.00 0.0005951090
```

Il modello migliore è il **Gradient Boosting** tunato sulle variabili selezionate con il modello Lasso. Perciò si porta avanti l'analisi con il modello grad_boost_lasso. Il primo 20% della popolazione, ordinata per posterior decrescente, cattura il 100% del totale degli asteroidi pericolosi.

STEP 3 – SCELTA DELLA SOGLIA

Si crea un dataset che contiene la variabile target e le posteriors corrette sia per l'event che per il non-event.

Distribuzione delle posteriors:



Il modello scelto sembra discriminare in modo ottimale le classi del target. Dal dataset creato, si selezionano solo il target osservato e la probabilità dell'evento di interesse, inoltre si rende il

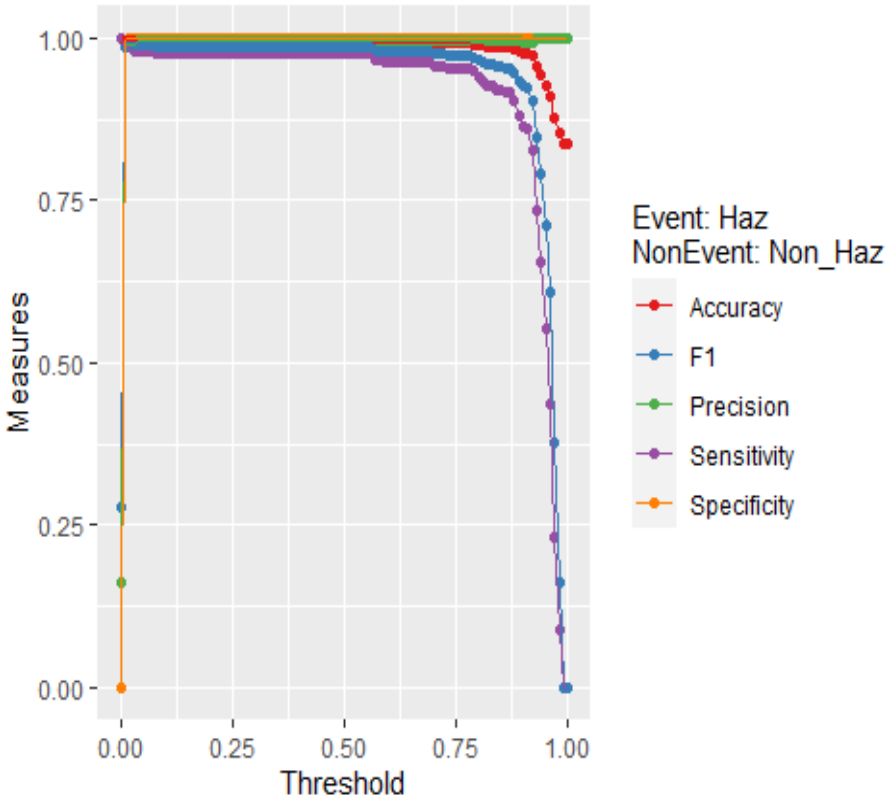
target una variabile character. Successivamente, si implementa un ciclo che, per ogni soglia da 0 a 1 con passo 0.01, calcola la Sensitivity, la Specificity, il numero di veri positivi, veri negativi e falsi negativi.

```
##    threshold Sensitivity Specificity true_Haz true_Non_Haz fn_Haz
## 1       0.00   1.0000000   0.0000000      215            0      0
## 2       0.01   0.9860465   0.9973214      212         1117      3
## 3       0.02   0.9860465   0.9982143      212         1118      3
## 4       0.03   0.9813953   0.9982143      211         1118      4
## 5       0.04   0.9813953   0.9991071      211         1119      4
```

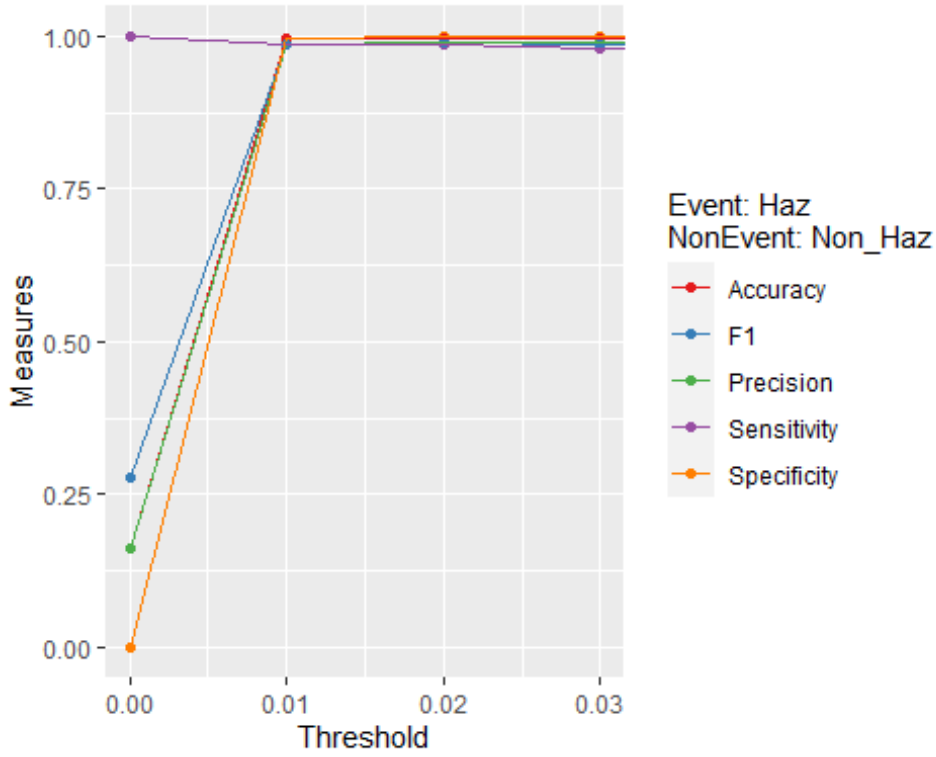
Si trova, per differenza, il numero di falsi positivi, l'Accuracy, la Precision e l'F1, ovvero la media armonica di Precision e Sensitivity.

Grafico finale

Si plottano le metriche per tutte le soglie a cui sono state calcolate:



Poiché il grafico non è chiaro per valori piccoli della soglia, si effettua uno zoom:



Si sceglie, osservando il grafico, una soglia di 0.02 e si calcolano il target previsto e la matrice di confusione per questa soglia.

```
##           Reference
## Prediction  Haz Non_Haz
##    Haz      212       2
##    Non_Haz    3    1118
##
##                Accuracy : 0.9963
##                  95% CI : (0.9913, 0.9988)
##                   Kappa : 0.9861
##
##             Sensitivity : 0.9860
##             Specificity : 0.9982
##          Pos Pred Value : 0.9907
##          Neg Pred Value : 0.9973
##              Prevalence : 0.1610
##
##        'Positive' Class : Haz
```

Il modello sembra avere ottime performance, anche se potrebbe non essere sufficiente per salvare l'umanità.

STEP 4 - SCORE DI NUOVI CASI

In questo ultimo step si considera il dataset score_data generato all'inizio dell'elaborato, con il 5% dei dati a disposizione. Si droppa la variabile target, poiché è quella che si vuole stimare e che normalmente non si conosce. Con la funzione *predict* si stimano le probabilità previste per l'evento e con la soglia scelta si ricava il target previsto.

```
##     Absolute.Magnitude Epoch.Date.Close.Approach Relative.Velocity.km.per.sec
## 12                25.8              790761600000                     17.274611
## 17                20.0              792835200000                      3.089004
## 64                20.6              801126000000                     15.455420
##     Orbit.Uncertainity Minimum.Orbit.Intersection Eccentricity Inclination
## 12                   4                 0.00186378    0.2048407   13.744233
## 17                   0                 0.24028800    0.3775494    5.660561
## 64                   7                 0.09478960    0.4908498   22.130166
##     Asc.Node.Longitude Orbital.Period Perihelion.Distance Perihelion.Arg
## 12           161.93352       504.8304           0.9866252       13.92021
## 17           112.16442      1000.8495           1.2188544       34.30692
## 64            72.21758       782.3624           0.8460297      100.31537
##     Perihelion.Time Mean.Anomaly Mean.Motion Miss.Dist..KM. Est.Dia.in.KM.
## 12          2457935     46.68496   0.7131108       64478851     0.01838887
## 17          2457772     82.35082   0.3596944       35727548     0.26580000
## 64          2457629    170.80325   0.4601448       16406943     0.20162992
##     prob_previste target_previsto
## 12   0.0001592154         Non_Haz
## 17   0.0006167147         Non_Haz
## 64   0.0006424633         Non_Haz
```

Poiché non è possibile costruire una matrice di confusione sui dati di score, non si può verificare se le osservazioni sono state classificate nel modo corretto. Tuttavia, avendo estratto il dataset di score con stratificazione rispetto alla variabile target, ci si aspetta di avere una proporzione di asteroidi pericolosi intorno al 16% del totale.

```
prop.table(table(score_data$target_previsto))
```

```
##       Haz   Non_Haz
## 0.1587983 0.8412017
```