Progetto Data Mining

Martina Chiesa 837484, Carlo Saccardi 839641, Davide Valoti 846737

Anna Accademico 2020-2021

Il dataset scelto contiene informazioni relative ad un e-commerce francese di successo, presente in diversi Paesi, basato sul modello economico C2C (customer to customer), in cui ogni utente è sia venditore, sia acquirente. Il dataset è composto da 98913 righe, ciascuna delle quali corrisponde a un utente registrato, e 24 colonne. Supponiamo che l'azienda sia interessata a ricevere un feedback da parte dai venditori sotto forma di questionario per svolgere delle analisi. Quindi mira a coinvolgere il maggior numero possibile di venditori, accettando di poter erroneamente inviare il questionario anche ad alcuni acquirenti, cercando di rendere minimo il numero di venditori esclusi dalla mailing list. Concretamente, cerchiamo un modello che riesca a massimizzare la sensitivity, e contemporaneamente a minimizzare i false negative (venditori classificati come acquirenti).

Creazione del target

Creiamo la variabile dummy *venditore*, che assume carattere 'c1' se l'utente ha venduto almeno un prodotto e 'c0' diversamente.

```
##    c0    c1
## 96877  2036
##         c0         c1
## 0.97941625 0.02058375
```

Solo il 2% degli utenti presenti nel dataset sono sia acquirenti sia venditori, invece, i restanti hanno solo comprato prodotti.

Cerchiamo ora di analizzare brevemente le variabili del dataset:

```
## 'data.frame':    98913 obs. of  25 variables:
##  $ identifierHash     : num  -1.10e+18 2.35e+18 6.87e+18 -4.64e+18 -5.18e+18 ...
##  $ type               : Factor w/ 1 level "user": 1 1 1 1 1 1 1 1 1 1 ...
##  $ country            : Factor w/ 200 levels "Â‰mirats arabes unis",..: 163 133 72 69 69 9 1
78 72 101 163 ...
##  $ language           : Factor w/ 5 levels "de","en","es",..: 2 2 4 2 2 1 2 4 5 2 ...
##  $ socialNbFollowers  : int  147 167 137 131 167 130 121 53 744 57 ...
##  $ socialNbFollows    : int  10 8 13 10 8 12 0 9 13764 8 ...
##  $ socialProductsLiked: int  77 2 60 14 0 1 1140 3 51671 45 ...
##  $ productsListed     : int  26 19 33 122 25 47 31 5 0 123 ...
##  $ productsSold       : int  174 170 163 152 125 123 108 106 104 92 ...
##  $ productsPassRate   : num  74 99 94 92 100 91 94 98 85 74 ...
##  $ productsWished     : int  104 0 10 7 0 0 531 0 1842 6 ...
##  $ productsBought     : int  1 0 3 0 0 0 105 0 0 2 ...
##  $ gender             : Factor w/ 2 levels "F","M": 2 1 1 1 1 1 1 1 1 1 ...
##  $ civilityGenderId   : int  1 2 2 2 2 2 3 2 2 3 ...
##  $ civilityTitle      : Factor w/ 3 levels "miss","mr","mrs": 2 3 3 3 3 3 1 3 3 1 ...
##  $ hasAnyApp          : Factor w/ 2 levels "False","True": 2 2 2 2 1 2 2 2 2 2 ...
##  $ hasAndroidApp      : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 2 1 1 1 ...
##  $ hasIosApp          : Factor w/ 2 levels "False","True": 2 2 2 2 1 2 1 2 2 2 ...
##  $ hasProfilePicture  : Factor w/ 2 levels "False","True": 2 2 1 1 2 1 1 2 1 2 ...
##  $ daysSinceLastLogin : int  11 12 11 12 22 11 11 11 14 11 ...
##  $ seniority          : int  3196 3204 3203 3198 2854 3196 3198 2857 3195 2856 ...
##  $ seniorityAsMonths  : num  106.5 106.8 106.8 106.6 95.1 ...
##  $ seniorityAsYears   : num  8.88 8.9 8.9 8.88 7.93 8.88 8.88 7.94 8.88 7.93 ...
```

```
##  $ countryCode        : Factor w/ 199 levels "ad","ae","af",..: 67 120 65 186 186 49 163 65
95 67 ...
##  $ venditore          : chr  "c1" "c1" "c1" "c1" ...
```

Per prima cosa procediamo con la rimozione della variabile identificativa *identifierHash*. In secondo luogo, eliminiamo *type*, poichè presenta un solo livello, quindi si tratta di una variabile non discriminante. Inoltre, abbiamo creato la nuova variabile *venditore*, di conseguenza non è più necessario conservare *productsSold*, che specifica il numero di prodotti venduti. Infine, trasformiamo *civilityGenderId* in factor.

Preprocessing

Valori mancanti

```
##             country            language   socialNbFollowers     socialNbFollows
##                   0                   0                   0                   0
## socialProductsLiked      productsListed    productsPassRate      productsWished
##                   0                4944                   0                3208
##      productsBought              gender    civilityGenderId       civilityTitle
##                   0                   0                   0                   0
##           hasAnyApp       hasAndroidApp           hasIosApp   hasProfilePicture
##                   0                   0                   0                   0
##  daysSinceLastLogin           seniority   seniorityAsMonths    seniorityAsYears
##                   0                   0                   0                   0
##         countryCode           venditore
##                   0                   0
```

productsListed e *productWished* presentano rispettavamente 4944 e 3208 valori mancanti. Quindi procediamo con la mice imputation con metodo pmm (predictive mean matching) con 5 ripetizioni.

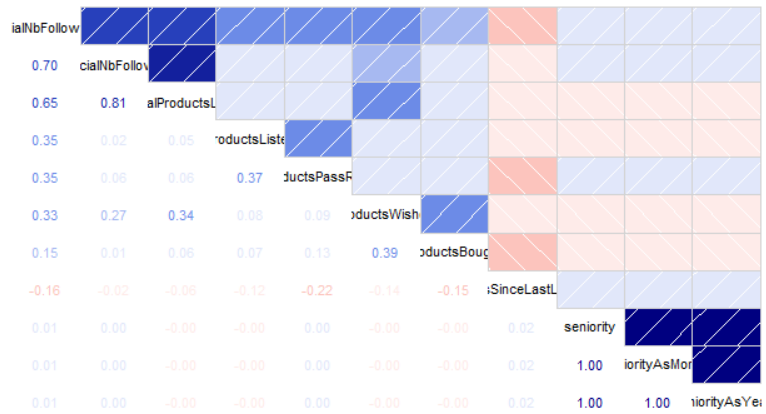
Collinearità

Valutiamo la presenza di collinearità tramite il pacchetto 'caret'.

```
##                     socialNbFollowers socialNbFollows socialProductsLiked
## socialNbFollowers         1.000000000     0.702766259        0.6535399873
## socialNbFollows           0.702766259     1.000000000        0.8094616307
## socialProductsLiked       0.653539987     0.809461631        1.0000000000
## productsListed            0.347529360     0.019276915        0.0462944883
## productsPassRate          0.351292886     0.055518508        0.0593945090
## productsWished            0.333847418     0.268396707        0.3355350756
## productsBought            0.147573023     0.006405666        0.0647406155
## daysSinceLastLogin       -0.159291999    -0.019811467       -0.0551656859
## seniority                 0.006033634     0.004150462       -0.0009995124
## seniorityAsMonths         0.006028219     0.004150835       -0.0009990087
## seniorityAsYears          0.006030095     0.004168001       -0.0009767599
##                     productsListed productsPassRate productsWished
## socialNbFollowers      0.347529360      0.351292886    0.333847418
## socialNbFollows        0.019276915      0.055518508    0.268396707
## socialProductsLiked    0.046294488      0.059394509    0.335535076
## productsListed         1.000000000      0.366978664    0.075739068
## productsPassRate       0.366978664      1.000000000    0.094569367
## productsWished         0.075739068      0.094569367    1.000000000
## productsBought         0.073157737      0.126696123    0.392855117
## daysSinceLastLogin    -0.115954137     -0.223065346   -0.136832462
## seniority             -0.003846331      0.001417019   -0.002884138
## seniorityAsMonths     -0.003845966      0.001414622   -0.002883190
## seniorityAsYears      -0.003873219      0.001434215   -0.002881212
##                     productsBought daysSinceLastLogin     seniority
```

```
## socialNbFollowers      0.147573023        -0.15929200  0.0060336337
## socialNbFollows        0.006405666        -0.01981147  0.0041504616
## socialProductsLiked    0.064740616        -0.05516569 -0.0009995124
## productsListed         0.073157737        -0.11595414 -0.0038463313
## productsPassRate       0.126696123        -0.22306535  0.0014170187
## productsWished         0.392855117        -0.13683246 -0.0028841379
## productsBought         1.000000000        -0.15342874 -0.0011375872
## daysSinceLastLogin    -0.153428742         1.00000000  0.0205184543
## seniority             -0.001137587         0.02051845  1.0000000000
## seniorityAsMonths     -0.001136058         0.02051829  0.9999998908
## seniorityAsYears      -0.001147286         0.02051288  0.9999806767
##                     seniorityAsMonths seniorityAsYears
## socialNbFollowers         0.0060282190     0.0060300950
## socialNbFollows           0.0041508353     0.0041680009
## socialProductsLiked      -0.0009990087    -0.0009767599
## productsListed           -0.0038459662    -0.0038732188
## productsPassRate          0.0014146219     0.0014342145
## productsWished           -0.0028831896    -0.0028812123
## productsBought           -0.0011360580    -0.0011472860
## daysSinceLastLogin        0.0205182911     0.0205128759
## seniority                 0.9999998908     0.9999806767
## seniorityAsMonths         1.0000000000     0.9999802972
## seniorityAsYears          0.9999802972     1.0000000000
```

I valori riportati in questo output si riferiscono alla correlazione semplice tra ogni coppia di variabili numeriche. Notiamo la presenza di alcuni valori pari a 1 che indicano perfetta correlazione positiva.



Decidiamo di procedere con la rimozione delle variabili la cui correlazione con altre è maggiore della soglia 0.95, cioè *seniorityAsMonths* e *seniorityAsYears*.

Per valutare l'associazione tra attributi qualitativi calcoliamo il chi quadrato normalizzato per ogni coppia di variabili. La presenza di un valore superiore a 0.95 indica forte associazione tra le due variabili e ne rimuoviamo una. Eliminiamo *civilityGenderId* e *gender* in quanto associate a *civilityTitle*, inoltre, escludiamo *country* associata a *countryCode*.

Zero-variance

Valutiamo ora zero variance e near zero variance.

```
##                       freqRatio percentUnique zeroVar   nzv
## socialNbFollowers     10.334469   0.090989051   FALSE FALSE
## socialNbFollows       39.770746   0.085934104   FALSE  TRUE
## socialProductsLiked   15.773997   0.424615571   FALSE FALSE
## productsListed       119.089461   0.064703325   FALSE  TRUE
## productsPassRate     222.174603   0.072791241   FALSE  TRUE
## productsWished        25.623209   0.276000121   FALSE  TRUE
## productsBought        28.357295   0.070769262   FALSE  TRUE
## daysSinceLastLogin     1.012131   0.706681629   FALSE FALSE
## seniority              1.039177   0.019208800   FALSE FALSE
## language               1.955256   0.005054947   FALSE FALSE
## civilityTitle          3.320639   0.003032968   FALSE FALSE
## hasAnyApp              2.779056   0.002021979   FALSE FALSE
## hasAndroidApp         19.525628   0.002021979   FALSE  TRUE
## hasIosApp              3.594834   0.002021979   FALSE FALSE
## hasProfilePicture     51.196834   0.002021979   FALSE  TRUE
## countryCode            1.220027   0.201186902   FALSE FALSE
## venditore             47.582024   0.002021979   FALSE  TRUE
```

Molte variabili presentano near zero variance, ma scegliamo di fissare a 100 la soglia minima di riferimento per rapporto di frequenza, quindi rimuoviamo *productsPassRate* e *productsListed* in quanto entrambe presentano valori elevati per tale rapporto.

Ricodifica livelli

countryCode possiede un numero molto elevato di livelli, ciascuno corrispondente a un Paese. Dopo averli ordinati per frequenza decidiamo di ricodificare la variabile (nominandola *optimal_country*) in modo tale da ricavare solo cinque livelli, di cui i primi quattro corrispondono ai Paesi più ricorrenti tra gli utenti del dataset, il quinto livello ('other') invece, racchiude i restanti.

```
##    fr    gb    it other    us
## 25135 11310  8015 33851 20602
```

Il dataset è ora composto da 14 variabili.

Under sampling

Come già osservato in precedenza, i venditori costituiscono solo 2% degli utenti presenti nel dataset iniziale, quindi è necessario bilanciarlo. Selezioniamo un campione di osservazioni in cui la percentuale di venditori raggiunge il 30% delle unità statistiche del campione stesso. In questo modo il nuovo dataset bilanciato presenterà tutti i 2036 eventi di interesse (venditori), ma solamente 4702 di 96877 non venditori verranno selezionati casualmente.

```
##   c0   c1
## 4702 2036
```

Divisione dataset

Il dataset bilanciato è in definitiva composto da 6738 unità statistiche, pertanto la numerosità è sufficiente per procedere con l'holdout method, ovvero la suddivisione in parti del dataset. Scegliamo di attribuire il 60% delle osservazioni al dataset di training, il 35% al test e il 5% è destinato al dataset di scoring.

Step 1

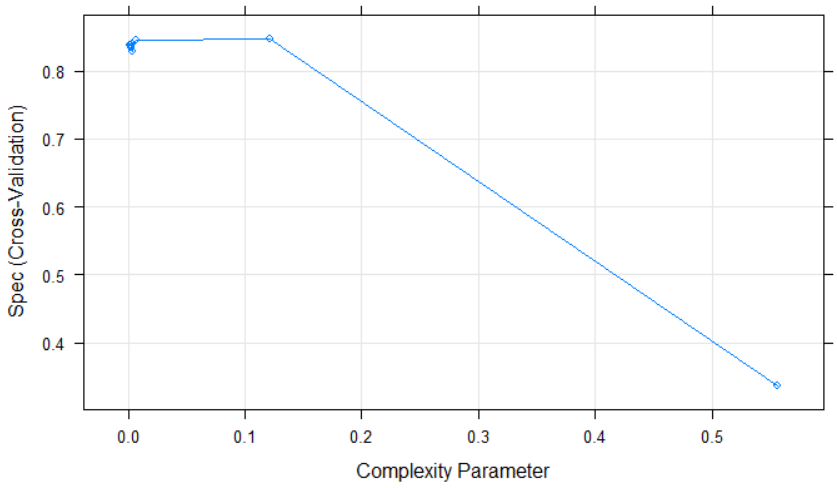
In questa prima fase, che segue le operazioni preliminari, creiamo diversi modelli servendoci del dataset di training e per ciascuno di questi stampiamo la matrice di confusione. La metrica con la quale abbiamo deciso di tunare i modelli è la sensitivity, in accordo con il nostro obiettivo di cercare di classificare al meglio possibile i veri venditori come tali. Tuttavia, nei commenti degli output ci concentreremo sulla specificity, in quanto la libreria caret interpreta come evento la classe c0, ovvero i non venditori. Tuniamo i modelli con crossvalidation a 10 fold, garantendo metriche non distorte e ottenendo informazione aggiuntiva per i modelli fittati, utile ad esempio per stimare la variabilità di tali metriche tramite intervalli di confidenza. I classificatori seguenti verranno fittati sul dataset preprocessato, le cui variabili non sono affette da collinearità, zero variance, near zero variance e missing data, quindi pratiche consigliate per rendere i modelli robusti a tali aspetti non verranno nuovamente applicate.

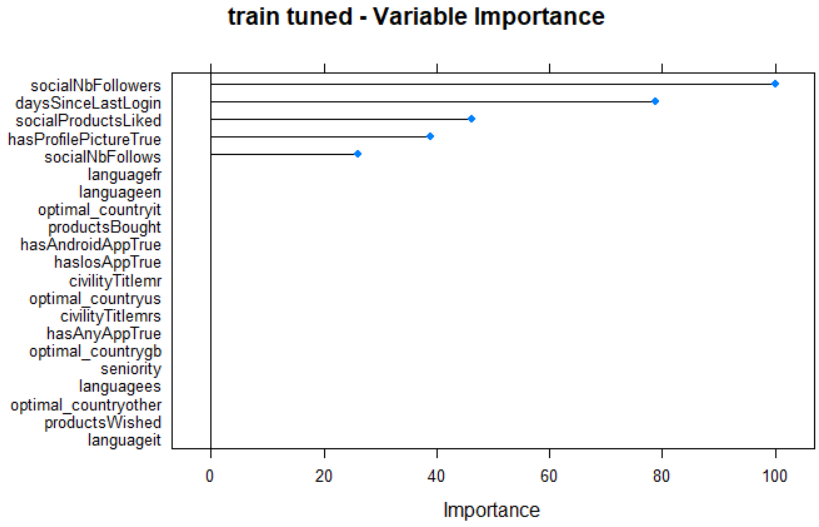
1. ALBERO DECISIONALE

Gli alberi decisionali permettono di analizzare la relazione di una variabile target rispetto alle covariate, di qualsiasi tipo siano. I vantaggi di questa tecnica sono la versatilità ad ogni tipologia di variabile e la non necessità di preprocessing. Questo algoritmo segue procedure di segmentazione e consente di partizionare i dati in gruppi, detti foglie, mutualmente esclusivi con l'obiettivo di renderli il più omogenei possibili. Un'osservazione appartenente ad una foglia verrà classificata in base alla classe più frequente del target.

```
## CART
##   cp           ROC        Sens       Spec
##   0.001022913  0.9445010  0.9277084  0.8372051
##   0.001091107  0.9445010  0.9277084  0.8372051
##   0.001636661  0.9384894  0.9316079  0.8388445
##   0.002182215  0.9381866  0.9316054  0.8380315
##   0.002454992  0.9378918  0.9330201  0.8347461
##   0.003273322  0.9371919  0.9358607  0.8282154
##   0.004091653  0.9307731  0.9319562  0.8396375
##   0.006546645  0.9118019  0.9287723  0.8453818
##   0.121931260  0.8733923  0.8943964  0.8462282
##   0.556464812  0.6389996  0.9401210  0.3378782
##
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.1219313.
```

Per potare l'albero viene utilizzata la misura di complessità R(alpha)=R+alpha*T dove R rappresenta il tasso di errata classificazione, alpha è il parametro di complessità e T è il numero di nodi terminali dell'albero. Alpha diventa quindi un parametro di penalizzazione per controllare la dimensione dell'albero. L'albero vincente ha un coefficiente di penalizzazione pari a 0.1219313 e assicura una sensitivity superiore a 0.84.





Notiamo come solamente 5 variabili vengono utilizzate come split per l'albero, in particolare la variabile *socialNbFollowers* risulta la più importante.

```
## yhat.tree         c0         c1
##        c0 0.60682493 0.04302671
##        c1 0.09099901 0.25914936
```

2. BAGGING

Il bagging appartiene alla famiglia dei metodi ensemble, ovvero algoritmi che cercano di superare i limiti dei semplici alberi decisionali. Nello specifico, il bagging cerca di risolvere il problema di elevata variabilità degli alberi aggregando i risultati di diversi alberi, ciascuno stimato su un diverso campione di osservazioni. Il modello predittivo utilizzato è lo stesso per ogni campione mentre i dati di training sono differenti. Questo comporta una perturbazione delle righe ma non delle colonne, infatti tutte le covariate vengono considerate per ogni albero, rendendoli tra loro correlati. Conseguentemente, la riduzione di variabilità non è significativa.

```
## Random Forest
##   ROC       Sens       Spec
##   0.962133  0.9316129  0.8486206
##
## Tuning parameter 'mtry' was held constant at a value of 15
```

L'algoritmo garantisce una performance sulla sensitivity crossvalidata superiore a 0.84, di poco migliore rispetto al semplice albero decisionale.

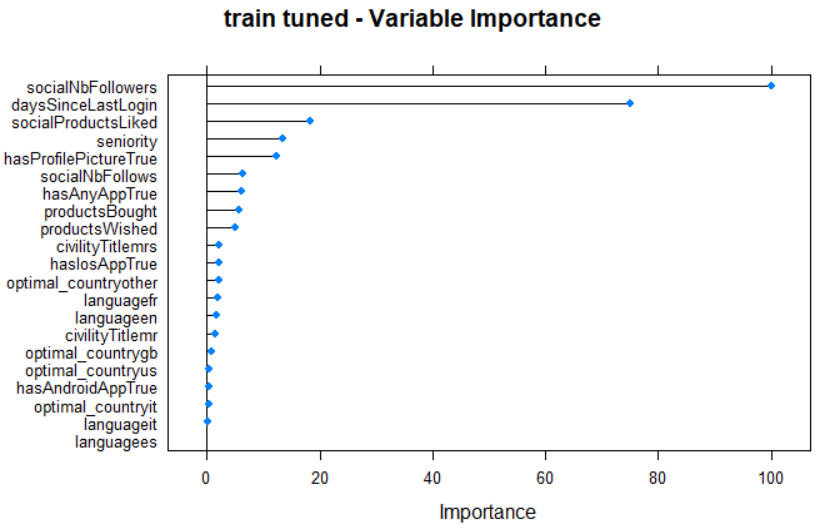
```
## yhat.bag           c0           c1
##       c0 0.6978239367 0.0007418398
##       c1 0.0000000000 0.3014342235
```

3. RANDOM FOREST

Procediamo al tuning di un altro metodo ensemble, il random forest. A differenza del bagging, questo algoritmo comporta una perturbazione sia delle righe sia delle colonne. Quest'ultima avviene selezionando un numero di covariate limitato in modo che ciascun albero stimato selezioni diverse variabili importanti. Questo modello richiede come parametro di tuning il numero di covariate utilizzate da ciascun albero. Tale parametro lo stimiamo intorno al valore della radice quadrata del numero di attributi del dataset. Questi risultano essere 15, pertanto scegliamo i valori compresi tra 2 e 7, per ciascuno dei quali stimiamo 250 alberi.

```
##           Reference
## Prediction   c0   c1
##         c0 65.2  4.6
##         c1  4.6 25.6
##
##  Accuracy (average) : 0.9083

## Random Forest
##   mtry  ROC        Sens       Spec
##   2     0.9599331  0.9475591  0.7929961
##   3     0.9632742  0.9415345  0.8159003
##   4     0.9643613  0.9383442  0.8338798
##   5     0.9639304  0.9376350  0.8371785
##   6     0.9635338  0.9348019  0.8429028
##   7     0.9630710  0.9340927  0.8486206
##
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.

## rf variable importance

##                         Overall
## socialNbFollowers     100.0000
## daysSinceLastLogin     75.0128
## socialProductsLiked    18.4002
## seniority              13.5216
## hasProfilePictureTrue  12.3337
## socialNbFollows         6.2981
## hasAnyAppTrue           6.0929
## productsBought          5.7119
## productsWished          5.0730
## civilityTitlemrs        2.2022
## hasIosAppTrue           2.1709
## optimal_countryother    2.0707
## languagefr              2.0456
## languageen              1.6417
## civilityTitlemr         1.4917
## optimal_countrygb       0.7688
## optimal_countryus       0.4969
## hasAndroidAppTrue       0.3064
```

```
## optimal_countryit       0.2899
## languageit              0.2371
```



Il numero di covariate che garantisce una migliore performance è 7, con una sensitivity associata superiore a 0.84. Le variabili importanti risultano essere molto più numerose rispetto a quelle prodotte dall'albero decisionale, come previsto. Decidiamo di mantenere come variabili significative solamente quelle che presentano un'importanza relativa superiore a 5, identificandone 9.

Essendo il random forest più robusto rispetto sia all'albero decisionale che rispetto al bagging, lo utilizzeremo anche come model selector e ci avvaleremo delle variabili importanti selezionate per i modelli successivi che richiederanno model selection come preprocessing.

4. GLM

Proponiamo un modello logistico a fini classificativi, non interpretativi, che richiede numerose procedure di preprocessing. Oltre a quelle già applicate, necessita model selection, quindi tuniamo il modello sul dataset appena creato con le variabili importanti selezionate dal random forest.

```
## Generalized Linear Model
##   ROC        Sens       Spec
##   0.9532584  0.9475453  0.7708717

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   c0   c1
##         c0 66.1  6.9
##         c1  3.7 23.3
##
##  Accuracy (average) : 0.8942
```

Il modello stimato ha una performance peggiore rispetto ai precedenti, con una sensitivity pari a 0.77. Procediamo alla stima di un nuovo modello logistico sul dataset di training non trattato.

```
## Generalized Linear Model
##   ROC        Sens       Spec
##   0.9586636  0.9482583  0.7904705
```

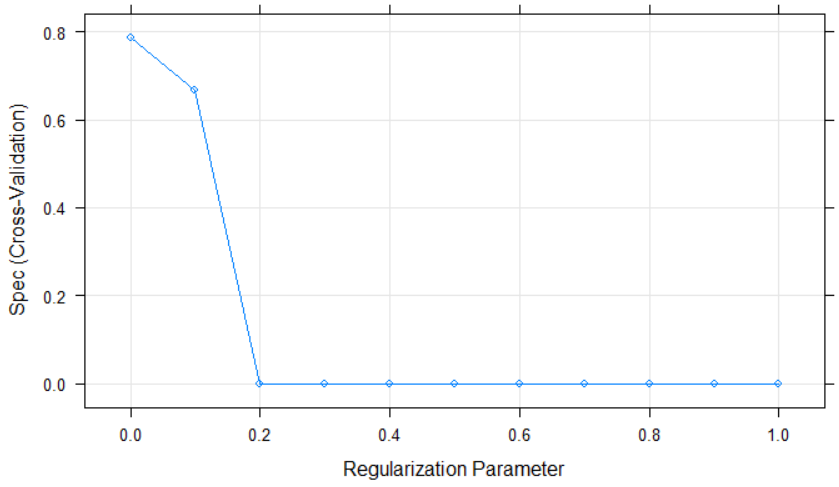
```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 66.2  6.3
##         c1  3.6 23.9
##
##  Accuracy (average) : 0.9006
```

La metrica di interesse risulta migliore rispetto al caso precedente, arrivando a 0.79.

5. LASSO

Vi sono molti modelli che in situazioni di alta complessità in termini di numero di variabili, necessitano nella fase di preprocessing di model selection. Infatti, questi, oltre ad essere difficili da interpretare, sono anche molto variabili e dunque poco replicabili su nuovi dati. Inoltre, la magnitudine dei coefficienti è sintomo di overfitting e instabilità. Il modello *lasso* supera questi problemi shrinkando i coefficienti a 0 mediante un parametro di tuning *lambda*.

```
## glmnet
##   lambda  ROC        Sens       Spec
##   0.0     0.9583527  0.9475491  0.7855591
##   0.1     0.9146615  0.9160076  0.6660869
##   0.2     0.9077808  1.0000000  0.0000000
##   0.3     0.9077808  1.0000000  0.0000000
##   0.4     0.5000000  1.0000000  0.0000000
##   0.5     0.5000000  1.0000000  0.0000000
##   0.6     0.5000000  1.0000000  0.0000000
##   0.7     0.5000000  1.0000000  0.0000000
##   0.8     0.5000000  1.0000000  0.0000000
##   0.9     0.5000000  1.0000000  0.0000000
##   1.0     0.5000000  1.0000000  0.0000000
##
## Tuning parameter 'alpha' was held constant at a value of 1
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.
```



```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 66.1  6.5
##         c1  3.7 23.7
##
##  Accuracy (average) : 0.8986
```

Osserviamo che il parametro lambda è pari a 0 e dunque il modello proposto coincide con un modello *OLS*.

6a. NAIVE BAYES

Il classificatore Naive Bayes è una semplificazione del classificatore Bayesiano. Questa versione semplificata approssima una densità multivariata in una produttoria di densità univariate sotto una forte assunzione, ovvero si assume che le densità delle variabili di input siano tra loro indipendenti. Il Naive Bayes soffre del cosiddetto *zero problem* se nella produttoria di densità univariate vi è anche un solo elemento pari a zero. In tal caso la probabilità a posteriori stimata dal modello sarà nulla.

```
## Naive Bayes
##   laplace  ROC        Sens       Spec
##   0        0.9161125  0.9741323  0.493456
##   1        0.9161125  0.9741323  0.493456
##
## Tuning parameter 'usekernel' was held constant at a value of FALSE
##
## Tuning parameter 'adjust' was held constant at a value of 0
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 0.

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 68.0 15.3
##         c1  1.8 14.9
##
##  Accuracy (average) : 0.8289
```

Il parametro di Laplace viene tunato pari a 0, quindi il nostro modello non soffre di zero problem. Dato che, la Sensitivity crossvalidata ottenuta mediante questo modello risulta essere molto scarsa, valutiamo se, applicando una model selection sul medesimo classificatore, migliora la metrica di interesse.

6b. NAIVE BAYES (con model selection)

```
## Naive Bayes
##   laplace  ROC        Sens       Spec
##   0        0.9376483  0.9755507  0.4705385
##   1        0.9376483  0.9755507  0.4705385
##
## Tuning parameter 'usekernel' was held constant at a value of FALSE
##
## Tuning parameter 'adjust' was held constant at a value of 0
```

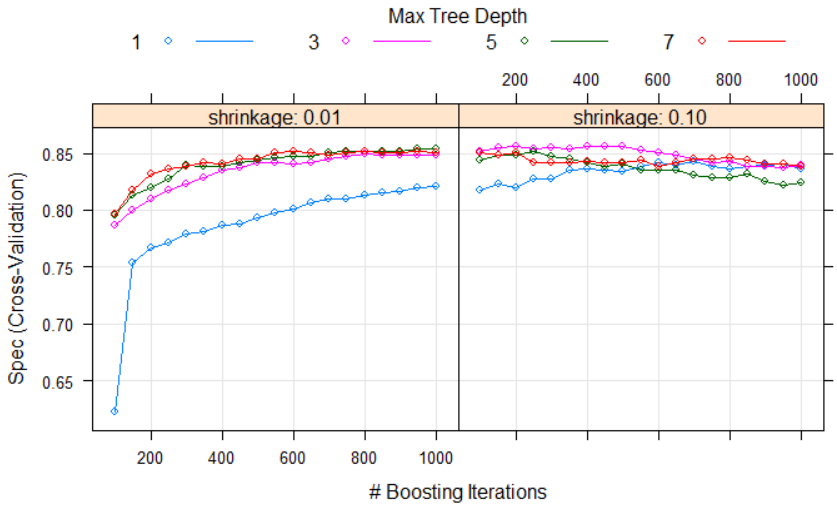
```
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 0.

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   c0   c1
##         c0 68.1 16.0
##         c1  1.7 14.2
##
##  Accuracy (average) : 0.8229
```

Questa soluzione risulta essere vincente, infatti la *sensitivity* crossvalidata aumenta da 0.55 a 0.70.

7. GRADIENT BOOSTING

Questo modello si serve di una regola classificativa che consiste in una somma pesata di altri modelli base indipendenti (solitamente alberi), dove i pesi sono dati dalla bontà classificativa dell'm-esimo modello base. Il gradient boosting è solitamente molto performante perchè se ci concentriamo sulle regole classificative dei vari alberi stimati notiamo che, all'm-esima iterazione, l'm-esimo modello base lavorerà sulla componente residua generata dal modello m-1esimo all'iterazione precedente. Congiuntamente, verrà stimato anche un peso alfa proporzionale alla bontà classificativa dell'm-esimo classificatore base. Dunque, non solo vengono considerate le previsioni delle iterazioni precedenti, ma queste vengono anche opportunamente pesate. Il parametro di tuning di questo modello è il numero di iterazioni, ovvero il numero di modelli base che verranno fittati sul dataset.

```
## Stochastic Gradient Boosting
##   shrinkage  interaction.depth  n.trees  ROC        Sens       Spec
##      0       7                 1000     0.9614574  0.9365687  0.8379915
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 200, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```



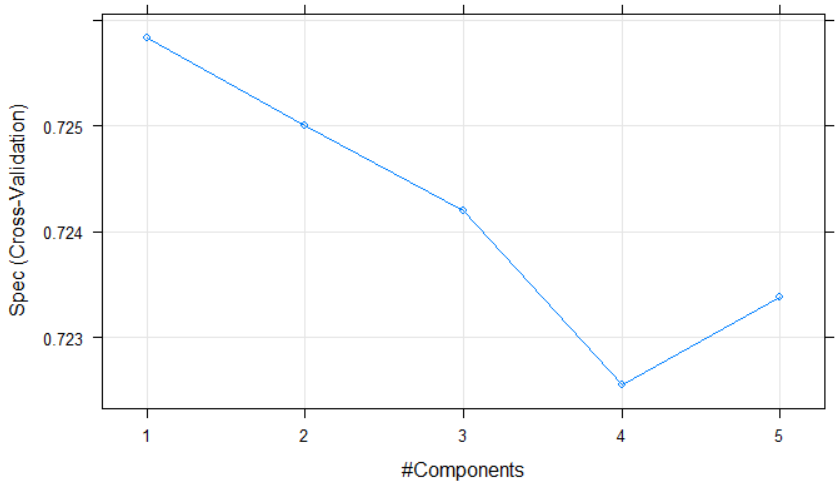
```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 65.8  4.4
##         c1  4.0 25.9
##
##  Accuracy (average) : 0.9167
```

Nel nostro caso, basandoci sulla sensitivity cross-validata, il modello che risulta essere più performate stima 200 alberi con shrinkage di 0.1, la profondità dell'albero è 3.

8. PARTIAL LEAST SQUARES REGRESSION

Questo modello trova nuove combinazioni PLS di covairate che massimizzano la loro variabilità con la risposta. Effettua riduzione della dimensionalità e regressione simultaneamente. Inoltre, ha la caratteristica di non richiedere preprocessing, quindi lo tuniamo sul dataset di training.

```
## Partial Least Squares
##   ncomp  ROC        Sens       Spec
##   1      0.9074029  0.8837556  0.7258230
##   2      0.9079967  0.8841077  0.7250033
##   3      0.9089335  0.8844636  0.7241903
##   4      0.9121696  0.8858795  0.7225576
##   5      0.9177425  0.8897765  0.7233773
##
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 1.
```



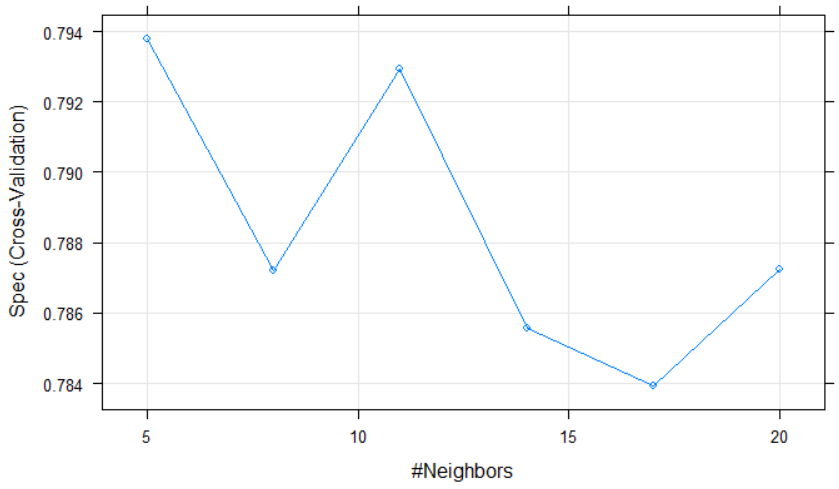

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 61.7  8.3
##         c1  8.1 21.9
##
##  Accuracy (average) : 0.8361
```

Il numero di componenti che preserviamo è 1, poichè la sensitivity associata è la più elevata (0.7258).

9a. k NEAREST NEIGHBOUR crossvalidato

Il KNN è un modello che lavora solo con variabili quantitative e individua, per ogni x, i k valori più vicini al test example x nel dataset di training con le distanze euclidee e analizza i corrispondenti valori del target. Se la maggior parte di questi yi è venditore, x sarà previsto come tale. Questo modello si basa su una stima locale e tende a soffrire di overfitting.

```
## k-Nearest Neighbors
##   k   ROC        Sens       Spec
##    5  0.9359162  0.9326680  0.7937692
##    8  0.9452532  0.9379834  0.7872184
##   11  0.9471057  0.9418816  0.7929495
##   14  0.9500937  0.9425920  0.7855858
##   17  0.9508508  0.9440055  0.7839531
##   20  0.9518892  0.9471932  0.7872318
##
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```



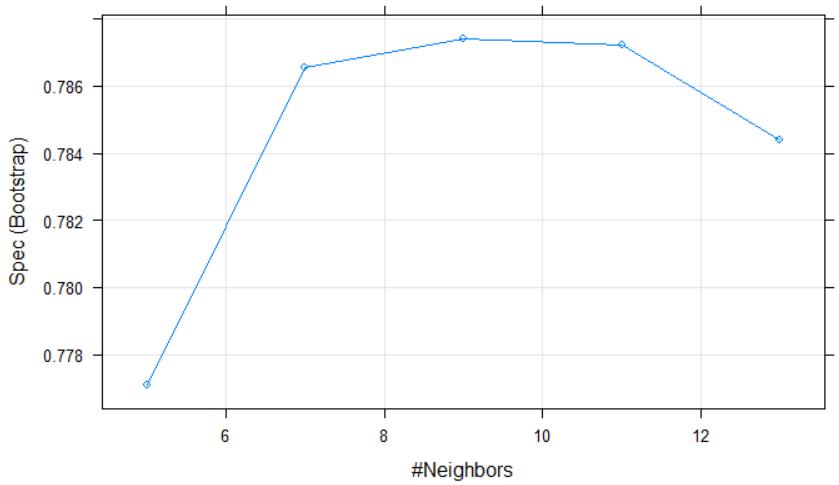

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   c0   c1
##         c0 65.1  6.2
##         c1  4.7 24.0
##
##  Accuracy (average) : 0.8907
```

Con k pari a 5 la sensitivity è massimizzata e pari a 0.7937, quindi un valore basso. Di conseguenza proviamo a utilizzare altri metodi quali Bootstrap e principal component dato che il modello KNN è instabile.

9b. KNN con Bootstrap

Tuniamo lo stesso modello con medoto Bootstrap ora:

```
## k-Nearest Neighbors
##   k   ROC        Sens       Spec
##    5  0.9165052  0.9220193  0.7771094
##    7  0.9269710  0.9270593  0.7865647
##    9  0.9335339  0.9295300  0.7873938
##   11  0.9370015  0.9316565  0.7872110
##   13  0.9394124  0.9340463  0.7844018
##
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```




```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 64.8  6.5
##         c1  4.9 23.8
##
##  Accuracy (average) : 0.8863
```

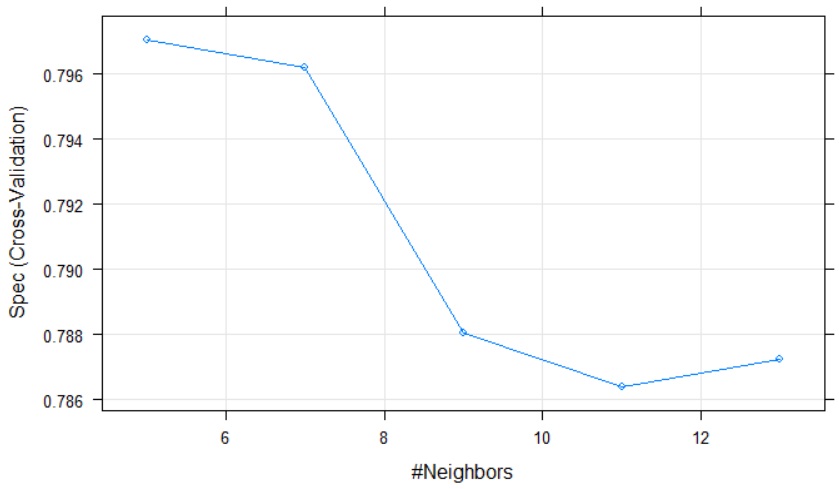
In questo caso il numero k di vicini scelto è 9, con una sensitivity di 0.7873.

9c. KNN con principal components

Procediamo ora con il metodo delle componenti principali, scelto poichè il KNN richiede come preprocessing lo scaling delle covariate e le PC risultano già standardizzate per costruzione.

```
## k-Nearest Neighbors
##   k   ROC        Sens       Spec
##    5  0.9351798  0.9319625  0.7970345
##    7  0.9425422  0.9337343  0.7962082
##    9  0.9456869  0.9351490  0.7880381
##   11  0.9476802  0.9411711  0.7864054
##   13  0.9495562  0.9422349  0.7872184
##
```

```
## Spec was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```




```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   c0   c1
##         c0 65.0  6.1
##         c1  4.7 24.1
##
##  Accuracy (average) : 0.8912
```

In questo caso k è pari a 5 e la relativa sensitivity è 0.7970.

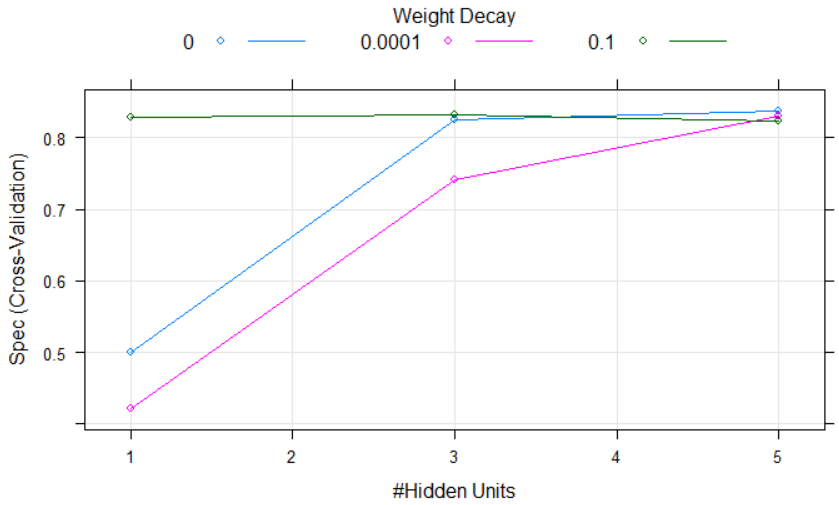
10 NEURAL NETS

Generiamo un nuovo dataset con le stesse variabili aventi target numerico e variabili dummy numeriche. Su questo nuovo dataset costruiamo alcune reti neurali, ovvero algoritmi che cercano di riprodurre le nozioni delle neuroscienze riguardo al funzionamento dei neuroni. Ogni rete infatti è composta da strati di neuroni nascosti in cui vengono aggregate le variabili. Ciascuna delle quali viene sottoposta a standardizzazione e ha un peso w associato. Le covariate vengono poi aggregate nei neuroni nascosti tramite combinazioni lineari pesate. Infine, nello strato di output viene impiegata una funzione di attivazione non lineare per stimare la classe del target utilizzando come input gli output dei neuroni nascosti dello strato precedente.

Stimiamo una prima rete utilizzando come griglia di parametri quella di default di R presente nel pacchetto caret. Centriamo le variabili sottraendo la media di ciascuna colonna. Non avendo specificato la funzione di attivazione, viene utilizzata da caret di default quella logistica.

```
## Neural Network
##   size  decay  ROC        Sens       Spec
##   1     0e+00  0.7747079  0.9688257  0.5000000
##   1     1e-04  0.7288925  0.9730672  0.4213115
##   1     1e-01  0.9564416  0.9408140  0.8281821
##   3     0e+00  0.9576005  0.9429429  0.8257364
##   3     1e-04  0.9071574  0.9511014  0.7404971
##   3     1e-01  0.9588917  0.9422362  0.8323071
##   5     0e+00  0.9593833  0.9432962  0.8380048
```

```
##   5     1e-04  0.9596738  0.9418816  0.8298481
##   5     1e-01  0.9603343  0.9457798  0.8240970
##
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were size = 5 and decay = 0.
```




```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 65.8  4.9
##         c1  4.0 25.3
##
##  Accuracy (average) : 0.9115
```

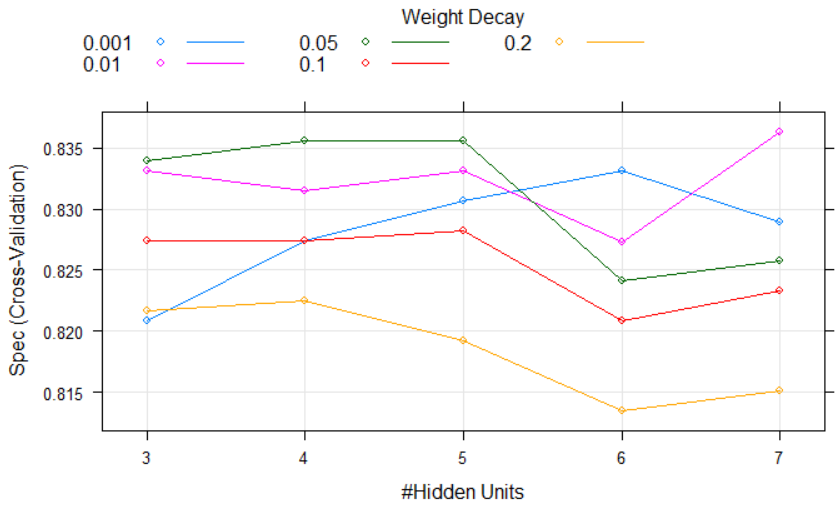
Il modello così ottenuto presenta 5 neuroni nello strato nascosto e un coefficiente di decay pari a 0. Questo ultimo è un parametro di regolarizzazione e viene utilizzato per evitare il fenomeno di overfitting.

```
##    TrainROC TrainSens TrainSpec method
## 1 0.9593833 0.9432962 0.8380048   nnet

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 65.8  4.9
##         c1  4.0 25.3
##
##  Accuracy (average) : 0.9115
```

La sesnsitivity del modello stimato è pari a 0.83.

Stimiamo ora una nuova rete avente una griglia personalizzata con valori simili a quelli osservati nella rete vincente precedente. Questo permette di verificare se i valori di massimo osservati per la rete con la griglia di default sono effettivamente ottimali o possono essere migliorati.

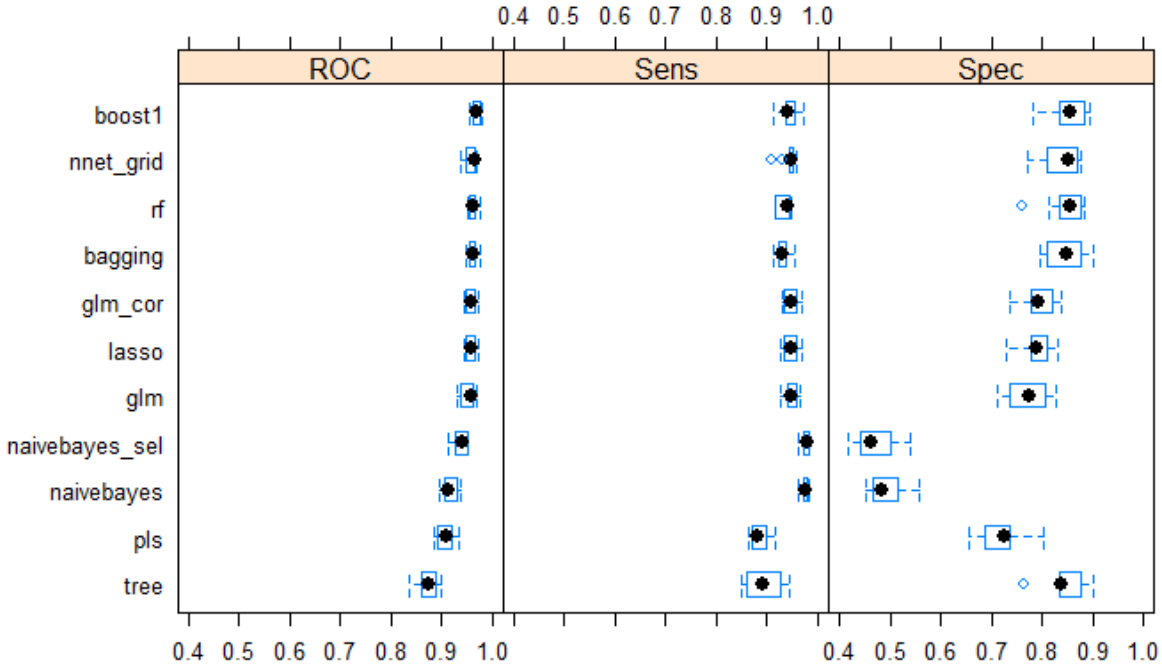
```
## Neural Network

##   size  decay  ROC  Sens       Spec
##   3     0.001  NaN  0.9394031  0.8208317
##   3     0.010  NaN  0.9425908  0.8331201
##   3     0.050  NaN  0.9443651  0.8339464
##   3     0.100  NaN  0.9411711  0.8273557
##   3     0.200  NaN  0.9436483  0.8216513
##   4     0.001  NaN  0.9429466  0.8274024
##   4     0.010  NaN  0.9429454  0.8314674
##   4     0.050  NaN  0.9425908  0.8355858
##   4     0.100  NaN  0.9457798  0.8273757
##   4     0.200  NaN  0.9489662  0.8224710
##   5     0.001  NaN  0.9422374  0.8306677
##   5     0.010  NaN  0.9440080  0.8331267
##   5     0.050  NaN  0.9461356  0.8355858
##   5     0.100  NaN  0.9457785  0.8282087
##   5     0.200  NaN  0.9464865  0.8191857
##   6     0.001  NaN  0.9418816  0.8331134
##   6     0.010  NaN  0.9429416  0.8273491
##   6     0.050  NaN  0.9432987  0.8240970
##   6     0.100  NaN  0.9454226  0.8208383
##   6     0.200  NaN  0.9475478  0.8134546
##   7     0.001  NaN  0.9376325  0.8289884
##   7     0.010  NaN  0.9443651  0.8363655
##   7     0.050  NaN  0.9440092  0.8257564
##   7     0.100  NaN  0.9457810  0.8232907
##   7     0.200  NaN  0.9461331  0.8150873
##
## Spec was used to select the optimal model using the largest value.
## The final values used for the model were size = 7 and decay = 0.01.
```




```
##   TrainROC TrainSens TrainSpec method
## 1      NaN 0.9443651 0.8363655   nnet

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   c0   c1
##         c0 65.9  4.9
```

```
##         c1  3.9 25.3
##
##  Accuracy (average) : 0.9117
```

Il nuovo modello presenta 7 neuroni nello strato nascosto e un coefficiente di decay pari a 0.1. La sensitivity osservata risulta pari a 0.83, valore molto prossimo a quello osservato nel modello precedente.



Da questo grafico possiamo comprendere la stabilità dei vari modelli. Le metriche di specificity e roc risultano essere poco variabili nelle 10 fold sulle quali sono state calcolate. D'altra parte, la variabilità della metrica di sensitivity è maggiore, poichè i venditori occupano una quota inferiore di utenti rispetto agli acquirenti. Notiamo inoltre che le sensitivity crossvalidate ottenute tramite Gradient Boosting e Radom Forest risultano essere più elevate rispetto a quelle dei restanti classificatori.

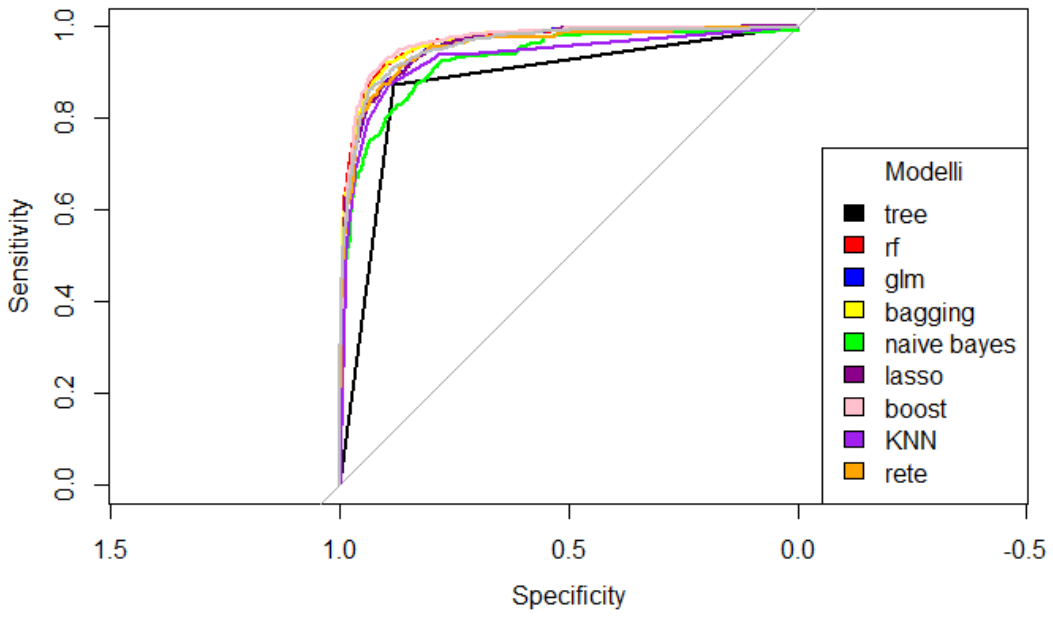
Step 2

Dopo aver costruito tutti i modelli sul training vogliamo valutarne la performance per confrontarli. Scegliamo di utilizzare curve roc, in quanto il nostro target è binario.

CURVE ROC

La curva roc mostra come varia la probabilità di corretta classificazione dei venditori (sensitivity) al variare dell'errata classificazione dei non venditori (1-specificity) per ciascuna soglia (ciascun punto della roc). La crescita della curva mostra la rapidità della corretta previsione dei venditori, minimizzando gli errori sui non venditori. L'algoritmo prevede che per calcolare le roc si debbano ordinare le osservazioni del test dataset rispetto alla probabilità prevista di essere venditore, poi, per ogni osservazione si calcolano sensitivity e 1-specificity. La rapidità della crescita della curva misura la bontà classificativa e si può approssimare con l'area sottostante la curva roc (AUC), tale valore varia da 0.5 a 1 (0.5 indica classificazione legata al caso, 1 perfetta classificazione in ogni unità statistica).

```
roc.tree
## Area under the curve: 0.8765
roc.rf
## Area under the curve: 0.9642
roc.glm_corr
## Area under the curve: 0.9576
roc.bagging
## Area under the curve: 0.9641
roc.naivebayes_sel
## Area under the curve: 0.9259
roc.lasso
## Area under the curve: 0.9573
roc.boost1
## Area under the curve: 0.9671
roc.knn_pc
## Area under the curve: 0.9309
roc.nnet_grid
## Area under the curve: 0.9529
roc.nnet_near
## Area under the curve: 0.9577
```



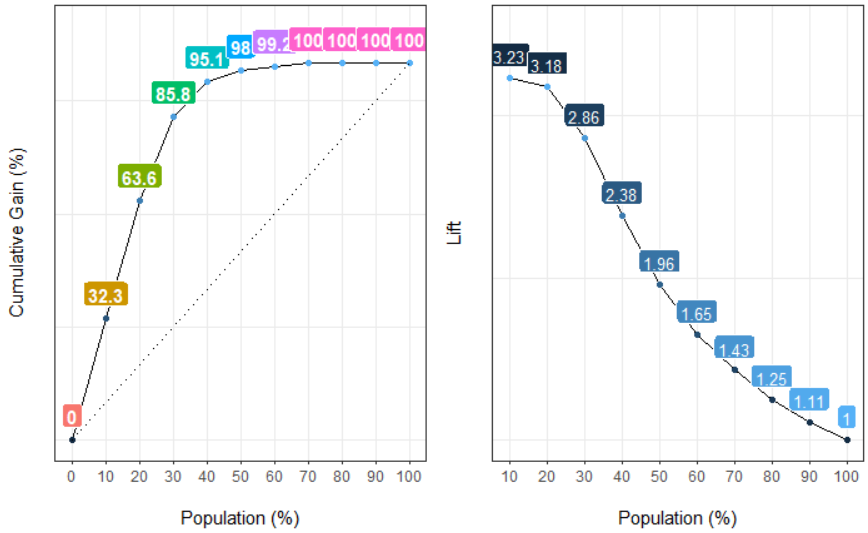
Questa figura riporta sull'asse delle ascisse la classificazione dell'evento essere venditore, mentre sulle ordinate la classificazione dei non venditori. La maggior parte delle curve cresce rapidamente e presenta un'area sottostante ampia, quindi deduciamo che sono buoni classificatori al variare delle soglie. Tuttavia, non è chiaro dal grafico quale modello performi in modo migliore dato che le curve si intersecano e nessuna di queste si posiziona superiormente rispetto alle altre. Quindi si ricorre alle curve lift, ottenute sui modelli con metriche di AUC maggiori, che generano indecisione. Nel nostro caso si tratta di: random forest, boosting e bagging.

CURVE LIFT

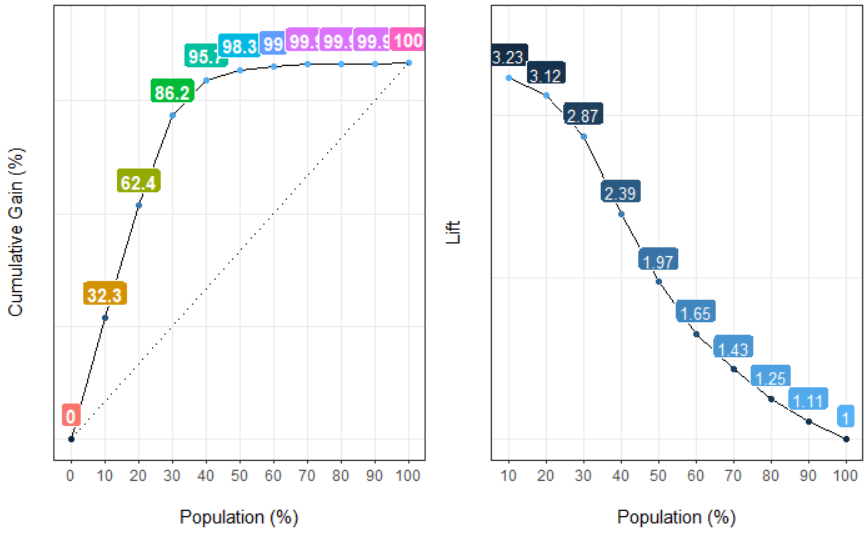
Le curve lift sono una metodologia di assessment adatta solo per target qualitativi. Coi modelli fittati si stimano sul validation le probabilità di successo per ogni i (pi), secondo le quali si ordinano in senso decrescente i soggetti. A questo punto le n unità statistiche vengono divise in decili, con il primo che indica maggior probabilità di essere venditore, infatti nel primo decile ci sono il 10% dei soggetti con maggiore probabilità. Il modello è buono se le probabilità tra un decile e il successivo sono particolarmente differenti, quindi c'è discrimination.

```
##        c0        c1
## 0.6979211 0.3020789
```

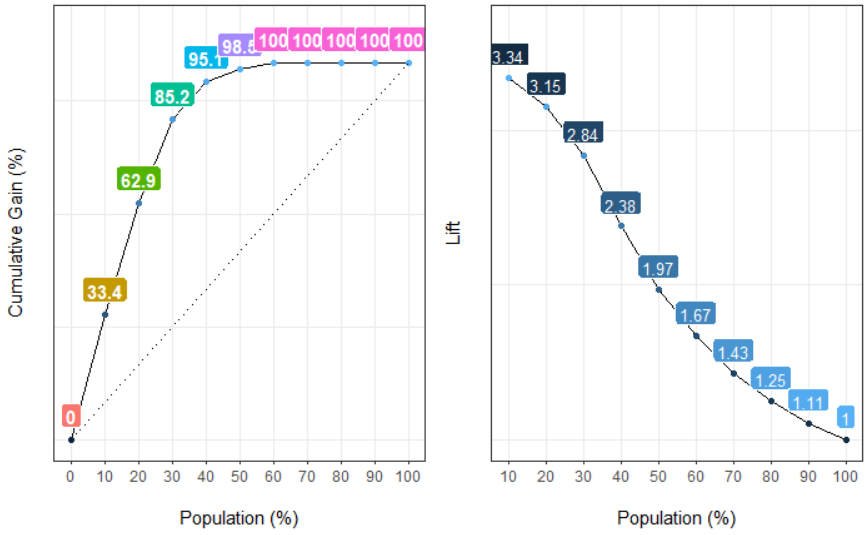
Random forest



Boosting



Bagging



Entrambi i grafici riportano sull'asse delle ascisse la percentuale di popolazione, mentre sulle ordinate è presente nel grafico a sinistra la percentuale di guadagno cumulativo che rappresenta il guadagno percentuale della probabilità pj rispetto a quella globale ed è ottenuto dalla formula seguente: ((pj-p)/p)*100 . Nel grafico di destra invece è riportata la lift, cioè la probabilità di successo del decile j rispetto alla probabilità globale: pj/p.

Il modello vincente è il random forest. Si nota ad esempio che, secondo il random forest, il primo 20% di utenti che presentano le più alte probabilità previste, catturano il 63.8% dei venditori totali.

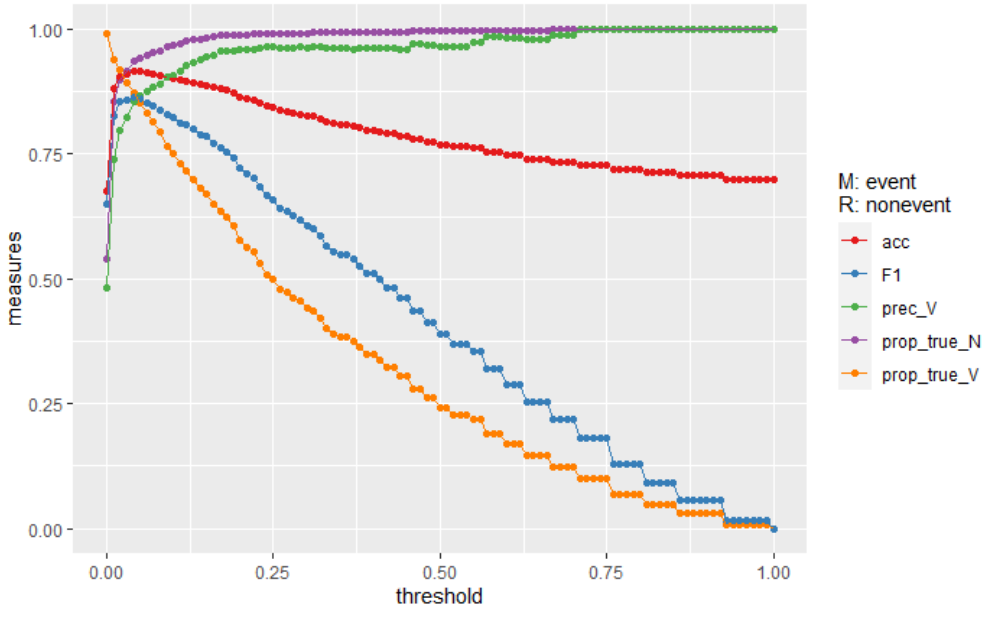
Step 3

Una volta individuato il modello vincente, ci poniamo come obiettivo quello di passare da probabilità a posteriori a target previsto. Per farlo scegliamo una soglia ottimale tale da ottenere una matrice di confusione che ci soddisfi dal punto di vista classificativo, ovvero con elevata sensitivity e basso false negative rate. Le soglie proposte coincidono con le probabilità a posteriori del modello vincente (random forest), valutato sul dataset di validation. Per ogni soglia si ottengono diverse matrici di confusione e, quindi, diverse metriche non distorte, in quanto stimate su dati mai presentati al modello fino ad ora nelle fasi precedenti. Tuttavia, le metriche ottenute non possono essere utilizzate per confrontare modelli. Infatti, in tal caso, si commetterebbe l'errore di valutare i diversi classificatori basandosi su metriche specifiche per ogni soglia e non per tutte le possibili soglie. La scelta della soglia gioca un ruolo fondamentale nel processo di classificazione, poichè la sua determinazione sarà usata come criterio di classificazione dei nuovi dati di score. Come prima cosa però aggiustiamo le posterior, in quanto il dataset è bilanciato.

```
##   threshold prop_true_V prop_true_N true_V true_N fn_V    n fp_V       acc
## 1      0.00   0.9915730   0.5392097    706    887    6 2357  758 0.6758591
## 2      0.01   0.9396067   0.8559271    669   1408   43 2357  237 0.8812049
## 3      0.02   0.9199438   0.8990881    655   1479   57 2357  166 0.9053882
## 4      0.03   0.8932584   0.9173252    636   1509   76 2357  136 0.9100552
## 5      0.04   0.8721910   0.9355623    621   1539   91 2357  106 0.9164192
## 6      0.05   0.8539326   0.9434650    608   1552  104 2357   93 0.9164192
##      prec_V        F1
## 1 0.4822404 0.6488971
## 2 0.7384106 0.8269468
## 3 0.7978076 0.8545336
## 4 0.8238342 0.8571429
## 5 0.8541953 0.8630994
## 6 0.8673324 0.8605803
```

Grafico con tutte le misure

Studiamo la soglia graficamente, osservando le variazioni delle metriche di interesse sul dataset di validation. Ricordando che il nostro obiettivo è quello di essere certi di classificare correttamente gli eventi di interesse, ovvero i venditori con buona probabilità, minimizzando il false negative rate, ci aspettiamo di scegliere una soglia piuttosto bassa.



Ricordando che il nostro interesse è quello di classificare correttamente un numero sufficientemente elevato di venditori, riteniamo come soddisfacente una quota di sensitivity pari al 90%. Grazie al grafico, siamo in grado di intuire che fissando la soglia a 0.025 siamo in grado di ottenere un valore della metrica di interesse pari a quello da noi desiderato.

A questo punto, utilizziamo la soglia stabilita e ricaviamo i valori previsti del target e valutiamo la sensitivity sulla matrice di confusione dei dati di validation.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    V
##          N 1495   66
##          V  150  646
##
##                Accuracy : 0.9084
##                  95% CI : (0.896, 0.9197)
##     No Information Rate : 0.6979
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.7897
##
##  Mcnemar's Test P-Value : 0.00000001629
##
##             Sensitivity : 0.9073
##             Specificity : 0.9088
##          Pos Pred Value : 0.8116
##          Neg Pred Value : 0.9577
##              Prevalence : 0.3021
##          Detection Rate : 0.2741
##    Detection Prevalence : 0.3377
##       Balanced Accuracy : 0.9081
##
##        'Positive' Class : V
```

Osserviamo che il false negative rate (venditori classificati come acquirenti) risulta essere più basso del false positive rate (acquirenti classificati come venditori) sul dataset di validation, come da obiettivo dell'analisi.

Step 4

Stimiamo sui dati di score il modello vincente selezionato allo step 2 e classifichiamo ogni soggetto in base alla soglia selezionata allo step 3. Questo passaggio permette di definire il target di nuovi dati indipendenti in base al modello vincente, così da valutare le reali performance su nuovi dati.

```
##  c0  c1
## 235 102
```

Nel dataset di score sono presenti 337 osservazioni, pari al 5% del dataset bilanciato. 102 sono venditori, pari al 30% circa del totale.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  c0  c1
##         c0 220  13
##         c1  15  89
##
##                Accuracy : 0.9169
##                  95% CI : (0.8822, 0.9441)
##     No Information Rate : 0.6973
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.8043
##
##  Mcnemar's Test P-Value : 0.8501
##
##             Sensitivity : 0.8725
##             Specificity : 0.9362
##          Pos Pred Value : 0.8558
##          Neg Pred Value : 0.9442
##              Prevalence : 0.3027
##          Detection Rate : 0.2641
##    Detection Prevalence : 0.3086
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : c1
```

Il nostro modello vincente ottiene buoni risultati sui dati di score, infatti la sensitivity è prossima a 0.90 e la numerosità di false negative è inferiore rispetto a quella dei false positive.