Progetto Data Mining

Giovanni Romanò, Davide Porcellini, Lucia Gerbi

11 febbraio 2019

# Trasformazione dataset

Il dataset di partenza è composto da 14758 osservazioni di 123 variabili, dove ciascuna riga contiene statistiche riguardanti una partita per entrambe le squadre partecipanti. Abbiamo considerato le partite dalla stagione 2012-2013 alla 2017-2018. Per ciascuna partita esistono due righe, ognuna dal punto di vista di una delle due squadre. Abbiamo trasformato il dataset in modo tale che per ogni partita ci fosse una sola riga, estratta casualmente tra le due della coppia, e che le variabili fossero le medie stagionali fino al giorno precedente la partita. Successivamente abbiamo eliminato alcune variabili che non abbiamo ritenuto necessarie, quindi quelle costruite come combinazione lineare di altre variabili e quelle in comune alle due squadre (minuti giocati, arbitri, etc).

## teamMin teamDayOff teamPTS teamAST teamTO  
## 302 240 2 96.70000 18.70000 14.60000  
## 303 240 2 102.36364 21.72727 16.72727  
## 306 240 2 92.25000 21.50000 15.33333

Dopo aver diviso il dataset nelle tre porzioni, il training ottenuto ha 4002 osservazioni, il validation 2060 e lo score 1214. Tutti e tre i dataset hanno 66 variabili.

# Costruzione dei modelli

Abbiamo costruito i seguenti modelli, provando a cambiare strategie di pre-processing sullo stesso modello e/o cambiando i parametri di tuning. Tutti i modelli sono stati tunati con la ROC, siccome sono di uguale importanza gli errori di classificazione di Event o nonEvent.

## Logistico

Abbiamo effettuato il pre-processing per il logistico. Abbiamo notato l’assenza di missing e di separation.

## teamPTS\_w teamAST\_w teamTO\_w teamSTL\_w   
## Min. : 79.00 Min. :13.50 Min. : 9.00 Min. : 3.000   
## 1st Qu.: 96.44 1st Qu.:20.46 1st Qu.:13.83 1st Qu.: 7.083   
## Median :100.49 Median :21.55 Median :14.76 Median : 7.742   
## Mean :100.24 Mean :21.65 Mean :14.78 Mean : 7.776   
## 3rd Qu.:103.80 3rd Qu.:22.85 3rd Qu.:15.48 3rd Qu.: 8.435   
## Max. :125.00 Max. :34.00 Max. :22.50 Max. :16.000

## teamPTS\_l teamAST\_l teamTO\_l teamSTL\_l   
## Min. : 83.00 Min. :14.00 Min. : 8.00 Min. : 3.000   
## 1st Qu.: 97.71 1st Qu.:20.67 1st Qu.:13.68 1st Qu.: 7.122   
## Median :101.80 Median :21.96 Median :14.63 Median : 7.833   
## Mean :101.50 Mean :22.16 Mean :14.58 Mean : 7.864   
## 3rd Qu.:104.88 3rd Qu.:23.36 3rd Qu.:15.35 3rd Qu.: 8.571   
## Max. :120.00 Max. :31.40 Max. :22.00 Max. :13.000

## Away Home  
## Loss 1170 843  
## Win 807 1182

Si notano alcune coppie di variabili a correlazione molto alta (>0.90). Abbiamo quindi tolto una variabile alla volta finché non abbiamo più osservato questo fenomeno. Molti aspetti del gioco erano descritti da due variabili simili, per cui abbiamo deciso di tenerne solo una (ad esempio: teamTO -palle perse- e teamTO% -% palle perse sul totale delle azioni). Non si osserva il problema di NearZero Variance.

## freqRatio percentUnique zeroVar nzv  
## teamLoc 1.024279 0.04997501 FALSE FALSE  
## teamRslt 1.012066 0.04997501 FALSE FALSE  
## teamDayOff 2.030132 0.14992504 FALSE FALSE  
## teamPTS 1.312500 79.11044478 FALSE FALSE  
## teamAST 1.000000 66.86656672 FALSE FALSE

Abbiamo proseguito costruendo due modelli attraverso due diverse model selection: AIC partendo dal modello completo e un albero prunato con cp.

## [1] "oppt2P%" "oppt3P%" "oppt3PA" "opptASST%" "opptAST/TO"  
## [6] "opptDayOff" "opptDREB%" "opptDrtg" "opptFT%" "opptFTA"   
## [11] "opptOrtg" "opptPlay%" "opptPTS" "opptSTL/TO" "opptTO%"   
## [16] "opptTREB%" "opptTS%" "team2P%" "team2PA" "team3P%"   
## [21] "team3PA" "teamASST%" "teamAST/TO" "teamBLKR" "teamDayOff"  
## [26] "teamDREB%" "teamDrtg" "teamFT%" "teamLoc" "teamOREB%"   
## [31] "teamOrtg" "teamPlay%" "teamPTS" "teamSTL%" "teamTO%"   
## [36] "teamTREB%" "teamTS%"

## [1] "oppt2P%" "oppt2PA" "oppt3P%" "oppt3PA" "opptASST%"   
## [6] "opptAST/TO" "opptBLKR" "opptDayOff" "opptDREB%" "opptDrtg"   
## [11] "opptFT%" "opptFTA" "opptOREB%" "opptOrtg" "opptPF"   
## [16] "opptPlay%" "opptPTS" "opptSTL%" "opptSTL/TO" "opptTO%"   
## [21] "opptTREB%" "opptTS%" "team2P%" "team2PA" "team3P%"   
## [26] "team3PA" "teamASST%" "teamAST/TO" "teamBLKR" "teamDayOff"  
## [31] "teamDREB%" "teamDrtg" "teamFT%" "teamFTA" "teamLoc"   
## [36] "teamOREB%" "teamOrtg" "teamPF" "teamPlay%" "teamPTS"   
## [41] "teamSTL%" "teamSTL/TO" "teamTO%" "teamTREB%" "teamTS%"

## [1] "Model selection con AIC"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.8 17.7  
## Win 16.5 32.0  
##   
## Accuracy (average) : 0.6577

## [1] "Model selection con Tree"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.8 17.8  
## Win 16.5 31.9  
##   
## Accuracy (average) : 0.6572

Abbiamo tolto i punti influenti a entrambi i modelli per verificare un eventuale miglioramento dei risultati. Abbiamo utilizzato come metrica i Dffits per eliminare i punti influenti sulle previsioni. Le osservazioni influenti sul modello selezionato con AIC sono 32, mentre sull’altro 195.

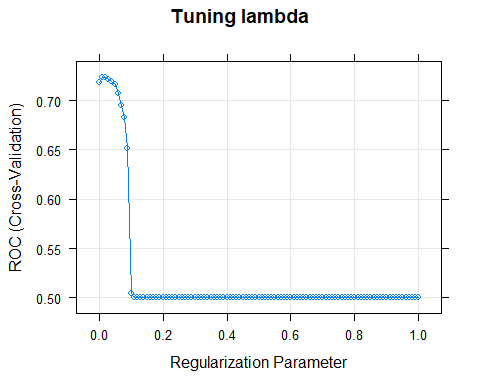
## [1] "Model selection con AIC"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.7 17.6  
## Win 16.6 32.0  
##   
## Accuracy (average) : 0.6577

## [1] "Model selection con Tree"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 34.9 16.2  
## Win 15.6 33.4  
##   
## Accuracy (average) : 0.6822

## LASSO

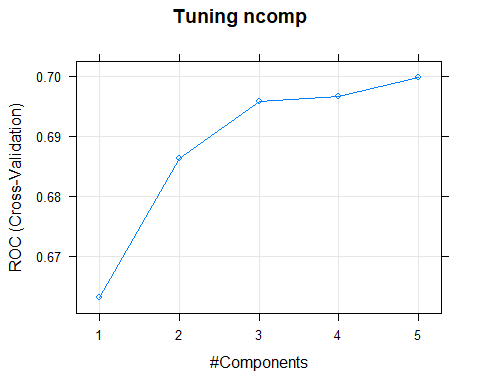
## alpha lambda  
## 2 1 0.01

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.8 17.5  
## Win 16.5 32.2  
##   
## Accuracy (average) : 0.6597

## [1] "Coefficienti LASSO"

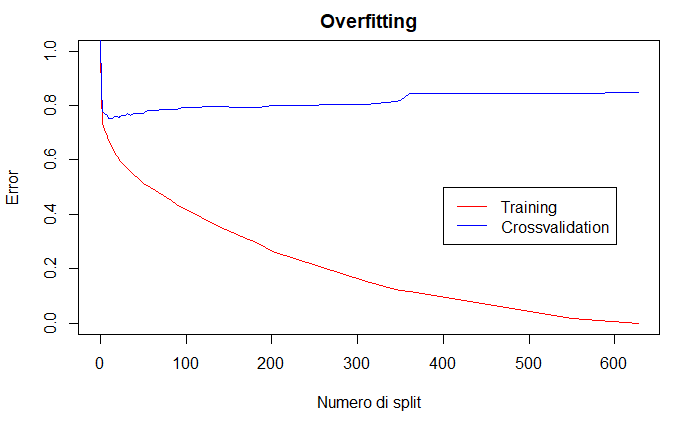
## (Intercept) teamLocHome team2PA teamOrtg teamDrtg   
## 4.9746486302 0.6935237972 -0.0262450733 0.0666709196 -0.0696237707   
## `teamPlay%` teamAR `teamAST/TO` opptDayOff1 opptDayOff3   
## 1.6297769854 0.0204615209 0.1885917918 0.1346691177 -0.0218033225   
## oppt2PA `opptTREB%` `opptTS%` `opptDREB%` opptOrtg   
## 0.0007661399 -0.0119686758 -3.3069854093 -0.0075621045 -0.0793954027   
## opptDrtg `opptAST/TO`   
## 0.0613999093 -0.0018263360

## PLS

## ncomp  
## 5 5

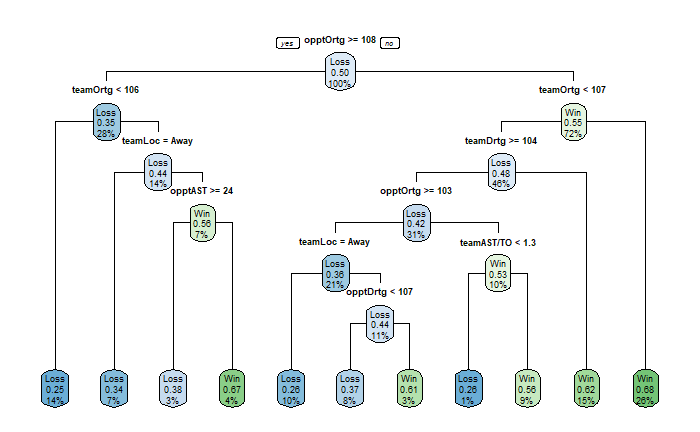
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.5 17.9  
## Win 16.8 31.8  
##   
## Accuracy (average) : 0.6532

## Tree

Abbiamo fittato l’albero largest e successivamente fatto post-pruning.

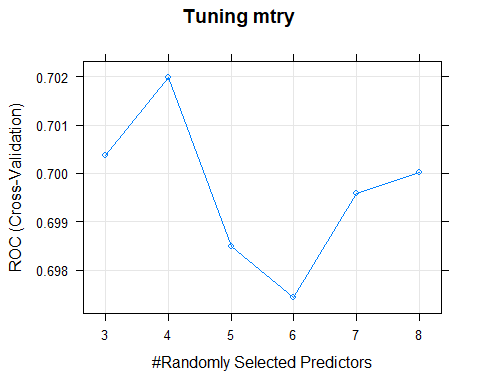
## CP nsplit rel error xerror xstd  
## 1 0.1588738059 0 1.00000000 1.0583208 0.01588131  
## 2 0.0525389643 1 0.84112619 0.8602313 0.01573491  
## 3 0.0125691302 3 0.73604827 0.7833082 0.01550819  
## 4 0.0105580694 4 0.72347914 0.7752640 0.01547877  
## 5 0.0075414781 5 0.71292107 0.7707391 0.01546174  
## 6 0.0070387129 8 0.68526898 0.7631976 0.01543257  
## 7 0.0062007709 10 0.67119155 0.7541478 0.01539629  
## 8 0.0055304173 13 0.65258924 0.7536450 0.01539423  
## 9 0.0050276521 14 0.64705882 0.7541478 0.01539629  
## 10 0.0047762695 17 0.62795375 0.7601810 0.01542063

L’albero prunato ha 11 foglie (quindi 10 split) e tra le sette variabili splittanti, le due più importanti sono l’Off Rtg delle due squadre.

## opptAST opptDrtg opptOrtg teamAST/TO teamDrtg teamLoc teamOrtg   
## 10.568997 10.123492 86.596612 7.195913 34.244623 26.425223 75.624636

## Random Forest

Seguendo il criterio di Breiman, dato che il numero delle covariate è 65, il numero ottimale di variabili estratte per ogni albero risulta essere 8. Impostiamo la griglia di tuning da 3 a 8.

## mtry  
## 2 4

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 32.6 18.0  
## Win 17.7 31.7  
##   
## Accuracy (average) : 0.6437

## rf variable importance  
##   
## Overall  
## opptOrtg 100.00  
## teamDrtg 96.93  
## `opptPlay%` 90.35  
## teamOrtg 90.16  
## `opptTS%` 88.58  
## opptDrtg 85.31  
## opptPPS 84.60  
## `teamTS%` 83.16  
## `opptEFG%` 82.69  
## `teamEFG%` 82.13  
## teamPPS 81.51  
## `teamPlay%` 79.10  
## `opptTREB%` 78.04  
## `team2P%` 75.94

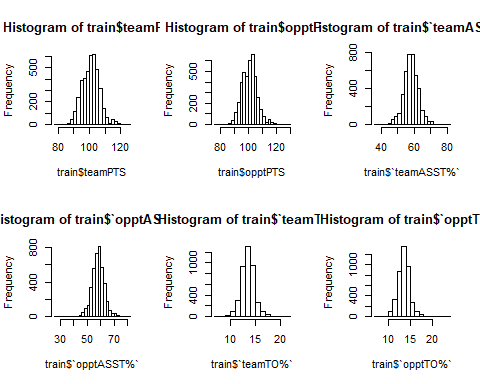
L’importanza delle variabili stampata è contaminata dal fatto che sono tenuti in considerazione gli split surrogati. Infatti, le variabili più importanti sono tra loro correlate e hanno importanza molto simile.

## Naive Bayes

Siccome è necessario togliere le variabili collineari ed è preferibile fare model selection, abbiamo recuperato il dataset usato nel logistico (model selection con l’albero). Non è presente zero-problem.

## Away Home  
## Loss 1170 843  
## Win 807 1182

##   
## 0 1 2 3 4 5+  
## Loss 31 569 1059 266 49 39  
## Win 19 493 1097 281 62 37

Inizialmente abbiamo provato ad usare un kernel gaussiano visto che le distribuzioni delle covariate sono tendenzialmente simmetriche.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Loss Win  
## Loss 695 366  
## Win 341 658  
##   
## Accuracy : 0.6568

Abbiamo comunque provato a usare una stima di densità kernel non parametrico, per verificare un possibile miglioramento di risultato.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Loss Win  
## Loss 689 365  
## Win 347 659  
##   
## Accuracy : 0.6544

A livello di Accuracy validata (hold-out) la KDE non porta miglioramenti.

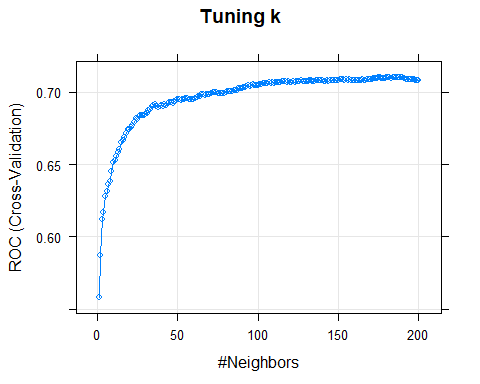
Proviamo a usare dataset dopo analisi di collinearità, ma senza model selection.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Loss Win  
## Loss 693 363  
## Win 343 661  
##   
## Accuracy : 0.6573

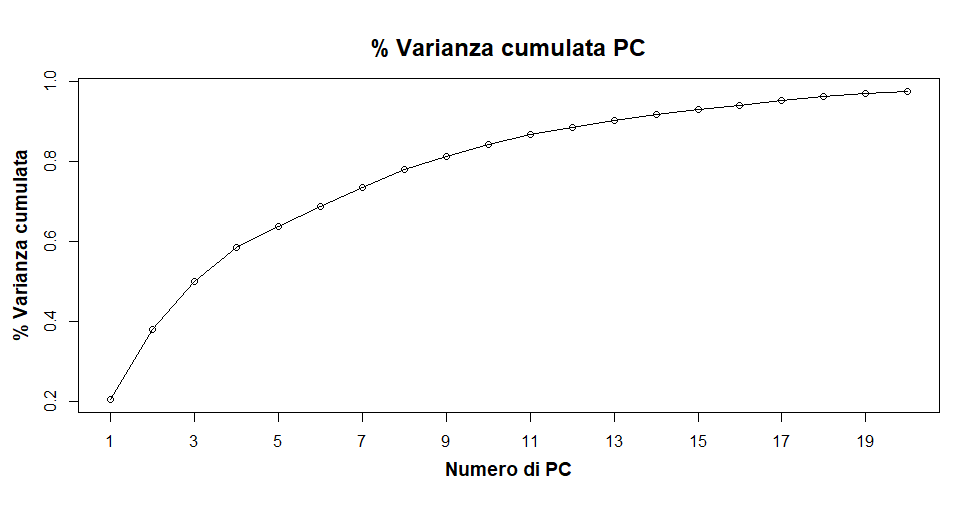
I miglioramenti a livello di Accuracy validata (hold-out) non sono significativi (cambia al terzo decimale).

## Nearest Neighbor

Abbiamo fatto model selection usando l’albero. Abbiamo tunato il parametro k per tutti i valori compresi tra 1 e 200. Le variabili in input sono state centrate e scalate.

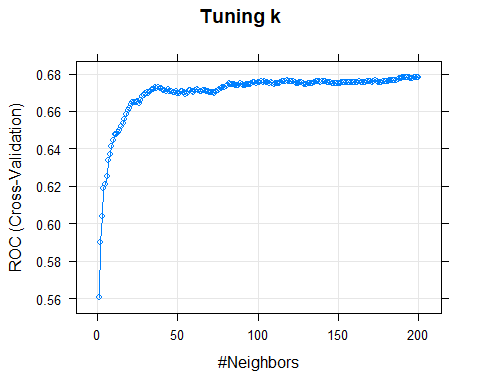
## k  
## 185 185

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.8 18.4  
## Win 16.5 31.3  
##   
## Accuracy (average) : 0.6512

Proviamo a fare le Componenti Principali per usare anche le categoriali nel Nearest Neighbor. Contemporaneamente otteniamo riduzione della dimensionalità e input centrati.

Teniamo le prime 13 pc così arriviamo a 90% di varianza riprodotta e comunque diminuiamo notevolmente il numero di variabili.

Anche in questo caso il parametro k è stato tunato per tutti i valori compresi tra 1 e 200.

## k  
## 191 191

## Cross-Validated (10 fold) Confusion Matrix   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 34.2 20.7  
## Win 16.1 29.0  
##   
## Accuracy (average) : 0.6319

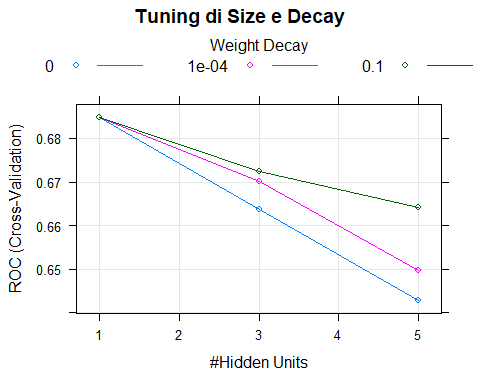
A livello di Accuracy cross-validata il modello con le PC in input è peggiore rispetto a quello con la model selection con l’albero.

## Neural Net

Necessitano analisi di collinearità (usiamo il dataset senza variabili correlate usato per il logistico), nzv (che abbiamo verificato non esserci) e serve fare model selection e centrare e scalare le variabili in input.

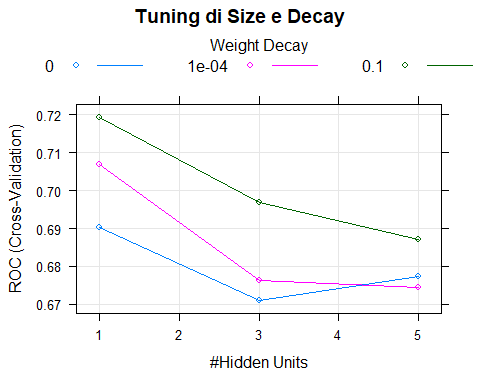
Come funzione di attivazione usiamo quella di default (sigmoid). Il parametro di size di default è tunato tra 1, 3 e 5 e quello di decay tra 0, 0.0001 e 0.1.

Neural Nets usando PC scalate (che sappiamo non essere correlate).

## size decay  
## 3 1 0.1

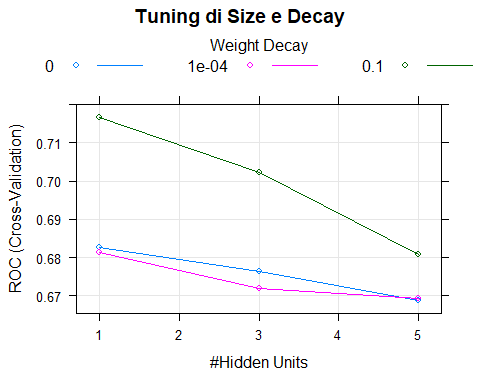
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 32.8 19.1  
## Win 17.5 30.6  
##   
## Accuracy (average) : 0.6334

Neural Nets usando model selection con albero (che sappiamo non essere correlate perché l’abbiamo usate nel logistico).

## size decay  
## 3 1 0.1

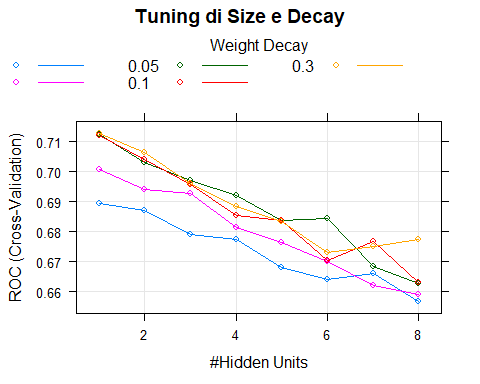
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.3 17.5  
## Win 17.0 32.2  
##   
## Accuracy (average) : 0.6549

Neural Nets usando le variabili selezionate con AIC per il modello logistico centrate e scalate (che sappiamo non essere correlate perché l’abbiamo usate nel logistico).

## size decay  
## 3 1 0.1

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.8 17.8  
## Win 16.5 31.9  
##   
## Accuracy (average) : 0.6569

Infine, abbiamo provato a fare le Neural Nets con le PC (implementate nel pre-processing di caret), ma aumentando i valori su cui tunare i parametri: per size abbiamo provato i valori compresi tra 1 e 8, mentre per decay tra 0.001, 0.01, 0.05, 0.1 e 0.3.

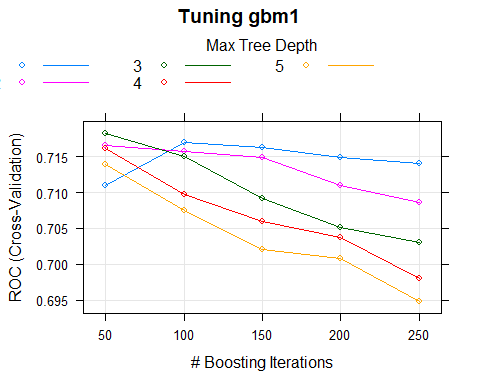
## size decay  
## 5 1 0.3

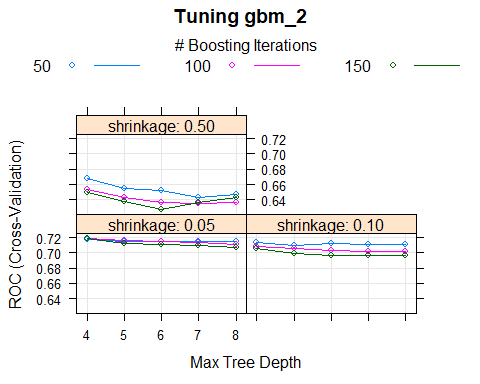
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.4 17.3  
## Win 16.9 32.4  
##   
## Accuracy (average) : 0.6579

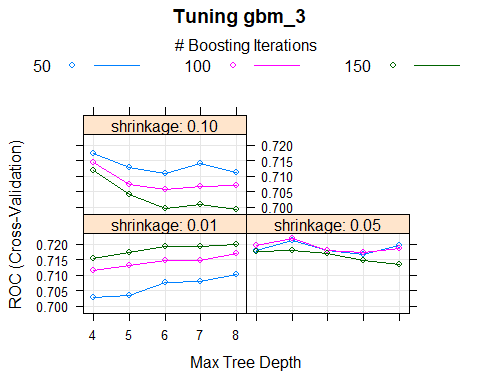
## Gradient Boosting

Abbiamo provato tre diversi Gradient Boosting:

1. gbm1 con parametri di default, quindi interaction.depth da 1 a 5, n.trees 50-100-150-200-250 e n.minobsinnode e shrinkage costanti a 10 e 0.1;
2. gbm\_2 con interaction.depth da 4 a 8, n.trees 50-100-150, shrinkage 0.05-0.1-0.5 e n.minobsinnode costante a 10;
3. gbm\_3 con interaction.depth da 4 a 8, n.trees 50-100-150, shrinkage 0.01-0.05-0.1 e n.minobsinnode costante a 10;

## n.trees interaction.depth shrinkage n.minobsinnode  
## 11 50 3 0.1 10

## n.trees interaction.depth shrinkage n.minobsinnode  
## 2 100 4 0.05 10

## n.trees interaction.depth shrinkage n.minobsinnode  
## 20 100 5 0.05 10

## [1] "Confusion Matrix di gbm1"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.4 17.8  
## Win 16.9 31.9  
##   
## Accuracy (average) : 0.6534

## [1] "Confusion Matrix di gbm\_2"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.9 17.7  
## Win 16.4 32.0  
##   
## Accuracy (average) : 0.6589

## [1] "Confusion Matrix di gbm\_3"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 34.1 17.4  
## Win 16.2 32.3  
##   
## Accuracy (average) : 0.6637

## 

## Stacking

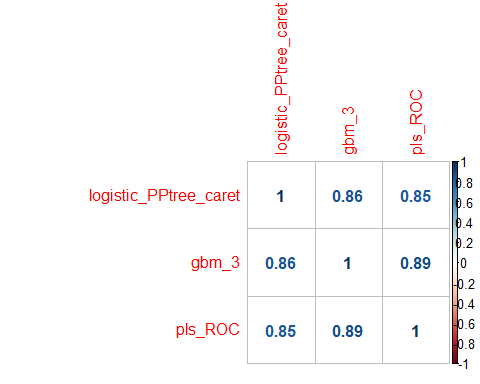
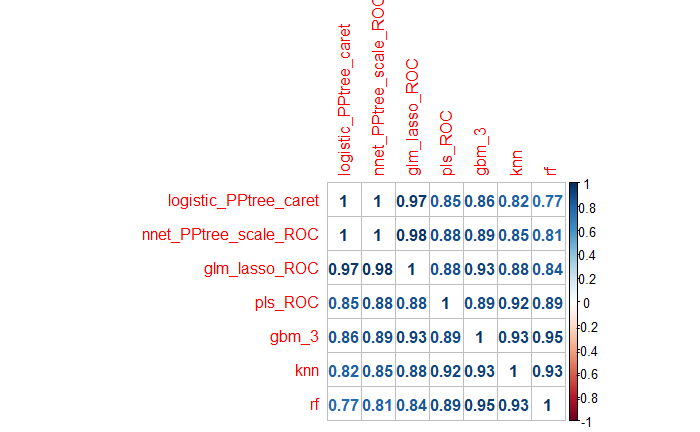
Abbiamo provato a fare uno stacking usando come meta-classificatore un modello logistico e come modelli base il logistico con model selction fatta con l’albero, il pls e un gradient boosting. Questi tre sono i modelli meno correlati tra quelli costruiti, nonostante ciò la correlazione è piuttosto alta, quindi non ci si aspetta un grande miglioramento.

## [1] "Correlazione media tutti modelli:"

## [1] 0.8919098

## [1] "Correlazione media modelli scelti:"

## [1] 0.8677552

## PWin\_glm PWin\_pls Pwin\_gbm teamRslt  
## 1 0.2331691 0.4201750 0.2690459 Win  
## 2 0.3797174 0.4619413 0.4410206 Loss  
## 3 0.7950336 0.6102570 0.3862832 Loss  
## 4 0.3670420 0.4939860 0.5194413 Win  
## 5 0.6104161 0.4224053 0.5692691 Win  
## 6 0.8159753 0.6518714 0.4917751 Loss

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.9 17.6  
## Win 16.4 32.1  
##   
## Accuracy (average) : 0.6599

# Modelli con statistiche semplici

Ci siamo chiesti quanto sarebbero cambiati i risultati tenendo soltanto le statistiche più semplici (come punti, rimbalzi, assist, etc). Se i modelli costruiti su queste covariate dovessero produrre risultati uguali a quelli dei modelli costruiti fino a qui, sarebbero da preferire poiché utilizzano dati più facilmente reperibili.

## teamLoc teamRslt teamDayOff teamPTS teamAST teamTO teamSTL teamBLK  
## teamPF team2PA team2P% team3PA team3P% teamFTA teamFT% teamORB teamDRB

## opptDayOff opptPTS opptAST opptTO opptSTL opptBLK opptPF oppt2PA  
## oppt2P% oppt3PA oppt3P% opptFTA opptFT% opptORB opptDRB

## Logistico

Abbiamo già verificato l’assenza di missing value e di separation. Ricontrolliamo l’assenza di variabili collineari.

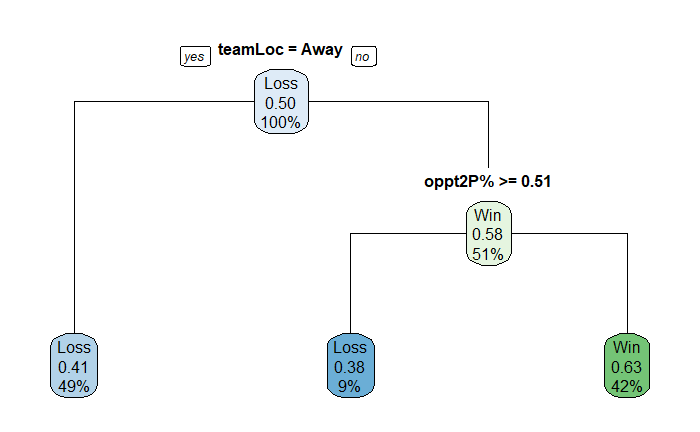
## [1] "Variabili correlate:"

## character(0)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.3 17.7  
## Win 17.0 32.0  
##   
## Accuracy (average) : 0.6529

## Tree

Abbiamo inizialmente fittato l’albero largest e successivamente fatto post-pruning secondo la regola del min(xerror)+sd.

## CP nsplit rel error xerror xstd  
## 1 0.1704374057 0 1.00000000 1.0583208 0.01588131  
## 2 0.0417295123 1 0.82956259 0.8295626 0.01565623  
## 3 0.0286576169 2 0.78783308 0.7908497 0.01553477  
## 4 0.0077928607 3 0.75917547 0.7858220 0.01551715  
## 5 0.0075414781 5 0.74358974 0.7838110 0.01550999  
## 6 0.0063683593 6 0.73604827 0.7823027 0.01550457  
## 7 0.0055304173 9 0.71694319 0.7797888 0.01549545  
## 8 0.0037707391 10 0.71141277 0.7817999 0.01550275  
## 9 0.0037204625 12 0.70387129 0.7812971 0.01550093  
## 10 0.0035193565 18 0.68124686 0.7878331 0.01552425

## [1] "Importanza delle variabili:"

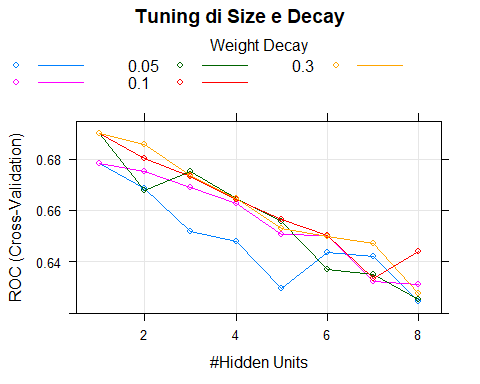
## oppt2P% teamLoc   
## 34.62899 61.62908

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Loss Win  
## Loss 712 508  
## Win 324 516  
##

## Accuracy : 0.5961

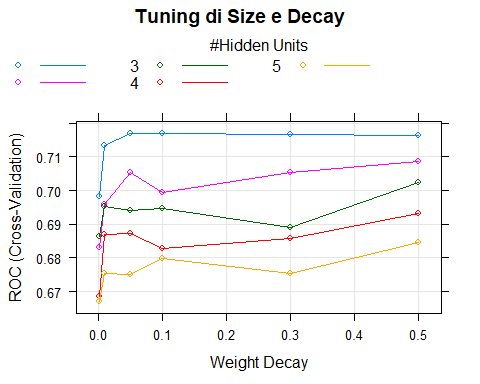
## Neural Net

Neural Net con pre-processing di caret (center, scale e pca), con size da 1 a 8 e decay 0.001-0.01-0.05-0.1-0.3.

## size decay  
## 5 1 0.3

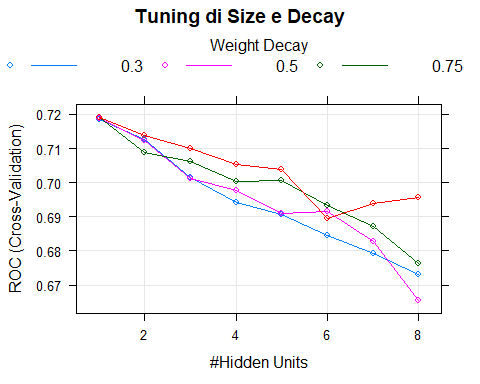
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 32.4 18.0  
## Win 17.9 31.7  
##   
## Accuracy (average) : 0.6402

Neural Net con pre-processing di caret (center e scale), con size da 1 a 5 e decay 0.001-0.01-0.05-0.1-0.3-0.5.

## size decay  
## 3 1 0.05

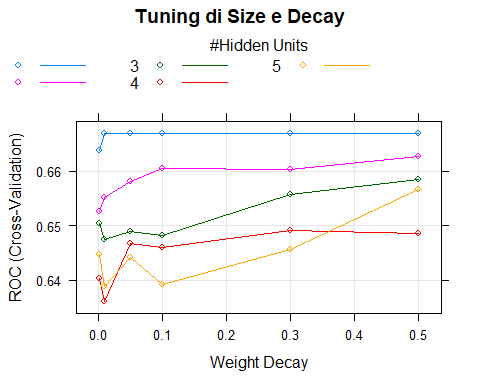
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.7 17.2  
## Win 16.6 32.5  
##   
## Accuracy (average) : 0.6617

Neural Net con pre-processing di caret (center), con size da 1 a 8 e decay 0.1-0.3-0.5-0.75.

## size decay  
## 4 1 0.75

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.9 17.4  
## Win 16.4 32.3  
##   
## Accuracy (average) : 0.6624

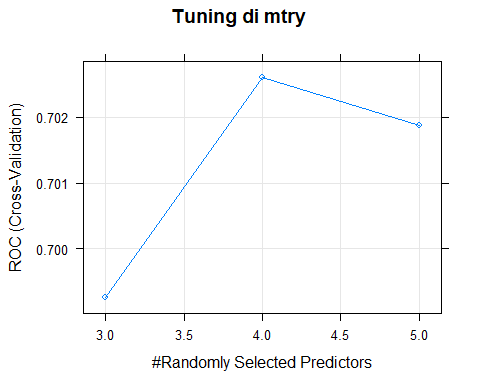
Neural Net con model selection con albero e pre-processing di caret (center e scale), con size da 1 a 5 e decay 0.001-0.01-0.05-0.1-0.3-0.5.

## size decay  
## 6 1 0.5

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 31.7 18.4  
## Win 18.6 31.3  
##   
## Accuracy (average) : 0.6302

## Random Forest

Seguendo il criterio consigliato da Breiman, siccome le covariate sono 31, il parametro mtry è stato tunato per un intervallo di valori centrato in 5 (da 2 a 8).

## mtry  
## 2 4

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 32.9 17.3  
## Win 17.4 32.4  
##   
## Accuracy (average) : 0.6527

## rf variable importance  
##   
## only 20 most important variables shown (out of 39)  
##   
## Overall  
## `oppt2P%` 100.00  
## `team2P%` 95.00  
## opptPTS 90.93  
## `oppt3P%` 89.68  
## team2PA 82.91  
## teamPTS 82.51  
## `team3P%` 82.46  
## opptDRB 80.67  
## opptTO 79.93  
## teamAST 79.61  
## opptAST 78.07  
## teamBLK 77.95  
## teamTO 76.02  
## team3PA 75.98  
## opptBLK 75.66  
## `opptFT%` 74.53  
## teamDRB 74.50  
## opptPF 74.44  
## oppt3PA 74.18  
## `teamFT%` 73.02

Come per la random forest fatta in precedenza l’importanza delle variabili è influenzata dal fatto che siano tenuti in conto gli split surrogati e che le variabili sono molto correlate.

## Naive Bayes

Abbiamo verificato l’assenza dello zero problem.

Away Home  
## Loss 1170 843  
## Win 807 1182

## 0 1 2 3 4 5+  
## Loss 31 569 1059 266 49 39  
## Win 19 493 1097 281 62 37

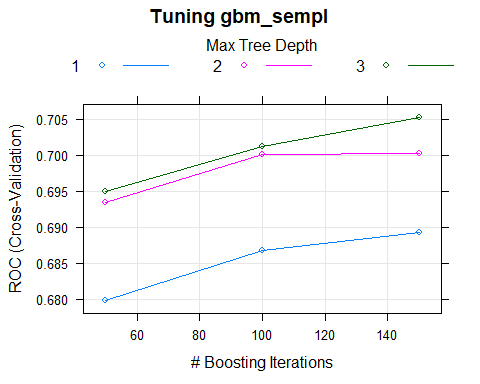
Visto che avevamo già verificato in precedenza la simmetria delle distribuzioni delle covariate continue, per stimarne la distribuzione di densità abbiamo usato il kernel gaussiano. Abbiamo effettuato model selection utilizzando l’albero precedentemente fittato.

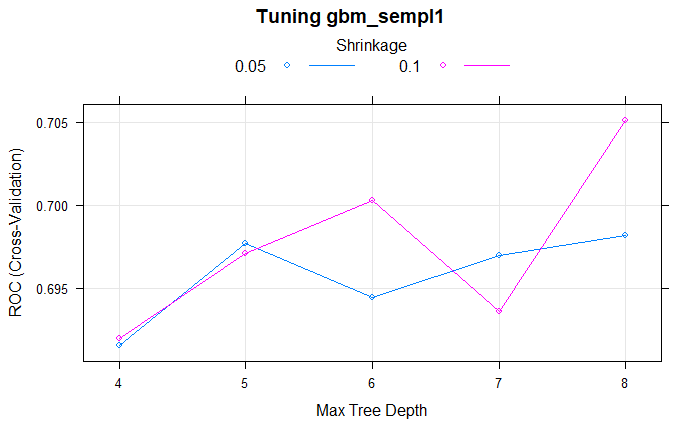
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Loss Win  
## Loss 588 344  
## Win 448 680  
##   
## Accuracy : 0.6155

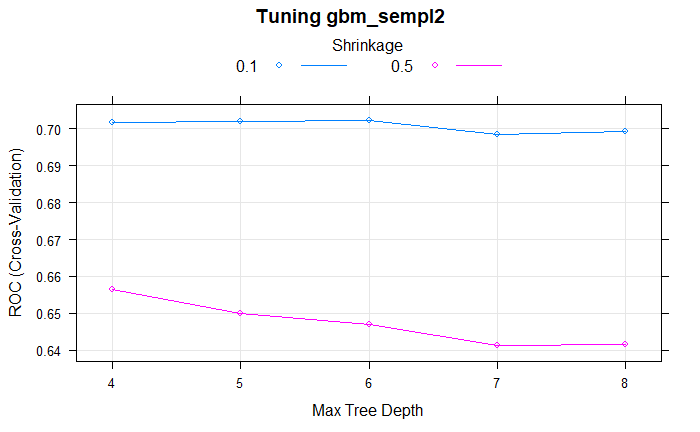
## Gradient Boosting

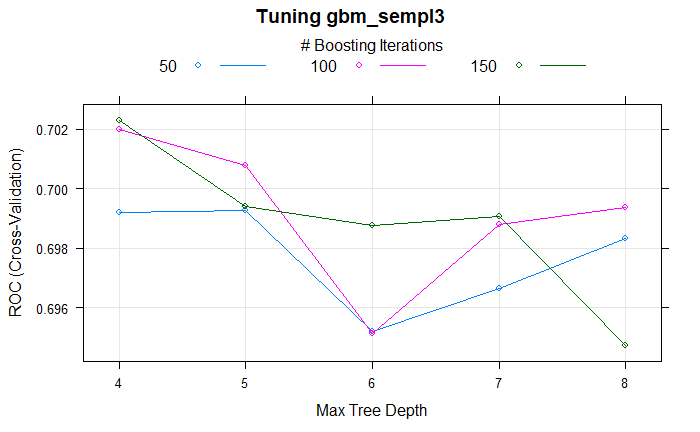
Abbiamo costruito i seguenti quattro modelli:

1. gbm\_sempl con i parametri di default, cioè interaction.depth da 1 a 3, n.trees 50-100-150 e n.minobsinnode e shrinkage costanti a 10 e 0.1;
2. gbm\_sempl1 con interaction.depth da 4 a 8, shrinkage 0.05-0.1 e n.trees e n.minobsinnode costanti a 50 e 10;
3. gbm\_sempl2 con interaction.depth da 4 a 8, shrinkage 0.1-0.5, e n.trees e n.minobsinnode costanti a 100 e 10;
4. gbm\_sempl3 con interacrion.depth da 4 a 8, n.trees 50-100-150 e shrinkage e n.minobsinnode costanti a 0.1 e 10.

## n.trees interaction.depth shrinkage n.minobsinnode  
## 9 150 3 0.1 10

## n.trees interaction.depth shrinkage n.minobsinnode  
## 10 50 8 0.1 10

## n.trees interaction.depth shrinkage n.minobsinnode  
## 3 100 6 0.1 10

## n.trees interaction.depth shrinkage n.minobsinnode  
## 3 150 4 0.1 10

## [1] "Confusion Matrix di gbm\_sempl"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.2 17.8  
## Win 17.1 31.9  
##   
## Accuracy (average) : 0.6502

## [1] "Confusion Matrix di gbm\_sempl1"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.5 17.4  
## Win 16.8 32.3  
##   
## Accuracy (average) : 0.6579

## [1] "Confusion Matrix di gbm\_sempl2"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.5 18.0  
## Win 16.8 31.7  
##   
## Accuracy (average) : 0.6517

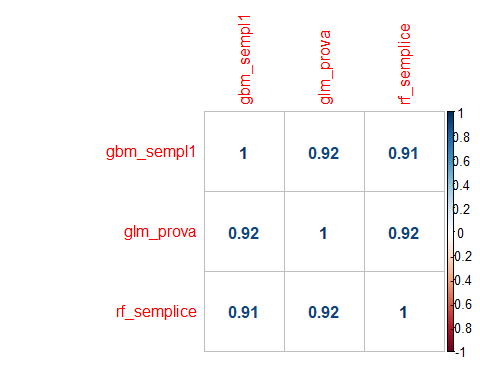
## [1] "Confusion Matrix di gbm\_sempl3"

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 32.6 17.8  
## Win 17.7 31.9  
##   
## Accuracy (average) : 0.6447

## Stacking

Per lo stacking abbiamo usato un logistico come meta-classificatore e come modelli base un logistico, una random forest e un gradient boosting. Nonstante abbiamo scelto i modelli meno correlati, la correlazione media rimane elevata quindi non ci aspettiamo grandi miglioramenti.

## [1] "Correlazione media:"

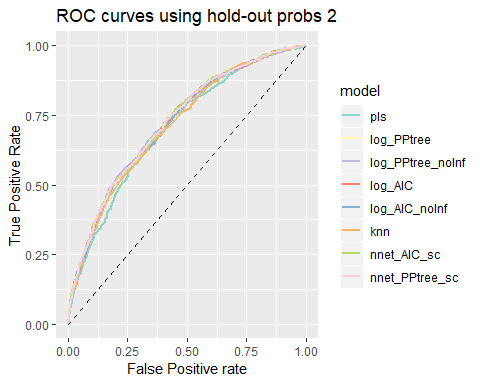
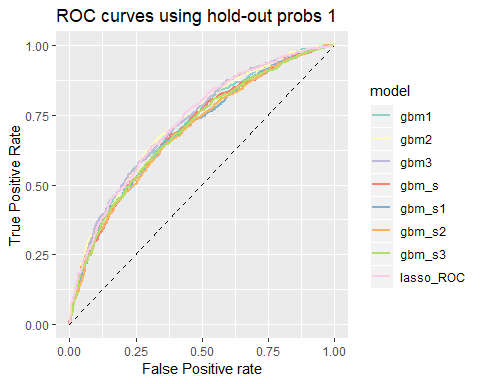
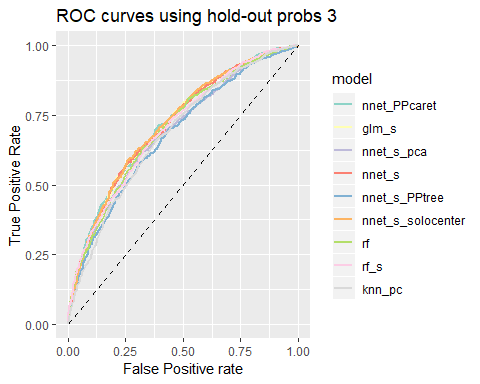
## [1] 0.9167112

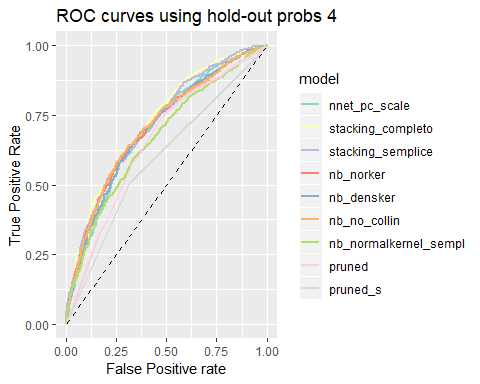
## Di seguito il dataset con le previsioni cross-validate dei modelli base usati nello stacking.

## PWin\_glm PWin\_gbm Pwin\_rf teamRslt  
## 1 0.4802468 0.1650502 0.410 Win  
## 2 0.5094718 0.4048968 0.524 Loss  
## 3 0.7420732 0.3622721 0.524 Loss  
## 4 0.7291716 0.4074137 0.520 Win  
## 5 0.4788468 0.4306951 0.478 Win  
## 6 0.5228034 0.2864595 0.514 Loss

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction Loss Win  
## Loss 33.6 17.4  
## Win 16.7 32.3  
##   
## Accuracy (average) : 0.6589

# Assesment

Abbiamo costruito i grafici delle curve ROC di tutti i modelli che abbiamo fittato.



Siccome non è presente una curva evidentemente migliore delle altre, abbiamo deciso di calcolare l’area sotto ciascuna curva.

# AUC

## model AUC  
## 1 stacking\_completo 0.7263306  
## 2 glm\_lasso\_ROC 0.7244783  
## 3 nnet\_logAIC\_scale\_ROC 0.7234283  
## 4 nnet\_PPtree\_scale\_ROC 0.7230588  
## 5 gbm\_3 0.7221237  
## 6 logistic\_PPtree\_caret 0.7220153  
## 7 gbm\_2 0.7220068  
## 8 logistic\_AIC\_caret 0.7210736  
## 9 logistic\_PPtree\_noInf\_ROC 0.7202346  
## 10 stacking\_semplice 0.7197247  
## 11 logistic\_AIC\_noInf\_ROC 0.7187632  
## 12 nnet\_semplice\_solocenter 0.7172239  
## 13 nnet\_ultimo 0.7166008  
## 14 knn 0.7139808  
## 15 glm\_prova 0.7137201  
## 16 gbm1 0.7132941  
## 17 nnet\_semplice\_nopca 0.7127426  
## 18 nb\_no\_collin 0.7079352  
## 19 pls\_ROC 0.7072952  
## 20 nb\_normalkernel 0.7050470  
## 21 rf\_semplice 0.7047906  
## 22 knn\_pc 0.7047906  
## 23 nnet\_pc\_scale\_ROC 0.7047906  
## 24 rf 0.7042373  
## 25 nb\_densitykernel 0.7026188  
## 26 gbm\_sempl 0.6987946  
## 27 gbm\_sempl3 0.6973882  
## 28 gbm\_sempl1 0.6967170  
## 29 nnet\_semplice 0.6916900  
## 30 gbm\_sempl2 0.6898311  
## 31 nnet\_semplice\_PPtree 0.6778729  
## 32 nb\_normalkernel\_sempl 0.6672514  
## 33 pruned 0.6583473  
## 34 tree\_prova\_pruned 0.5938311

Coerentemente con il grafico delle curve ROC, nessun modello ha un AUC decisamente maggiore degli altri. Perciò per procedere con la scelta del modello migliore calcoliamo Gain e Lift.

# Gain e Lift

Di seguito gain e lift dei primi due decili.

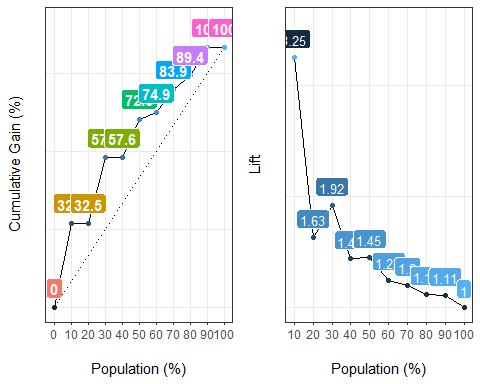
## model Population Gain Lift Score.Point  
## 1 tree\_prova\_pruned 20 50.39 2.52 0.6260454  
## 2 pruned 20 32.52 1.63 0.6810345  
## 3 gbm\_2 20 31.84 1.59 0.6558679  
## 4 knn 20 31.45 1.57 0.5913978  
## 5 stacking\_semplice 20 31.45 1.57 0.6841889  
## 6 gbm\_3 20 31.35 1.57 0.6583665  
## 7 rf\_semplice 20 31.25 1.56 0.6120000  
## 8 stacking\_completo 20 31.15 1.56 0.6908748  
## 9 gbm1 20 31.05 1.55 0.6534333  
## 10 gbm\_sempl1 20 31.05 1.55 0.6523100

Di seguito gain e lift dei primi tre decili.

## model Population Gain Lift Score.Point  
## 1 pruned 30 57.62 1.92 0.6202946  
## 2 tree\_prova\_pruned 30 50.39 1.68 0.6260454  
## 3 stacking\_completo 30 45.51 1.52 0.6204691  
## 4 knn 30 44.92 1.50 0.5537634  
## 5 logistic\_PPtree\_caret 30 44.82 1.49 0.6111799  
## 6 gbm\_3 30 44.73 1.49 0.6020471  
## 7 logistic\_AIC\_caret 30 44.63 1.49 0.6104817  
## 8 logistic\_AIC\_noInf\_ROC 30 44.53 1.48 0.6135612  
## 9 glm\_lasso\_ROC 30 44.43 1.48 0.5905220  
## 10 nnet\_logAIC\_scale\_ROC 30 44.43 1.48 0.6292563

Nei primi due decili i modelli migliori sono i due alberi, mentre considerando anche il terzo decile l’albero che ha in input tutte le statistiche (e non solo quelle semplici) risulta essere il migliore con distacco. Inoltre l’altro albero, producendo solo due split, rischiava di fornire una regola decisionale eccessivamente semplice.

Di seguito gain e lift dell’albero costruito su tutte le statistiche.

## Population Gain Lift Score.Point  
## 1 10 32.52 3.25 0.6810345  
## 2 20 32.52 1.63 0.6810345  
## 3 30 57.62 1.92 0.6202946  
## 4 40 57.62 1.44 0.6202946  
## 5 50 72.56 1.45 0.5611702  
## 6 60 74.90 1.25 0.3762376  
## 7 70 83.89 1.20 0.3389831  
## 8 80 89.36 1.12 0.2628866  
## 9 90 100.00 1.11 0.2477064  
## 10 100 100.00 1.00 0.2477064

# Scelta della soglia per il modello migliore

## threshold prop\_true\_Event prop\_true\_nonEvent Accuracy Precision\_Event  
## 49 0.48 0.7255859 0.5395753 0.6320388 0.6090164  
## 50 0.49 0.7255859 0.5395753 0.6320388 0.6090164  
## 51 0.50 0.7255859 0.5395753 0.6320388 0.6090164  
## 52 0.51 0.7255859 0.5395753 0.6320388 0.6090164  
## 53 0.52 0.7255859 0.5395753 0.6320388 0.6090164  
## 54 0.53 0.7255859 0.5395753 0.6320388 0.6090164  
## 55 0.54 0.7255859 0.5395753 0.6320388 0.6090164  
## 56 0.55 0.7255859 0.5395753 0.6320388 0.6090164  
## 57 0.56 0.7255859 0.5395753 0.6320388 0.6090164  
## 58 0.57 0.6093750 0.6361004 0.6228155 0.6233766  
## 59 0.58 0.6093750 0.6361004 0.6228155 0.6233766

La metrica di interesse per il nostro modello è l’Accuracy siccome Event e nonEvent hanno lo stesso peso (e perciò non preferiamo una tra Specificity e Sensitivity). Il massimo dell’Accuracy si osserva con soglia da 0.38 a 0.56. Però notiamo una notevole differenza fra Specificity e Sensitivity, perciò scegliamo come soglia 0.57, sacrificando meno dell’1% di Accuracy, ma bilanciando i risultati in modo soddisfacente.