Elaborato\_ML\_Schiavi\_Nada

Marco Schiavi ; Edoardo Nada

## INTRODUZIONE

Il dataset scaricato da Kaggle è relativo ad un sondaggio telefonico sulla salute raccolto ogni anno dal CDC “centre for disease control and prevention”.

il dataset contiene 22 variabili e 253680 osservazioni

1. **DIABETES\_BINARY** : 0= non diabetico 1= diabetico” o prediabetico da gravidanza, ovaio policistico ecc.…”
2. **HIGHBP**: alta pressione sanguigna 0=no 1=si
3. **HIGHCHOL**: colesterolo nel sangue alto 0=no 1=si
4. **CHOLCHECK**: fatto check colesterolo nell’ultimo anno 0=no 1=si
5. **BMI**: indice di massa grassa
6. **SMOKER**: fumato almeno 100 sigarette nella propria vita 0=no 1=si
7. **STROKE**: hai mai avuto ictus 0=no 1=si
8. **HERTDISEASORATTACK**: hai mai avuto attacchi di cuore o problemi alle coronarie 0=no 1=si
9. **PHYSACTIVITY**: praticato attività fisica” non lavoro” negli ultimi 30 giorni 0=no 1=si
10. **FRUITS**: mangi almeno un frutto al giorno 0=no 1=si
11. **VAGGIES**: mangi almeno una versura al giorno 0=no 1=si
12. **HVYALCOOHLCONSUMP**: forte bevitore uomo +14 drink a settimana donna + 7 drink 0=no 1=si
13. **ANYHEALTHCARE**: hai assicurazione sanitaria 0= no 1=si
14. **NODOCBCCOST**: ultimo anno hai avuto bisogno di un medico ma non lo hai consultato essendo troppo costoso 0=no 1=si
15. **GENHLTH**: come giudici la tua salute mentale in una scala da 1=ottima 5 =pessima
16. **MENTHLTH**: negli ultimi 30 giorni per quanti giorni la tua salute mentale non e stata buona
17. **PHYSHLT**: negli ultimi 30 giorni per quanti giorni la tua salute fisica non e stata buona
18. **DIFFWAL**: hai difficolta a camminare 0= no 1=si
19. **SEX**: 0=femmina 1=maschio
20. **AGE**: in quale categoria di età sei [1-13] nella 1 i più giovani nella 13 i più anziani
21. **EDUCATION**: livello educazione [1-6] dove 1 è nessuno titolo di studio e 6 massimo titolo di studio
22. **INCOME**: scale di reddito [1-8] dove 1 è la più bassa e 8 la più alta

OBBIETTIVO:

Analizzare con diversi “Modelli” i dati per cercare di arrivare ad utilizzare quello con una capacita previsiva affidabile dei soggetti diabetici migliore .

La letteratura ci fornisce diverse informazioni riguardanti alcune covariate e la loro eventuale influenza nell’insorgenza del diabete

(Fonte: https://www.epicentro.iss.it/diabete/epidemiologia-italia)

## Correzione del dataset: Missing Correlazione e Zero Variance

file1 <- **read.csv**("diabetes\_binary\_health\_indicators\_BRFSS2015.csv", sep=";",na.strings=**c**("NA","NaN", ""))  
**options**(scipen = 999, digits = 3)  
  
  
*# SISTEMARE NATURA VARIABILI#*  
file1**$**Diabetes\_binary<-**factor**(file1**$**Diabetes\_binary)  
file1**$**HighBP <-**factor**(file1**$**HighBP)  
file1**$**HighChol<-**factor**(file1**$**HighChol)  
file1**$**CholCheck<-**factor**(file1**$**CholCheck)  
*#file1$BMI<-numeric(file1$BMI)*  
file1**$**Smoker<-**factor**(file1**$**Smoker)  
file1**$**Stroke<-**factor**(file1**$**Stroke)  
file1**$**HeartDiseaseorAttack<-**factor**(file1**$**HeartDiseaseorAttack)  
file1**$**PhysActivity<-**factor**(file1**$**PhysActivity)  
file1**$**Fruits<-**factor**(file1**$**Fruits)  
file1**$**Veggies<-**factor**(file1**$**Veggies)  
file1**$**HvyAlcoholConsump<-**factor**(file1**$**HvyAlcoholConsump)  
file1**$**AnyHealthcare<-**factor**(file1**$**AnyHealthcare)  
file1**$**NoDocbcCost<-**factor**(file1**$**NoDocbcCost)  
file1**$**GenHlth<-**factor**(file1**$**GenHlth)  
*#file1$MentHlth<-int(file1$MentHlth)*  
*#file1$PhysHlth<-int(file1$PhysHlth)*  
file1**$**DiffWalk<-**factor**(file1**$**DiffWalk)  
file1**$**Sex<-**factor**(file1**$**Sex)  
file1**$**Age<-**factor**(file1**$**Age)  
file1**$**Education<-**factor**(file1**$**Education)  
file1**$**Income<-**factor**(file1**$**Income)  
file1**$**Diabetes\_binary<-**factor**(file1**$**Diabetes\_binary, levels = **c**(1,0), labels=**c**('c1','c0'))  
*# c0= non diabetico"0" c1=diabetico"1"*  
*#str(file1)*  
  
*# ANALIZZO SE CLASSI SONO DISTRIBUITE EQUAMENTE NEI DATASET*  
**table**(file1**$**Diabetes\_binary)

##   
## c1 c0   
## 35346 218334

*# column %*  
**prop.table**(**table**(file1**$**Diabetes\_binary))

##   
## c1 c0   
## 0.139 0.861

*# controllo multicollinearita*   
**library**(psych)  
**library**(corrgram)  
**require**(corrgram)  
**library**(caret)

**library**(car)

predictors=**c**('Diabetes\_binary','HighBP','HighChol','CholCheck','BMI','Smoker','Stroke','HeartDiseaseorAttack','PhysActivity','Fruits','Veggies','HvyAlcoholConsump','AnyHealthcare','NoDocbcCost','GenHlth','MentHlth','PhysHlth','DiffWalk','Sex','Age','Education','Income')  
predictors\_df <- file1[predictors] *# X variables*  
numeric <- **sapply**(predictors\_df, **function**(x) **is.numeric**(x))  
numeric <-predictors\_df[, numeric]  
more8=predictors\_df**$**Diabetes\_binary  
numeric <-**cbind**(numeric, more8)  
numeric <- **sapply**(predictors\_df, **function**(x) **is.numeric**(x))  
numeric <-predictors\_df[, numeric]  
correlatedPredictors = **findCorrelation**(**cor**(numeric), cutoff = 0.90, names = TRUE)  
correlatedPredictors

## character(0)

*# controllo Missing #*  
**library**(VIM)

**library**(datasets)  
**sapply**(file1, **function**(x)(**sum**(**is.na**(x))))

## Diabetes\_binary HighBP HighChol   
## 0 0 0   
## CholCheck BMI Smoker   
## 0 0 0   
## Stroke HeartDiseaseorAttack PhysActivity   
## 0 0 0   
## Fruits Veggies HvyAlcoholConsump   
## 0 0 0   
## AnyHealthcare NoDocbcCost GenHlth   
## 0 0 0   
## MentHlth PhysHlth DiffWalk   
## 0 0 0   
## Sex Age Education   
## 0 0 0   
## Income   
## 0

*# controllo zero variance o near zero variance*  
nzv = **nearZeroVar**(file1, saveMetrics = TRUE)  
**head**(nzv[**order**(nzv**$**percentUnique, decreasing = FALSE), ], n = 20)

## freqRatio percentUnique zeroVar nzv  
## Diabetes\_binary 6.18 0.000788 FALSE FALSE  
## HighBP 1.33 0.000788 FALSE FALSE  
## HighChol 1.36 0.000788 FALSE FALSE  
## CholCheck 25.79 0.000788 FALSE TRUE  
## Smoker 1.26 0.000788 FALSE FALSE  
## Stroke 23.65 0.000788 FALSE TRUE  
## HeartDiseaseorAttack 9.62 0.000788 FALSE FALSE  
## PhysActivity 3.11 0.000788 FALSE FALSE  
## Fruits 1.73 0.000788 FALSE FALSE  
## Veggies 4.30 0.000788 FALSE FALSE  
## HvyAlcoholConsump 16.79 0.000788 FALSE FALSE  
## AnyHealthcare 19.43 0.000788 FALSE TRUE  
## NoDocbcCost 10.88 0.000788 FALSE FALSE  
## DiffWalk 4.94 0.000788 FALSE FALSE  
## Sex 1.27 0.000788 FALSE FALSE  
## GenHlth 1.18 0.001971 FALSE FALSE  
## Education 1.54 0.002365 FALSE FALSE  
## Income 2.09 0.003154 FALSE FALSE  
## Age 1.03 0.005125 FALSE FALSE  
## MentHlth 13.46 0.012220 FALSE FALSE

file1**$**Stroke<-NULL  
file1**$**CholCheck<-NULL  
file1**$**AnyHealthcare<-NULL

## Creazione dataset Training, Validation, Step\_4

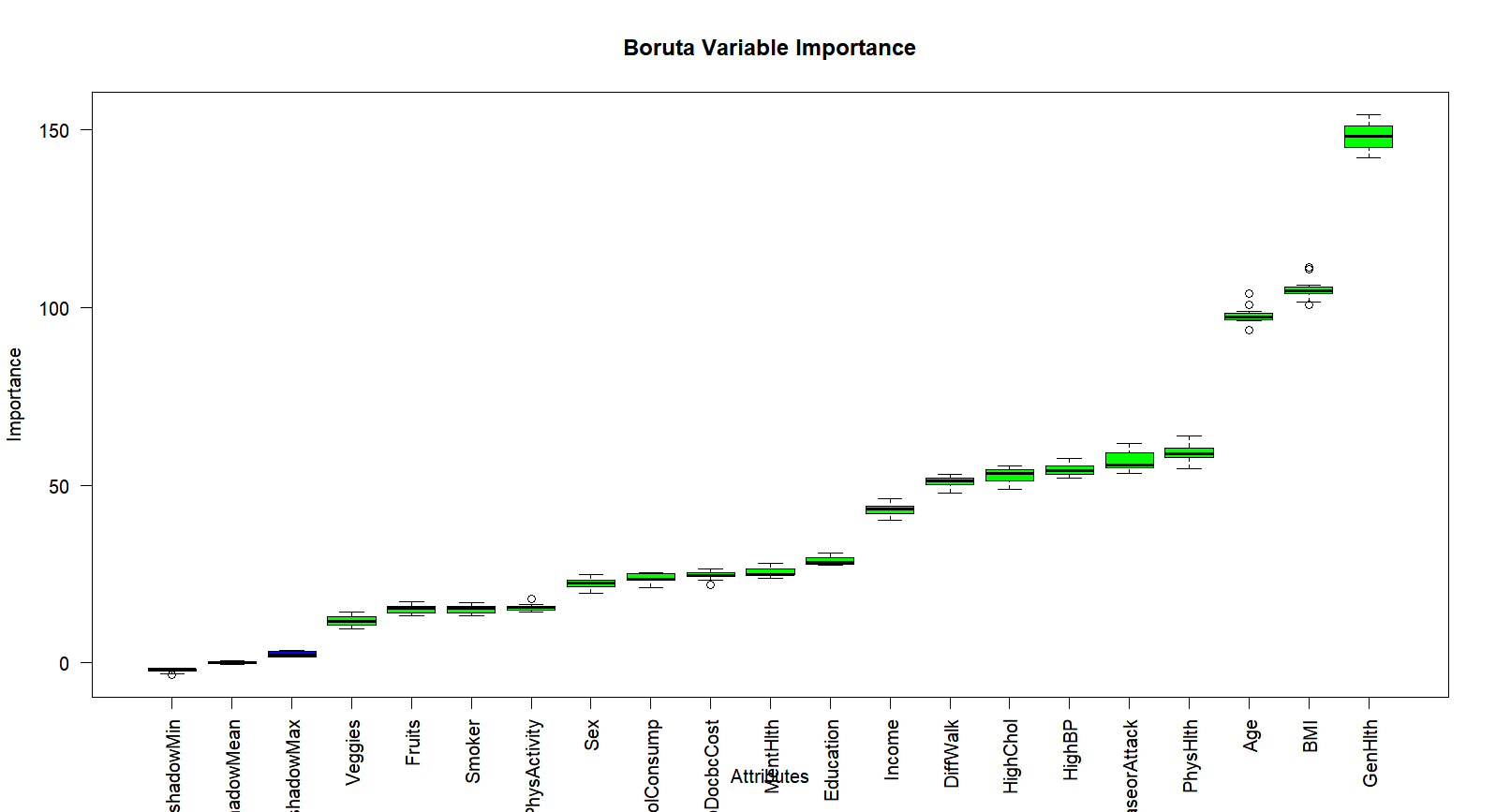
*# Dividi il dataset (70% training, 30% test)*  
*#creazione DATASET step 4*  
**set.seed**(1234)   
Dati\_partizionati <- **createDataPartition**(file1**$**Diabetes\_binary, p = 0.8, list = FALSE)  
  
*# Dati per creare e controllare il modello*  
Dati <- file1[Dati\_partizionati, ]  
  
*# Dati per controllo Finale Step\_4*  
dati\_step\_4 <- file1[**-**Dati\_partizionati, ]  
  
  
train\_indices <- **createDataPartition**(Dati**$**Diabetes\_binary, p = 0.7, list = FALSE)  
  
*# Training set*  
dati\_training <- Dati[train\_indices, ]  
  
*# Validation set*  
dati\_validation <- Dati[**-**train\_indices, ]

## Model Selection Boruta

**library**(Boruta)

**library**(caret)  
  
  
set.seed(123)boruta\_output <- Boruta(Diabetes\_binary ~ .,

data = file1,  
 doTrace = 2, *# Livello di logging (2 fornisce dettagli)*  
 maxRuns = 100 *# Numero massimo di iterazioni*  
)  
  
  
*# Variabili confermate, eliminate e rilevanti*  
final\_vars <- getSelectedAttributes(boruta\_output, withTentative = TRUE)  
  
*# Visualizzazione grafica*  
**plot**(boruta\_output, las = 2, main = "Boruta Variable Importance")



*# Creazione di un dataset ridotto basato sulle variabili selezionate*  
MODEL\_SELECTION <- file1[, c(final\_vars, "Diabetes\_binary")]  
*#creazione partizione*  
train\_indices\_MS <- createDataPartition(Dati\_Model\_Selection$Diabetes\_binary, p = 0.7, list = FALSE) *# Training set*  
dati\_training\_MS <- Dati\_Model\_Selection[train\_indices\_MS, ]

## Regressione Logistica

*# dati\_training\_MS$Age <- relevel(dati\_training\_MS$Age13)*

set.seed(1234)Control=trainControl(method= "cv",number=5,classProbs = TRUE,summaryFunction=twoClassSummary)glmPP=train(Diabetes\_binary~.,data=dati\_training\_MS , method = "glm", preProcess=c( "corr", "nzv","BoxCox"),trControl = Control, tuneLength=5, trace=FALSE,na.action = na.pass,metric='Sens')**confusionMatrix**(glmPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 1.9 1.7  
## c0 12.0 84.4  
##   
## Accuracy (average) : 0.8627

*#summary(glmPP)*

R=1-(91884/114716) *# pseudo R*  
**print(R)**

## 0.1990306

*#OVERFITTING*  
**library**(pROC)

*# Previsioni sui dati di training*  
train\_PRED\_logist <- predict(glmPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_logist <- train\_PRED\_logist[, "c1"] roc\_train\_logist <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_logist)  
*# Previsioni sui dati di validation*  
validation\_PRED\_logist <- predict(glmPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_logist <- validation\_PRED\_logist[, "c1"] roc\_validation\_logist <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_logist)  
auc\_train\_logist <- **auc**(roc\_train\_logist)  
auc\_validation\_logist <- **auc**(roc\_validation\_logist)  
*#print(roc\_train\_logist)*  
*#print(roc\_validation\_logist)*  
**print**(auc\_train\_logist)

## Area under the curve: 0.816

**print**(auc\_validation\_logist)

## Area under the curve: 0.816

overfitting\_logist=(auc\_train\_logist**-**auc\_validation\_logist)**/**auc\_train\_logist  
**print**(overfitting\_logist)

## [1] -0.000333

*# NON OVERFITTTANO*

## Lasso

*# dati\_training$Age <- relevel(dati\_training\_$Age13)*

set.seed(1234)grid = expand.grid(.alpha=1,.lambda=seq(0, 1, by = 0.01))Control=trainControl(method= "cv",number=5, classProbs=TRUE,summaryFunction=twoClassSummary)lassoPP=train(Diabetes\_binary~.,data=dati\_training , method = "glmnet", family ="binomial",trControl = Control, tuneLength=5,preProcess=c("scale","nzv") ,tuneGrid=grid,metric='Sens')  
**confusionMatrix**(lassoPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 2.1 1.7  
## c0 11.8 84.4  
##   
## Accuracy (average) : 0.8648

*#OVERFITTING Lasso*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_lasso <- predict(lassoPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_lasso <- train\_PRED\_lasso[, "c1"] roc\_train\_lasso <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_lasso)  
*# Previsioni sui dati di validation*  
validation\_PRED\_lasso <- predict(lassoPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_lasso <- validation\_PRED\_lasso[, "c1"] roc\_validation\_lasso <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_lasso) *#print(roc\_train\_lasso)*  
*#print(roc\_validation\_lasso)*  
auc\_train\_lasso <- **auc**(roc\_train\_lasso)  
auc\_validation\_lasso <- **auc**(roc\_validation\_lasso)  
**print**(auc\_train\_lasso)

## Area under the curve: 0.823

**print**(auc\_validation\_lasso)

## Area under the curve: 0.823

overfitting\_lasso=(auc\_train\_lasso**-**auc\_validation\_lasso)**/**auc\_train\_lasso  
**print**(overfitting\_lasso)

## [1] 0.0000747

*# NON OVERFITTTANO*

## PLS

set.seed(1234)preProc <- c("center", "scale")Control=trainControl(method= "none", classProbs=TRUE,summaryFunction=twoClassSummary)tuneGrid = expand.grid(.ncomp = 3)plsPP=train(Diabetes\_binary~.,data=dati\_training\_MS , method = "pls", preProcess = preProc,trControl = Control, tuneGrid=tuneGrid,metric='Sens')**confusionMatrix**(plsPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 0.7 0.5  
## c0 13.2 85.6  
##   
## Accuracy (average) : 0.8631

*#OVERFITTING PLS*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_PLS <- predict(plsPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_PLS <- train\_PRED\_PLS[, "c1"] roc\_train\_PLS <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_PLS) *# Previsioni sui dati di validation*  
validation\_PRED\_PLS <- predict(plsPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_PLS <- validation\_PRED\_PLS[, "c1"] roc\_validation\_PLS <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_PLS) *#print(roc\_train\_PLS)*  
*#print(roc\_validation\_PLS)*  
auc\_train\_PLS <- **auc**(roc\_train\_PLS)  
auc\_validation\_PLS <- **auc**(roc\_validation\_PLS)  
**print**(auc\_train\_PLS)

## Area under the curve: 0.816

**print**(auc\_validation\_PLS)

## Area under the curve: 0.817

overfitting\_PLS=(auc\_train\_PLS**-**auc\_validation\_PLS)**/**auc\_train\_PLS  
**print**(overfitting\_PLS)

## [1] -0.00116

*# NON OVERFITTTANO*

## Naive Bayes

set.seed(1234)preProc <- c("nzv","corr")Control=trainControl(method= "cv",number=5, classProbs=TRUE,summaryFunction=twoClassSummary)Naive\_BayesPP=train(Diabetes\_binary~.,data=dati\_training , method = "nb", family ="binomial",trControl = Control, tuneLength=5,na.action = na.exclude,metric='Sens')**confusionMatrix**(Naive\_BayesPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 9.1 21.6  
## c0 4.9 64.4  
##   
## Accuracy (average) : 0.7351

*#OVERFITTING NAIVE BAYES*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_Naive <- predict(Naive\_BayesPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_Naive <- train\_PRED\_Naive[, "c1"] roc\_train\_Naive <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_Naive) *# Previsioni sui dati di validation*  
validation\_PRED\_Naive <- predict(Naive\_BayesPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_Naive <- validation\_PRED\_Naive[, "c1"] roc\_validation\_Naive <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_Naive) *#print(roc\_train\_Naive)*  
*#print(roc\_validation\_Naive)*  
auc\_train\_Naive <- **auc**(roc\_train\_Naive)  
auc\_validation\_Naive <- **auc**(roc\_validation\_Naive)  
**print**(auc\_train\_Naive)

## Area under the curve: 0.768

**print**(auc\_validation\_Naive)

## Area under the curve: 0.768

overfitting\_Naive=(auc\_train\_Naive**-**auc\_validation\_Naive)**/**auc\_train\_Naive  
**print**(overfitting\_Naive)

## [1] -0.000685

*# NON OVERFITTA*

## K Nearest Neighbour

set.seed(1234)grid = expand.grid(.k=seq(5,20, by=3))preProc <- c( "scale","corr","nzv")Control=trainControl(method= "cv",number=5,search="grid", classProbs=TRUE,summaryFunction=twoClassSummary)knn\_modelPP=train(Diabetes\_binary~.,data=dati\_training\_MS , method = "knn",preProcess = preProc,trControl = Control, tuneLength=5, tuneGrid=grid, na.action = na.exclude,metric='Sens') **confusionMatrix**(knn\_modelPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 4.0 9.7  
## c0 10.0 76.4  
##   
## Accuracy (average) : 0.8035

*#OVERFITTING knn*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_KNN <- predict(knn\_modelPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_KNN <- train\_PRED\_KNN[, "c1"] roc\_train\_KNN <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_KNN) *# Previsioni sui dati di validation*  
validation\_PRED\_KNN <- predict(knn\_modelPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_KNN <- validation\_PRED\_KNN[, "c1"] roc\_validation\_KNN <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_KNN) *#print(roc\_train\_KNN)*  
*#print(roc\_validation\_KNN)*  
auc\_train\_KNN <- **auc**(roc\_train\_KNN)  
auc\_validation\_KNN <- **auc**(roc\_validation\_KNN)  
**print**(auc\_train\_KNN)

## Area under the curve: 0.82

**print**(auc\_validation\_KNN)

## Area under the curve: 0.817

overfitting\_Knn=(auc\_train\_KNN**-**auc\_validation\_KNN)**/**auc\_train\_KNN  
**print**(overfitting\_Knn)

## [1] 0.00249

*#NON OVERFITTA*

## Gradient Boosting

set.seed(123)control<- trainControl( method="cv", number=5,search="grid",summaryFunction=twoClassSummary,classProbs = TRUE)gradien\_boostingPP <- train(Diabetes\_binary~ ., data=dati\_training, method = "gbm", trControl = control)  
**confusionMatrix**(gradien\_boostingPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 2.5 2.0  
## c0 11.4 84.1  
##   
## Accuracy (average) : 0.8659

*#OVERFITTING Gradient Boosting*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_GradBoost <- predict(gradien\_boostingPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_GradBoost <- train\_PRED\_GradBoost[, "c1"] roc\_train\_GradBoost <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_GradBoost) *# Previsioni sui dati di validation*  
validation\_PRED\_GradBoost <- predict(gradien\_boostingPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_GradBoost <- validation\_PRED\_GradBoost[, "c1"] roc\_validation\_GradBoost <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_GradBoost) *#print(roc\_train\_GradBoost)*  
*#print(roc\_validation\_GradBoost)*  
auc\_train\_GradBoost <- **auc**(roc\_train\_GradBoost)  
auc\_validation\_GradBoost <- **auc**(roc\_validation\_GradBoost)  
**print**(auc\_train\_GradBoost)

## Area under the curve: 0.826

**print**(auc\_validation\_GradBoost)

## Area under the curve: 0.826

overfitting\_gradientBoost=(auc\_train\_GradBoost**-**auc\_validation\_GradBoost)**/**auc\_train\_GradBoost  
**print**(overfitting\_gradientBoost)

## [1] -0.000642

*# NON OVERFITTA*

## Albero Decisionale

**library**(rpart)  
**library**(rpart.plot)

control <- trainControl(method="cv", number=5, search="grid",summaryFunction = twoClassSummary, classProbs = TRUE)alberoPP <- train(Diabetes\_binary ~ ., data = dati\_training, method = "rpart", trControl=control,metric='Sens',tuneLength = 10  *# Ricerca automatica dei migliori iperparametri*  
)  
**confusionMatrix**(alberoPP)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 1.9 1.5  
## c0 12.1 84.6  
##   
## Accuracy (average) : 0.8644

*#OVERFITTING Albero*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_Tree <- predict(alberoPP, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_Tree <- train\_PRED\_Tree[, "c1"] roc\_train\_Tree <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_Tree)  
*# Previsioni sui dati di validation*  
validation\_PRED\_Tree <- predict(alberoPP, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_Tree <- validation\_PRED\_Tree[, "c1"] roc\_validation\_Tree <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_Tree) *#print(roc\_train\_Tree)*  
*#print(roc\_validation\_Tree)*  
auc\_train\_Tree <- **auc**(roc\_train\_Tree)  
auc\_validation\_Tree <- **auc**(roc\_validation\_Tree)  
**print**(auc\_train\_Tree)

## Area under the curve: 0.721

**print**(auc\_validation\_Tree)

## Area under the curve: 0.725

overfitting\_albero=(auc\_train\_Tree**-**auc\_validation\_Tree)**/**auc\_train\_Tree  
**print**(overfitting\_albero)

## [1] -0.0061

*# NON OVERFITTA*  
  
Vimportance <- **varImp**(alberoPP)  
**plot**(Vimportance) *# plot importanza variabili*

library(rpart)

library(rpart.plot)

library(caret)

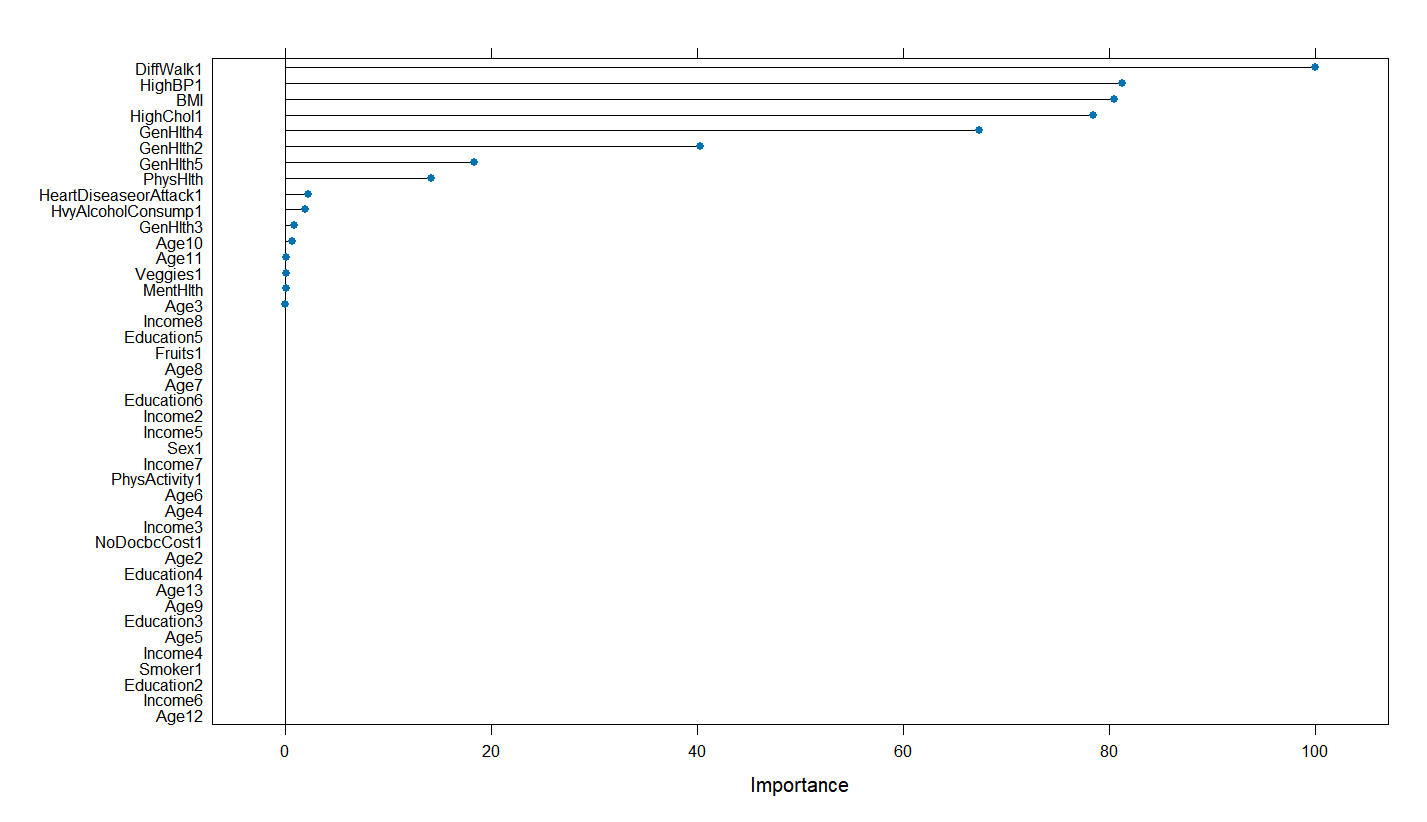
*#print(alberoPP)*

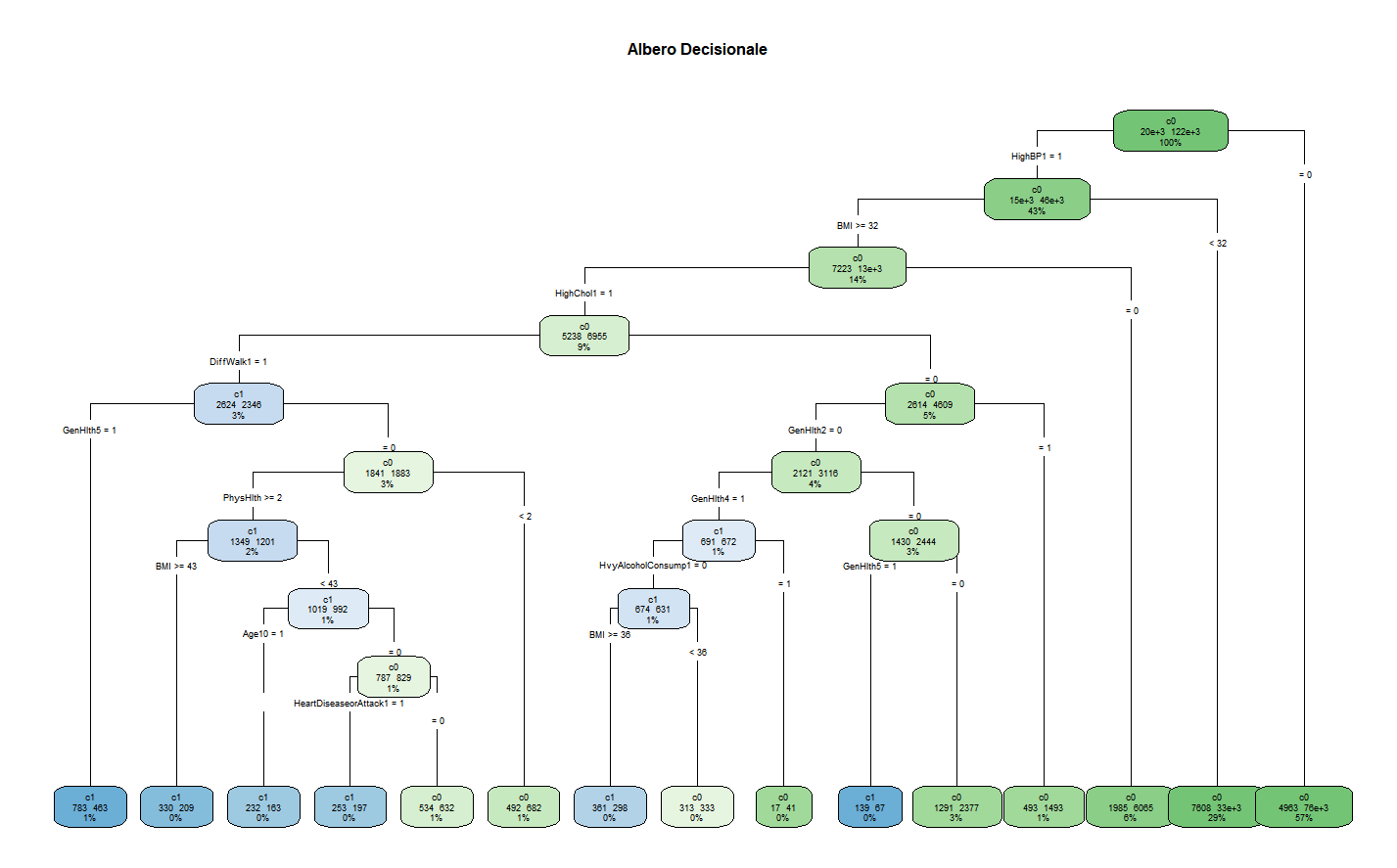
albero\_rpart <- alberoPP$finalModel

*#print(albero\_rpart)*

*# Visualizzare l'albero con rpart.plot*

rpart.plot(albero\_rpart, type = 4, extra = 101,split.font = 0.9, ycompress=FALSE, cex=.45, main = "Albero Decisionale")





## Random Forest

set.seed(1234)metric <- "Sens"control <- trainControl(method="cv", number=5, search="grid",summaryFunction = twoClassSummary, classProbs #= TRUE)tunegrid <- expand.grid(.mtry=c(4:10)) *# numereo covariate random per ogni albero uso radice di n =6,5*

FORESTPP2 <- train(Diabetes\_binary~., data=dati\_training, method="rf", metric=metric, tuneGrid=tunegrid,trControl=control, ntree=250)**confusionMatrix**(FORESTPP2)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 2.4 2.2  
## c0 11.5 83.9  
##   
## Accuracy (average) : 0.8631

*#OVERFITTING Random Forest*  
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_RandomF <- predict(FORESTPP2, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_RandomF <- train\_PRED\_RandomF[, "c1"] roc\_train\_RandomF <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_RandomF) *# Previsioni sui dati di validation*  
validation\_PRED\_RandomF <- predict(FORESTPP2, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_RandomF <- validation\_PRED\_RandomF[, "c1"] roc\_validation\_RandomF <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_RandomF) *#print(roc\_train\_RandomF)*  
*#print(roc\_validation\_RandomF)*  
auc\_train\_RandomF <- **auc**(roc\_train\_RandomF)  
auc\_validation\_RandomF <- **auc**(roc\_validation\_RandomF)  
**print**(auc\_train\_RandomF)

## Area under the curve: 0.994

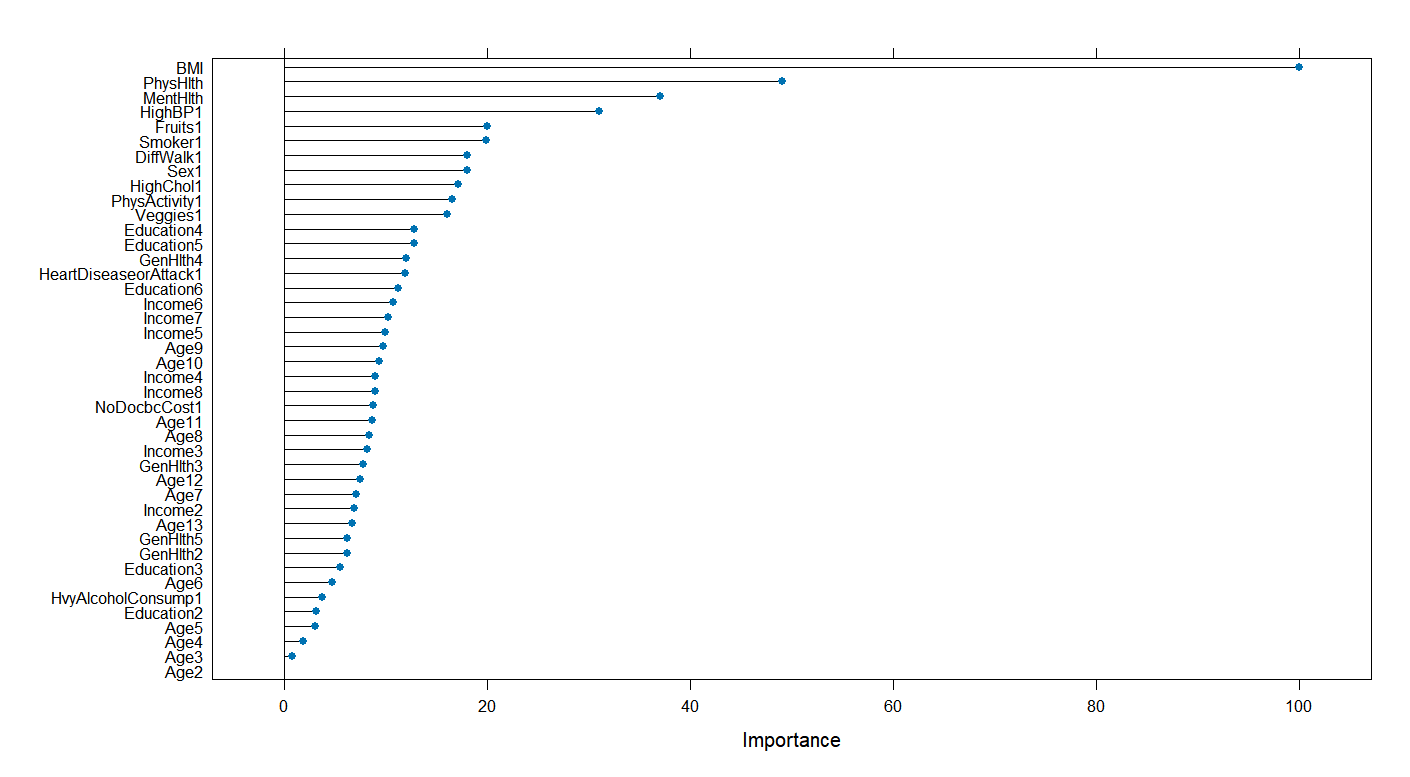
**print**(auc\_validation\_RandomF)

## Area under the curve: 0.801

overfitting\_RandomForest=(auc\_train\_RandomF**-**auc\_validation\_RandomF)**/**auc\_train\_RandomF  
**print**(overfitting\_RandomForest)

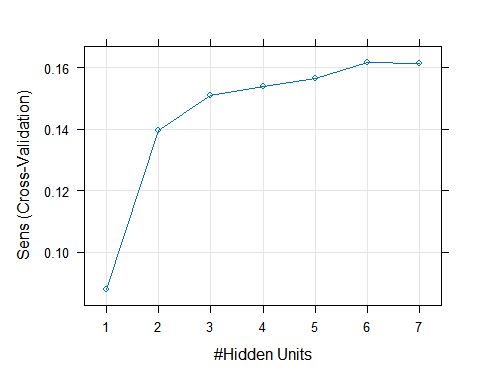
## [1] 0.195

*#OVERFITTING*  
  
Vimportance <- **varImp**(FORESTPP2)  
**plot**(Vimportance) *# plot importanza variabili*



## Reti Neurali

Grid0 = expand.grid(.size =seq(1,7, by=1), .decay = 0.1 )set.seed(1234) control = trainControl(method="cv", number=5, search = "grid",summaryFunction = twoClassSummary,classProbs = TRUE)nnetPP2 <- train(dati\_training\_MS[-1], dati\_training\_MS$Diabetes\_binary,method = "nnet",tuneGrid=Grid0,preProcess=c("scale","corr","nzv"), metric="Sens", trControl=control,trace = TRUE, # uso per vedere se converge se non converge provare ad aumentare numero max iterazionimaxit = 300) # numero massimo di iterazioniprint(nnetPP2)  
**plot**(nnetPP2) *#vedo andamento neuroni nascosti e metrica dopo il 6 inizia a decrescere*



| Neural Network  142062 samples  18 predictor  2 classes: 'c1', 'c0'  Pre-processing: scaled (3), ignore (15)  Resampling: Cross-Validated (5 fold)  Summary of sample sizes: 113650, 113650, 113649, 113650, 113649  Resampling results across tuning parameters:  size ROC Sens Spec  1 0.7593056 0.0878561 0.9889015  2 0.8262125 0.1395880 0.9829718  3 0.8273610 0.1508538 0.9819331  4 0.8273345 0.1537841 0.9812707  5 0.8272559 0.1563605 0.9808535  6 0.8273921 0.1616652 0.9802647  7 0.8266831 0.1612611 0.9798394  Tuning parameter 'decay' was held constant at a value of 0.1  Sens was used to select the optimal model using the largest value.  The final values used for the model were size = 6 and decay = 0.1. |
| --- |
|  |
| |  | | --- | |

**confusionMatrix**(nnetPP2)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction c1 c0  
## c1 2.3 1.7  
## c0 11.7 84.4  
##   
## Accuracy (average) : 0.8662

*#OVERFITTING Reti neurali*   
**library**(pROC)  
*# Previsioni sui dati di training*  
train\_PRED\_nnet <- predict(nnetPP2, newdata = dati\_training, type = "prob")train\_PRED\_prob\_c1\_nnet <- train\_PRED\_nnet[, "c1"] roc\_train\_nnet <- roc(dati\_training$Diabetes\_binary, train\_PRED\_prob\_c1\_nnet) *# Previsioni sui dati di validation*  
validation\_PRED\_nnet <- predict(nnetPP2, newdata = dati\_validation, type = "prob")validation\_PRED\_prob\_c1\_nnet <- validation\_PRED\_nnet[, "c1"] roc\_validation\_nnet <- roc(dati\_validation$Diabetes\_binary, validation\_PRED\_prob\_c1\_nnet) *#print(roc\_train\_nnet)*  
*#print(roc\_validation\_nnet)*  
auc\_train\_nnet <- **auc**(roc\_train\_nnet)  
auc\_validation\_nnet <- **auc**(roc\_validation\_nnet)  
**print**(auc\_train\_nnet)

## Area under the curve: 0.83

**print**(auc\_validation\_nnet)

## Area under the curve: 0.831

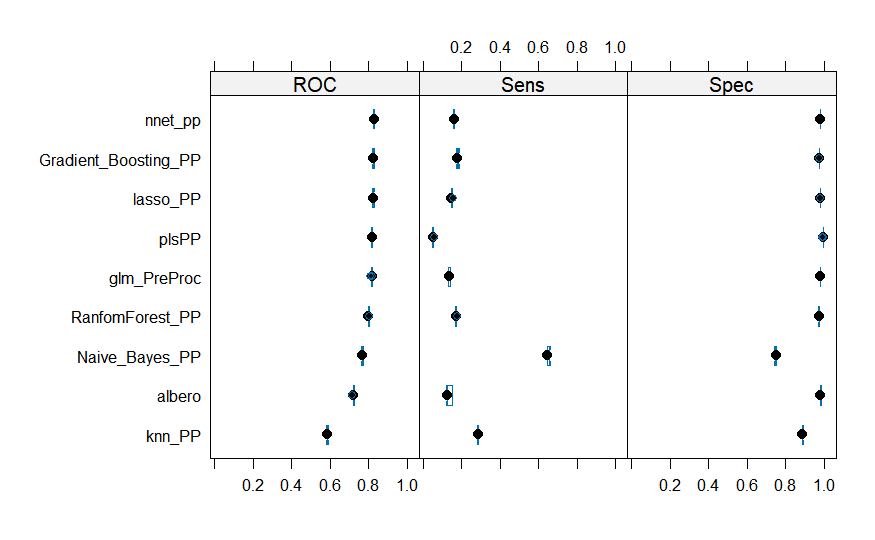
overfitting\_neuralnet=(auc\_train\_nnet**-**auc\_validation\_nnet)**/**auc\_train\_nnet  
**print**(overfitting\_neuralnet)

## [1] -0.0014

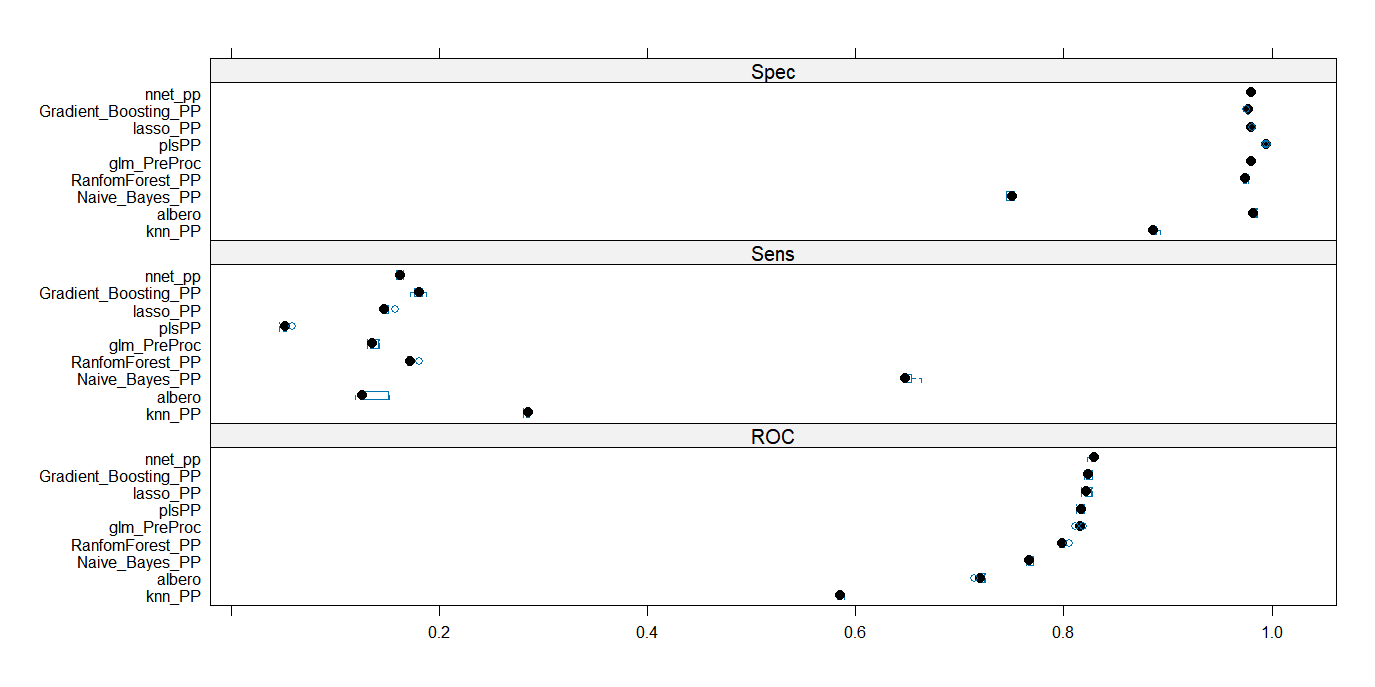
*# NON OVERFITTA*

## Confronto Box Plot

results <- **resamples**(**list**(albero=alberoPP,glm\_PreProc=glmPP,plsPP=plsPP,lasso\_PP=lassoPP,Naive\_Bayes\_PP=Naive\_BayesPP,knn\_PP=knn\_modelPP,nnet\_pp=nnetPP2,RanfomForest\_PP=FORESTPP2,Gradient\_Boosting\_PP=gradien\_boostingPP))   
**bwplot**(results)



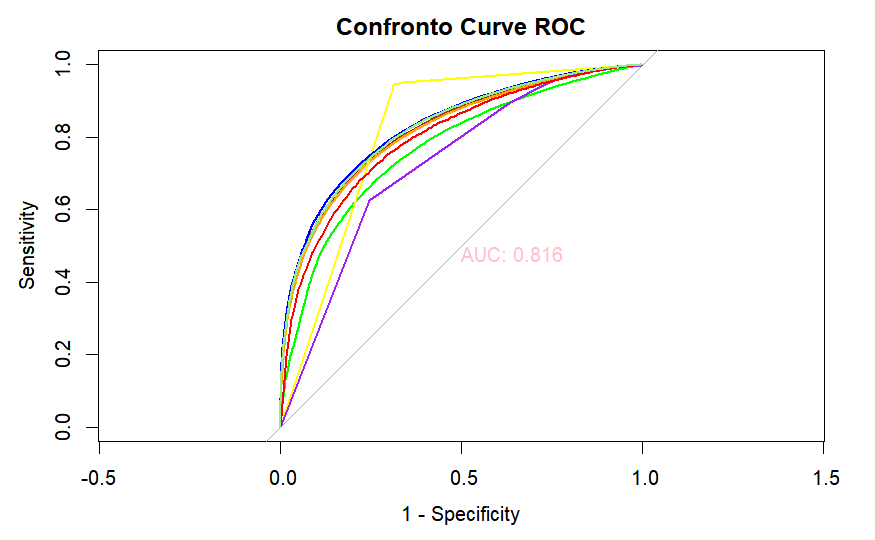
**bwplot**(results,layout=**c**(1,3))



## Confronto Curve Roc

**library**(funModeling)

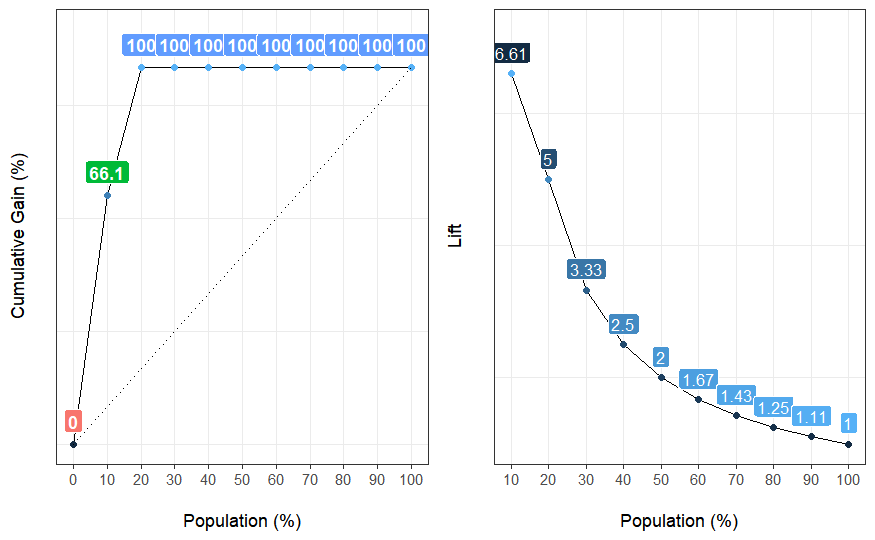
**plot**(roc\_validation\_logist, col="pink", lwd=2, main="Confronto Curve ROC", print.auc=TRUE, legacy.axes=TRUE)  
**lines**(roc\_validation\_lasso, col="brown", lwd=2)  
**lines**(roc\_validation\_PLS, col="orange", lwd=2)  
**lines**(roc\_validation\_Naive, col="green", lwd=2)  
**lines**(roc\_validation\_KNN, col="yellow", lwd=2)  
**lines**(roc\_validation\_Tree, col="purple", lwd=2)  
**lines**(roc\_validation\_RandomF, col="red", lwd=2)  
**lines**(roc\_validation\_nnet, col="blue", lwd=2)  
**lines**(roc\_validation\_GradBoost, col="lightgreen", lwd=2)



## Curve Lift

*#KNN*

p\_Nayve<- **predict**(knn\_modelPP, newdata = dati\_validation, type = "prob")  
posterior\_Naive\_c1 <- p\_Nayve[, "c1"]   
  
test**$**posterior\_Naive=posterior\_Naive\_c1  
test=**na.omit**(test)  
*#head(test)*  
**library**(funModeling)  
**gain\_lift**(data = test, score = 'posterior\_Naive', target = 'Diabetes\_binary')



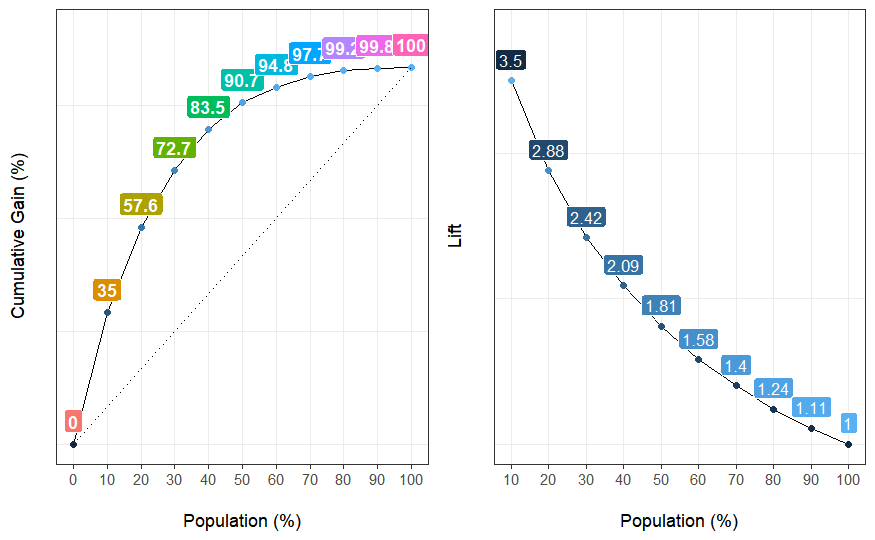
## Population Gain Lift Score.Point  
## 1 10 66.1 6.61 1  
## 2 20 100.0 5.00 0  
## 3 30 100.0 3.33 0  
## 4 40 100.0 2.50 0  
## 5 50 100.0 2.00 0  
## 6 60 100.0 1.67 0  
## 7 70 100.0 1.43 0  
## 8 80 100.0 1.25 0  
## 9 90 100.0 1.11 0  
## 10 100 100.0 1.00 0

validation\_PRED\_KNN <- **predict**(knn\_modelPP, newdata = dati\_validation, type = "raw")  
conf\_matrix\_KNN <- **confusionMatrix**(validation\_PRED\_KNN, dati\_validation**$**Diabetes\_binary)  
**print**(conf\_matrix\_KNN)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 5700 2665  
## c0 2783 49735  
##   
## Accuracy : 0.911   
## 95% CI : (0.908, 0.913)   
## No Information Rate : 0.861   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.625   
##   
## Mcnemar's Test P-Value : 0.113   
##   
## Sensitivity : 0.6719   
## Specificity : 0.9491   
## Pos Pred Value : 0.6814   
## Neg Pred Value : 0.9470   
## Prevalence : 0.1393   
## Detection Rate : 0.0936   
## Detection Prevalence : 0.1374   
## Balanced Accuracy : 0.8105   
##   
## 'Positive' Class : c1   
##

*# RETE NEURALE*

p\_nnet <- **predict**(nnetPP2, newdata = dati\_validation, type = "prob")  
posterior\_nnet\_c1 <- p\_nnet[, "c1"]   
  
test**$**posterior\_nnet=posterior\_nnet\_c1  
test=**na.omit**(test)  
*#head(test)*  
**library**(funModeling)  
**gain\_lift**(data = test, score = 'posterior\_nnet', target = 'Diabetes\_binary')



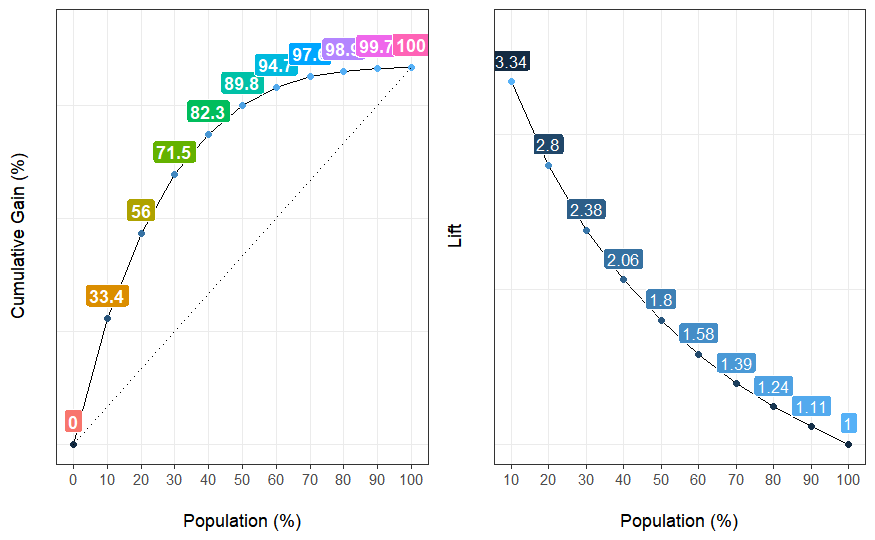
## Population Gain Lift Score.Point  
## 1 10 35.0 3.50 0.374653  
## 2 20 57.6 2.88 0.251773  
## 3 30 72.7 2.42 0.175690  
## 4 40 83.5 2.09 0.120718  
## 5 50 90.7 1.81 0.078833  
## 6 60 94.8 1.58 0.048780  
## 7 70 97.7 1.40 0.027307  
## 8 80 99.2 1.24 0.013967  
## 9 90 99.8 1.11 0.005748  
## 10 100 100.0 1.00 0.000366

validation\_PRED\_nnet <- **predict**(nnetPP2, newdata = dati\_validation, type = "raw")  
conf\_matrix\_nnet <- **confusionMatrix**(validation\_PRED\_nnet, dati\_validation**$**Diabetes\_binary)  
**print**(conf\_matrix\_nnet)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 1392 957  
## c0 7091 51443  
##   
## Accuracy : 0.868   
## 95% CI : (0.865, 0.87)   
## No Information Rate : 0.861   
## P-Value [Acc > NIR] : 0.000000152   
##   
## Kappa : 0.209   
##   
## Mcnemar's Test P-Value : < 0.0000000000000002  
##   
## Sensitivity : 0.1641   
## Specificity : 0.9817   
## Pos Pred Value : 0.5926   
## Neg Pred Value : 0.8789   
## Prevalence : 0.1393   
## Detection Rate : 0.0229   
## Detection Prevalence : 0.0386   
## Balanced Accuracy : 0.5729   
##   
## 'Positive' Class : c1   
##

*#LASSO*

p\_Lasso <- **predict**(lassoPP, newdata = dati\_validation, type = "prob")  
posterior\_Lasso\_c1 <- p\_Lasso[, "c1"]   
  
test**$**posterior\_Lasso=posterior\_Lasso\_c1  
test=**na.omit**(test)  
*#head(test)*  
**library**(funModeling)  
**gain\_lift**(data = test, score = 'posterior\_Lasso', target = 'Diabetes\_binary')



## Population Gain Lift Score.Point  
## 1 10 33.4 3.34 0.363735  
## 2 20 56.0 2.80 0.241615  
## 3 30 71.5 2.38 0.165639  
## 4 40 82.3 2.06 0.114510  
## 5 50 89.8 1.80 0.079210  
## 6 60 94.7 1.58 0.053770  
## 7 70 97.5 1.39 0.035369  
## 8 80 98.9 1.24 0.022350  
## 9 90 99.7 1.11 0.012525  
## 10 100 100.0 1.00 0.000875

validation\_PRED\_Lasso <- **predict**(lassoPP, newdata = dati\_validation, type = "raw")  
conf\_matrix\_Lasso <- **confusionMatrix**(validation\_PRED\_Lasso, dati\_validation**$**Diabetes\_binary)  
**print**(conf\_matrix\_Lasso)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 1252 975  
## c0 7231 51425  
##   
## Accuracy : 0.865   
## 95% CI : (0.862, 0.868)   
## No Information Rate : 0.861   
## P-Value [Acc > NIR] : 0.000578   
##   
## Kappa : 0.187   
##   
## Mcnemar's Test P-Value : < 0.0000000000000002  
##   
## Sensitivity : 0.1476   
## Specificity : 0.9814   
## Pos Pred Value : 0.5622   
## Neg Pred Value : 0.8767   
## Prevalence : 0.1393   
## Detection Rate : 0.0206   
## Detection Prevalence : 0.0366   
## Balanced Accuracy : 0.5645   
##   
## 'Positive' Class : c1   
##

## 

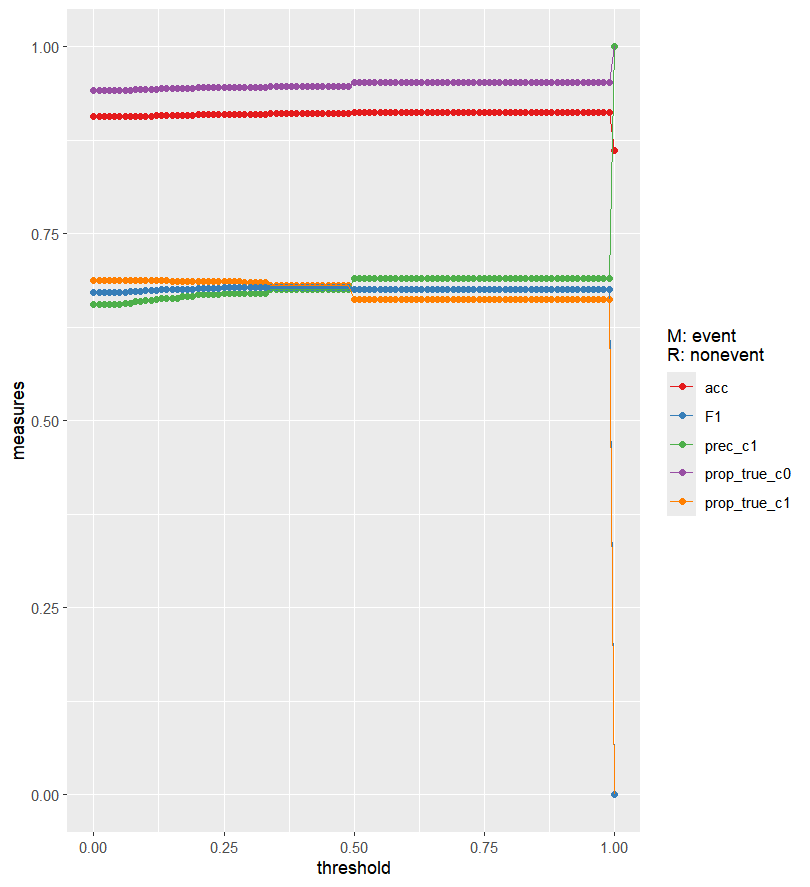
## Analisi Miglioramento Soglia KNN

STEP4\_PRED\_Knn <- **predict**(knn\_modelPP, newdata = dati\_validation, type = "prob")  
*#head(STEP4\_PRED\_Knn)*  
df=**data.frame**(**cbind**(dati\_validation**$**Diabetes\_binary , STEP4\_PRED\_Knn))  
*#head(df)*  
**colnames**(df)=**c**("Class","Probc1","Probc0")  *probabilita di essere c0 o c1 utilizzando le predict*  
*#head(df)#*   
  
*# seleziono prime due colonne target osservato e probabilita evento di interesse*  
df=df[,1**:**2]  
*#head(df)*   
  
**library**(dplyr)  
  
thresholds <- **seq**(from = 0, to = 1, by = 0.01)  
prop\_table <- **data.frame**(threshold = thresholds, prop\_true\_c1 = NA, prop\_true\_c0 = NA, true\_c1 = NA, true\_c0 = NA ,fn\_c1=NA)  
  
**for** (threshold **in** thresholds) {  
 pred <- **ifelse**(df**$**Probc1 **>** threshold, "c1", "c0")   
 pred\_t <- **ifelse**(pred **==** df**$**Class, TRUE, FALSE)  
   
 group <- **data.frame**(df, "pred" = pred\_t) **%>%**  
 **group\_by**(Class, pred) **%>%**  
 dplyr**::summarise**(n = **n**())  
   
 group\_c1 <- **filter**(group, Class **==** "c1")  
   
 true\_c1=**sum**(**filter**(group\_c1, pred **==** TRUE)**$**n)  
 prop\_c1 <- **sum**(**filter**(group\_c1, pred **==** TRUE)**$**n) **/** **sum**(group\_c1**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c1"] <- prop\_c1  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c1"] <- true\_c1  
   
 fn\_c1=**sum**(**filter**(group\_c1, pred **==** FALSE)**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "fn\_c1"] <- fn\_c1  
   
   
 group\_c0 <- **filter**(group, Class **==** "c0")  
   
 true\_c0=**sum**(**filter**(group\_c0, pred **==** TRUE)**$**n)  
 prop\_c0 <- **sum**(**filter**(group\_c0, pred **==** TRUE)**$**n) **/** **sum**(group\_c0**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c0"] <- prop\_c0  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c0"] <- true\_c0  
   
}

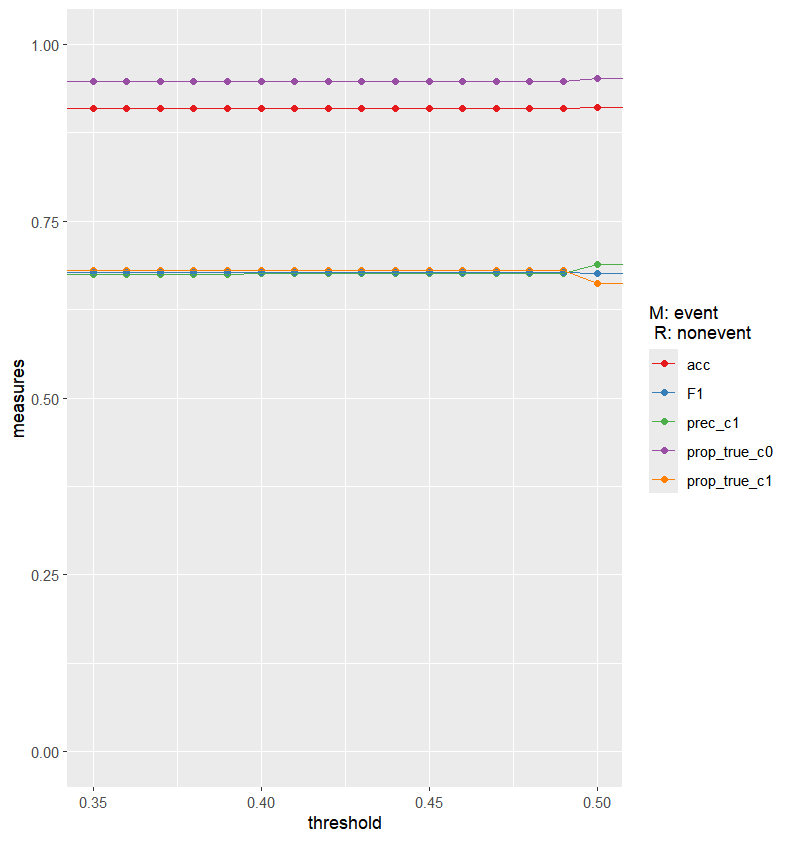
*#head(prop\_table, n=10)*  
  
  
prop\_table**$**n=**nrow**(dati\_validation)  
prop\_table**$**fp\_c1=**nrow**(dati\_validation)**-**prop\_table**$**true\_c0**-**prop\_table**$**true\_c1**-**prop\_table**$**fn\_c1 *# falsi positivi*  
prop\_table**$**acc=(prop\_table**$**true\_c0**+**prop\_table**$**true\_c1)**/nrow**(dati\_validation) *# accuracy*  
*# precision= veri c1 sul totale previsti come c1*  
prop\_table**$**prec\_c1=prop\_table**$**true\_c1**/**(prop\_table**$**true\_c1**+**prop\_table**$**fp\_c1)  
prop\_table**$**F1=2**\***(prop\_table**$**prop\_true\_c1**\***prop\_table**$**prec\_c1)**/**(prop\_table**$**prop\_true\_c1**+**prop\_table**$**prec\_c1)  
  
  
  
*#tail(prop\_table)*  
*# inputazione dati per non avere NaN zero denominatore manda valori a infinito*  
**library**(Hmisc)  
prop\_table**$**prec\_c1=**impute**(prop\_table**$**prec\_c1, 1)  
prop\_table**$**F1=**impute**(prop\_table**$**F1, 0)  
*#tail(prop\_table)*  
*# elimino colonne conteggi che non mi servono*   
prop\_table2 = prop\_table[,**-c**(4**:**8)]   
  
**library**(dplyr)  
**library**(tidyr)  
*#head(prop\_table2)*  
*# impilo metriche*   
gathered=prop\_table2 **%>%**  
 **gather**(x, y, prop\_true\_c1**:**F1)

## Warning: attributes are not identical across measure variables; they will be  
## dropped

*#head(gathered)*  
*#faccio plot metriche*   
**library**(ggplot2)  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n**R: nonevent")



*# plot mi da informazioni su soglia da scegliere*   
*# precision metrica da prediligire ovvero percentUale di c1 veri sul totale di previsti c1*   
  
*# zoom*  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n** R: nonevent") **+**  
 **coord\_cartesian**(xlim = **c**(0.35, 0.50))



*# applico soglia sul dataset validation*   
df**$**decision=**ifelse**(df**$**Probc1**>**0.40,"c1","c0")  
  
*# matrice confusione*  
**table**(df**$**Class,df**$**decision)

##   
## c0 c1  
## c1 2710 5573  
## c0 49631 2769

**confusionMatrix**(**as.factor**(df**$**decision),df**$**Class, positive = "c1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 5573 2769  
## c0 2710 49631  
##   
## Accuracy : 0.91   
## 95% CI : (0.9077, 0.9123)   
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : <2e-16  
##   
## Kappa : 0.6259   
##   
## Mcnemar's Test P-Value : 0.4333   
##   
## Sensitivity : 0.68054   
## Specificity : 0.94716   
## Pos Pred Value : 0.67584   
## Neg Pred Value : 0.94822   
## Prevalence : 0.13933  
## Detection Rate : 0.09482   
## Detection Prevalence : 0.14030   
## Balanced Accuracy : 0.81385   
##   
## 'Positive' Class : c1   
##

*#head(df,n=20)*

## Analisi Miglioramento Soglia reti neurali

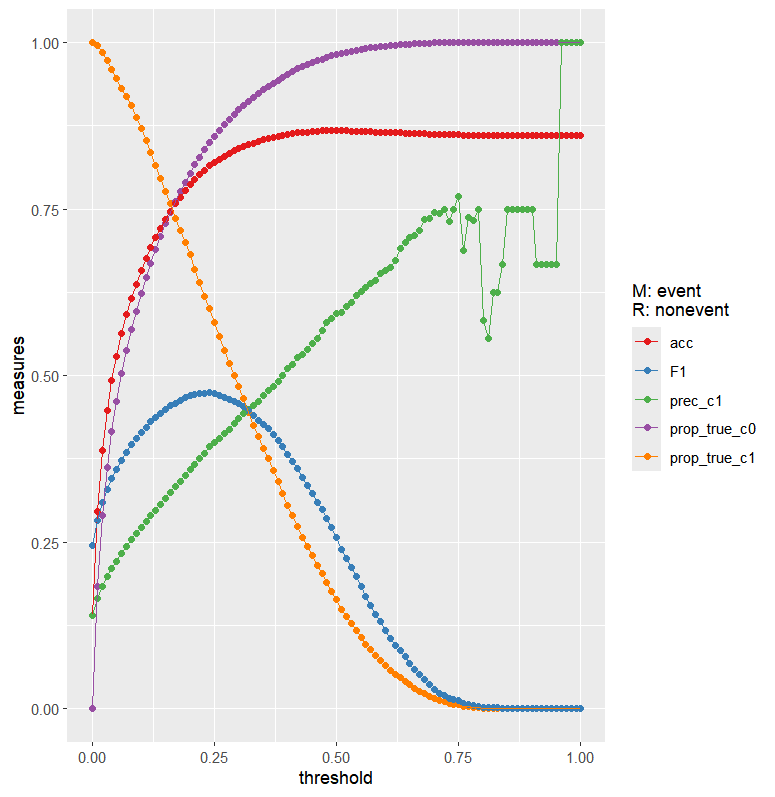
STEP4\_PRED\_nnet <- **predict**(nnetPP2, newdata = dati\_validation, type = "prob")  
*#head(STEP4\_PRED\_nnet)*  
df=**data.frame**(**cbind**(dati\_validation**$**Diabetes\_binary , STEP4\_PRED\_nnet))  
*#head(df)*  
**colnames**(df)=**c**("Class","Probc1","Probc0")

*#head(df)*  
  
  
df=df[,1:2]  
*#head(df)*   
  
**library**(dplyr)  
  
thresholds <- **seq**(from = 0, to = 1, by = 0.01)  
prop\_table <- **data.frame**(threshold = thresholds, prop\_true\_c1 = NA, prop\_true\_c0 = NA, true\_c1 = NA, true\_c0 = NA ,fn\_c1=NA)  
  
**for** (threshold **in** thresholds) {  
 pred <- **ifelse**(df**$**Probc1 **>** threshold, "c1", "c0")   
 pred\_t <- **ifelse**(pred **==** df**$**Class, TRUE, FALSE)  
   
 group <- **data.frame**(df, "pred" = pred\_t) **%>%**  
 **group\_by**(Class, pred) **%>%**  
 dplyr**::summarise**(n = **n**())  
   
 group\_c1 <- **filter**(group, Class **==** "c1")  
   
 true\_c1=**sum**(**filter**(group\_c1, pred **==** TRUE)**$**n)  
 prop\_c1 <- **sum**(**filter**(group\_c1, pred **==** TRUE)**$**n) **/** **sum**(group\_c1**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c1"] <- prop\_c1  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c1"] <- true\_c1  
   
 fn\_c1=**sum**(**filter**(group\_c1, pred **==** FALSE)**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "fn\_c1"] <- fn\_c1  
   
   
 group\_c0 <- **filter**(group, Class **==** "c0")  
   
 true\_c0=**sum**(**filter**(group\_c0, pred **==** TRUE)**$**n)  
 prop\_c0 <- **sum**(**filter**(group\_c0, pred **==** TRUE)**$**n) **/** **sum**(group\_c0**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c0"] <- prop\_c0  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c0"] <- true\_c0  
   
}

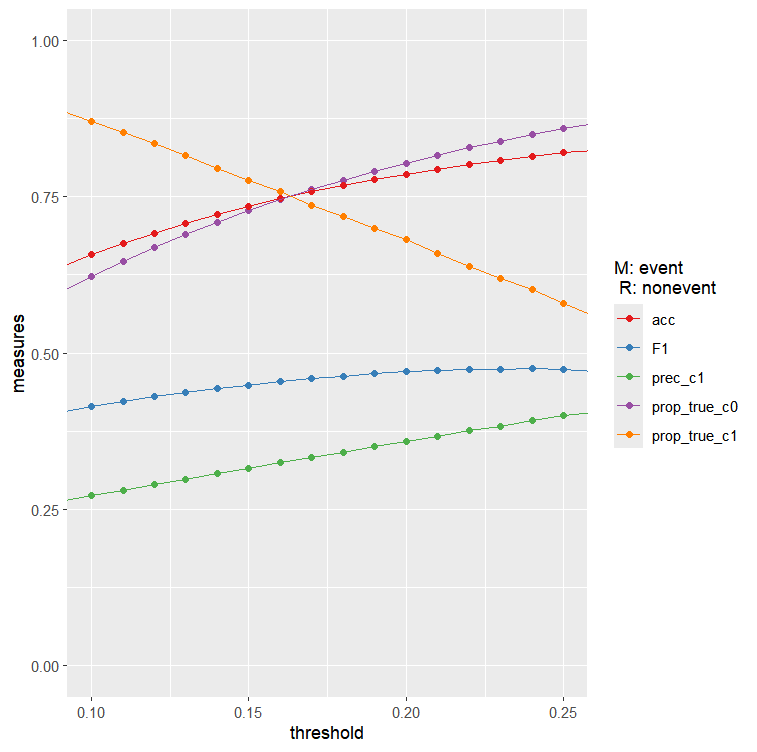
*#head(prop\_table, n=10)*  
  
  
prop\_table**$**n=**nrow**(dati\_validation)  
prop\_table**$**fp\_c1=**nrow**(dati\_validation)**-**prop\_table**$**true\_c0**-**prop\_table**$**true\_c1**-**prop\_table**$**fn\_c1 *# falsi positivi*  
prop\_table**$**acc=(prop\_table**$**true\_c0**+**prop\_table**$**true\_c1)**/nrow**(dati\_validation) *# accuracy*  
*# precision= veri c1 sul totale previsti come c1*  
prop\_table**$**prec\_c1=prop\_table**$**true\_c1**/**(prop\_table**$**true\_c1**+**prop\_table**$**fp\_c1)  
prop\_table**$**F1=2**\***(prop\_table**$**prop\_true\_c1**\***prop\_table**$**prec\_c1)**/**(prop\_table**$**prop\_true\_c1**+**prop\_table**$**prec\_c1)  
  
  
  
*#tail(prop\_table)*  
*# inputazione dati per non avere NaN zero denominatore manda valori a infinito*  
**library**(Hmisc)  
prop\_table**$**prec\_c1=**impute**(prop\_table**$**prec\_c1, 1)  
prop\_table**$**F1=**impute**(prop\_table**$**F1, 0)  
*#tail(prop\_table)*  
*# elimino colonne conteggi che non mi servono*   
prop\_table2 = prop\_table[,**-c**(4**:**8)]   
  
**library**(dplyr)  
**library**(tidyr)  
*#head(prop\_table2)*  
*# impilo metriche*   
gathered=prop\_table2 **%>%**  
 **gather**(x, y, prop\_true\_c1**:**F1)

## Warning: attributes are not identical across measure variables; they will be  
## dropped

*#head(gathered)*  
*#faccio plot metriche*   
**library**(ggplot2)  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n**R: nonevent")



*# zoom*  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n** R: nonevent") **+**  
 **coord\_cartesian**(xlim = **c**(0.10, 0.25))



*# applico soglia*

df**$**decision=**ifelse**(df**$**Probc1**>**0.15,"c1","c0")  
  
*# matrice confusione*  
**table**(df**$**Class,df**$**decision)

##   
## c0 c1  
## c1 1903 6580  
## c0 38126 14174

**confusionMatrix**(**as.factor**(df**$**decision),df**$**Class, positive = "c1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 6580 14274  
## c0 1903 38126  
##   
## Accuracy : 0.7343   
## 95% CI : (0.7308, 0.7378)   
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3124   
##   
## Mcnemar's Test P-Value : <2e-16  
##   
## Sensitivity : 0.7757   
## Specificity : 0.7276   
## Pos Pred Value : 0.3155   
## Neg Pred Value : 0.9525   
## Prevalence : 0.1393   
## Detection Rate : 0.1081   
## Detection Prevalence : 0.3425   
## Balanced Accuracy : 0.7516   
##   
## 'Positive' Class : c1   
##

*#table(dati\_validation$Diabetes\_binary)*  
  
*#head(df,n=20)*

## Analisi Miglioramento Soglia Lasso

STEP4\_PRED\_Lasso <- **predict**(lassoPP, newdata = dati\_validation, type = "prob")  
*#head(STEP4\_PRED\_Lasso)*  
df=**data.frame**(**cbind**(dati\_validation**$**Diabetes\_binary , STEP4\_PRED\_Lasso))  
*#head(df)*  
**colnames**(df)=**c**("Class","Probc1","Probc0")

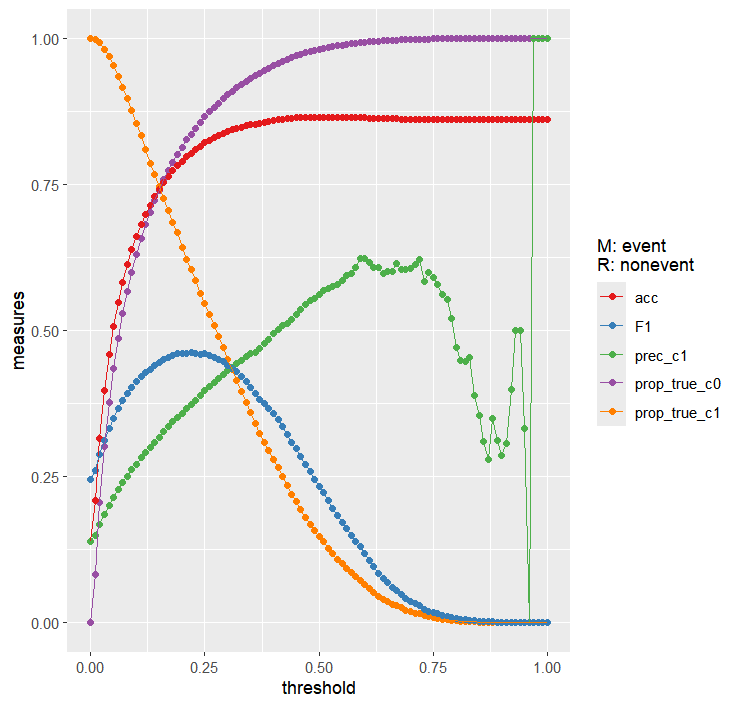
*#head(df)#*   
  
df=df[,1**:**2]  
*#head(df)*   
  
**library**(dplyr)  
  
thresholds <- **seq**(from = 0, to = 1, by = 0.01)  
prop\_table <- **data.frame**(threshold = thresholds, prop\_true\_c1 = NA, prop\_true\_c0 = NA, true\_c1 = NA, true\_c0 = NA ,fn\_c1=NA)  
  
**for** (threshold **in** thresholds) {  
 pred <- **ifelse**(df**$**Probc1 **>** threshold, "c1", "c0")

pred\_t <- **ifelse**(pred **==** df**$**Class, TRUE, FALSE)  
   
 group <- **data.frame**(df, "pred" = pred\_t) **%>%**  
 **group\_by**(Class, pred) **%>%**  
 dplyr**::summarise**(n = **n**())  
   
 group\_c1 <- **filter**(group, Class **==** "c1")  
   
 true\_c1=**sum**(**filter**(group\_c1, pred **==** TRUE)**$**n)  
 prop\_c1 <- **sum**(**filter**(group\_c1, pred **==** TRUE)**$**n) **/** **sum**(group\_c1**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c1"] <- prop\_c1  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c1"] <- true\_c1  
   
 fn\_c1=**sum**(**filter**(group\_c1, pred **==** FALSE)**$**n)  
  
 prop\_table[prop\_table**$**threshold **==** threshold, "fn\_c1"] <- fn\_c1  
   
   
 group\_c0 <- **filter**(group, Class **==** "c0")  
   
 true\_c0=**sum**(**filter**(group\_c0, pred **==** TRUE)**$**n)  
 prop\_c0 <- **sum**(**filter**(group\_c0, pred **==** TRUE)**$**n) **/** **sum**(group\_c0**$**n)  
   
 prop\_table[prop\_table**$**threshold **==** threshold, "prop\_true\_c0"] <- prop\_c0  
 prop\_table[prop\_table**$**threshold **==** threshold, "true\_c0"] <- true\_c0  
   
}

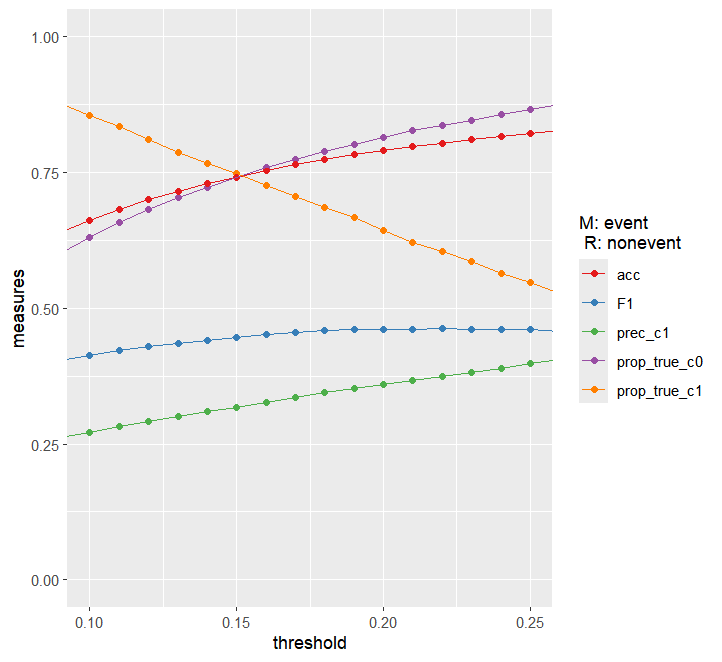
*#head(prop\_table, n=10)*  
  
  
prop\_table**$**n=**nrow**(dati\_validation)  
prop\_table**$**fp\_c1=**nrow**(dati\_validation)**-**prop\_table**$**true\_c0**-**prop\_table**$**true\_c1**-**prop\_table**$**fn\_c1 *# falsi positivi*  
prop\_table**$**acc=(prop\_table**$**true\_c0**+**prop\_table**$**true\_c1)**/nrow**(dati\_validation) *# accuracy*  
*# precision= veri c1 sul totale previsti come c1*  
prop\_table**$**prec\_c1=prop\_table**$**true\_c1**/**(prop\_table**$**true\_c1**+**prop\_table**$**fp\_c1)  
prop\_table**$**F1=2**\***(prop\_table**$**prop\_true\_c1**\***prop\_table**$**prec\_c1)**/**(prop\_table**$**prop\_true\_c1**+**prop\_table**$**prec\_c1)  
  
  
  
*#tail(prop\_table)*  
*# inputazione dati per non avere NaN zero denominatore manda valori a infinito*  
**library**(Hmisc)  
prop\_table**$**prec\_c1=**impute**(prop\_table**$**prec\_c1, 1)  
prop\_table**$**F1=**impute**(prop\_table**$**F1, 0)  
*#tail(prop\_table)*  
*# elimino colonne conteggi che non mi servono*   
prop\_table2 = prop\_table[,**-c**(4**:**8)]   
  
**library**(dplyr)  
**library**(tidyr)  
*#head(prop\_table2)*  
*# impilo metriche*   
gathered=prop\_table2 **%>%**  
 **gather**(x, y, prop\_true\_c1**:**F1)

## Warning: attributes are not identical across measure variables; they will be  
## dropped

*#head(gathered)*  
*#faccio plot metriche*   
**library**(ggplot2)  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n**R: nonevent")



*# zoom*  
gathered **%>%**  
 **ggplot**(**aes**(x = threshold, y = y, color = x)) **+**  
 **geom\_point**() **+**  
 **geom\_line**() **+**  
 **scale\_color\_brewer**(palette = "Set1") **+**  
 **labs**(y = "measures",  
 color = "M: event**\n** R: nonevent") **+**  
 **coord\_cartesian**(xlim = **c**(0.10, 0.25))



*# applico soglia*   
df**$**decision=**ifelse**(df**$**Probc1**>**0.15,"c1","c0")  
  
*# matrice confusione*  
**table**(df**$**Class,df**$**decision)

##   
## c0 c1  
## c1 2148 6335  
## c0 38785 13615

**confusionMatrix**(**as.factor**(df**$**decision),df**$**Class, positive = "c1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 6335 13615  
## c0 2148 38785  
##   
## Accuracy : 0.7411   
## 95% CI : (0.7376, 0.7446)   
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3109   
##   
## Mcnemar's Test P-Value : <2e-16  
##   
## Sensitivity : 0.7468   
## Specificity : 0.7402   
## Pos Pred Value : 0.3175   
## Neg Pred Value : 0.9475   
## Prevalence : 0.1393   
## Detection Rate : 0.1041   
## Detection Prevalence : 0.3277   
## Balanced Accuracy : 0.7435   
##   
## 'Positive' Class : c1   
##

*#head(df,n=20)*

## Applicazione Lasso su Dataset Esterno

dati\_plus\_prevLasso<-dati\_step\_4  
predizione<- **predict**(lassoPP, newdata = dati\_plus\_prevLasso, type = "prob")  
dati\_plus\_prevLasso**$**diabete=**ifelse**(predizione**$**c1**>**0.15,"c1","c0")  
dati\_plus\_prevLasso**$**diabete<-**factor**(dati\_plus\_prevLasso**$**diabete, levels = **c**('c1','c0'), labels=**c**('c1','c0'))  
*#str(dati\_plus\_prevLasso)*  
conf\_matrix\_STEP4 <- **confusionMatrix**(dati\_plus\_prevLasso**$**diabete, dati\_plus\_prevLasso**$**Diabetes\_binary)  
**print**(conf\_matrix\_STEP4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction c1 c0  
## c1 5289 11409  
## c0 1780 32257  
##   
## Accuracy : 0.74   
## 95% CI : (0.736, 0.744)   
## No Information Rate : 0.861   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.31   
##   
## Mcnemar's Test P-Value : <0.0000000000000002  
##   
## Sensitivity : 0.748   
## Specificity : 0.739   
## Pos Pred Value : 0.317   
## Neg Pred Value : 0.948   
## Prevalence : 0.139   
## Detection Rate : 0.104   
## Detection Prevalence : 0.329   
## Balanced Accuracy : 0.743   
##   
## 'Positive' Class : c1   
##

**table**(dati\_plus\_prevLasso **$**Diabetes\_binary)

##   
## c1 c0   
## 7069 43666