LABORATORIO R PER LA BIOSTATISTICA

Lavoro di gruppo svolto da: T. D'Adda, G. Doi, A. Giugni, A. Lucini Paioni, M. Susi

INTRODUZIONE

La neovascolarizzazione coroideale (CNV) è una complicanza oculare molto pericolosa per la vista che colpisce spesso gli adulti in età lavorativa ed è una delle principali cause di perdita della vista in tutto il mondo. In particolare, la CNV rappresenta una complicanza che interessa circa l'1% della popolazione affetta da miopia grave o patologica. Tale patologia è caratterizzata dalla proliferazione anomala, sotto e all'interno della retina, di nuovi vasi sanguigni; questi vasi possono rompersi e causare uno stravaso di sangue o fluido nella retina, provocando un deterioramento irreversibile della visione. Nei pazienti affetti da CNV miopica non trattata, la prognosi a lungo termine è sfavorevole nel 90% circa dei casi, ovvero il 90% circa dei pazienti affetti da una grave forma di CNV sviluppa una grave perdita della vista entro cinque anni.

Obiettivo del lavoro

L'obiettivo dell'analisi è quello di discriminare i soggetti a cui è stata diagnosticata la neovascolarizzazione coroideale dai soggetti a cui è stata diagnosticata una qualsiasi altra patologia oculistica in supporto alla pratica clinica. L'analisi viene effettuata attraverso la consultazione e lo studio del seguente database:

```
library(readxl)
path <- "C:/Users/andre/OneDrive/Documenti/unimib/magistrale/secondo anno/lab R per biostat/progetto/DA
DB <- read_excel(path)
rm(path)
```

Il database `DB` è fornito in formato excel e contiene 122 osservazioni relative a 65 variabili. Ciascuna osservazione è relativa ad un paziente e la variabile risposta d'interesse è `FINAL DIAGNOSIS`, una variabile categoriale che classifica i pazienti osservati in funzione della malattia oculistica che è stata loro diagnosticata. Nello specifico, se la variabile `FINAL DIAGNOSIS` ha valore `CNV`, ciò significa che al paziente è stata diagnosticata la neovascolarizzazione coroideale; invece, se la variabile `FINAL DIAGNOSIS` ha valore `HEMO`, `FIBROSIS`, `TDS` o `RPE IPERLASY`, ciò significa che al paziente è stata diagnosticata un'altra patologia.
L'obiettivo del lavoro è quello di caratterizzare i pazienti a cui è stata diagnosticata la CNV sfruttando le informazioni contenute nelle altre variabili del database.

METODI E ANALISI STATISTICA

1 - CLEANING

Verifica della correttezza dei valori inseriti nel database

Abbiamo verificato, colonna per colonna, che i valori delle variabili fossero stati inseriti correttamente. Ad esempio, per quanto riguarda la variabile `SEX`, essa può assumere solo due valori: `F` se il paziente è femmina,

`M` se il paziente è maschio.

```
table(DB$SEX)
```

```
##
##  F  m  M
## 83  1 38
```

Dall'analisi risulta che un'osservazione è stata inserita con la lettera `m` minuscola.
Questo errore deve essere corretto.

```
library(stringr)
DB$SEX <- toupper(DB$SEX)
table(DB$SEX)
```

```
##
##  F  M
## 83 39
```

In riferimento alla variabile `EVAL`, questa dovrebbe rappresentare una data, ma è stata inserita in modo scorretto nel database. Visto che si tratta di una variabile che non è utile ai fini dell'analisi, è stata rimossa.

```
DB <- DB[, -4]
```

Rimozione delle variabili 'ridondanti'

Innanzitutto abbiamo rimosso la prima colonna del database relativa agli ID dei pazienti in quanto viene inserita di default anche da R, quindi risulta doppia.

```
DB <- DB[, -1]
```

Dopodichè, per ciascuna variabile abbiamo valutato la presenza di eventuali missing values. Nello specifico abbiamo rimosso dal database le variabili che presentano una percentuale superiore al 30% di missing values, ossia: `RPE PERSISTENT and FLAT 1`, `RPE PERSISTENT and FLAT 2`, `RPE PERSISTENT and FLAT, WITH FOCAL INTERRUPTION 1`, `RPE PERSISTENT and FLAT, WITH FOCAL INTERRUPTION 2`, `RPE PERSISTENT and ELEVATED 1`, `RPE PERSISTENT and ELEVATED 2`, `RPE PERSISTENT and ELEVATED, WITH FOCAL INTERRUPTION 1`, `RPE PERSISTENT and ELEVATED, WITH FOCAL INTERRUPTION 2`, `VASCULAR NETWORK on OCTA 1` e `VASCULAR NETWORK on OCTA 2`.

```
DB <- DB[, -c(43, 44, 45, 46, 47, 48, 49, 50, 57, 58)]
```

Inoltre, considerando che la maggior parte delle variabili presenti nel database sono doppie, ossia sono frutto

della rilevazione di due clinici diversi, abbiamo calcolato l'indice di concordanza tra queste coppie di variabili in modo da poterne escludere una nel momento in cui risultassero concordi. Nello specifico, abbiamo definito una funzione (`funz_conc`) che calcola l'indice di concordanza tra due variabili (in modo da verificare che due variabili risultassero concordi con sufficiente frequenza) come:

$$Indice\ di\ concordanza = 1 - \frac{n\ di\ osservazioni\ uguali}{n\ totale\ di\ osservazioni}$$

Ad esempio, per quanto riguarda la coppia di variabili `FUZZY 1` e `FUZZY 2`, l'indice di concordanza è stato calcolato nel modo seguente:

```
funz_conc <- function(n_col) {
  uguali = 0
  diversi = 0
  for (i in seq(1:122)){
    if (DB[i, n_col] == DB[i, n_col+1]){
      uguali = uguali + 1
    } else {
      diversi = diversi + 1
    }
  }
  concordanza = 1 - diversi/(uguali+diversi)
  return(concordanza)
}

funz_conc(13)
```

```
## [1] 0.8606557
```

Le due variabili risultano concordi all'86%, per cui è possibile decidere di considerarne solo una ai fini dell'analisi. Abbiamo deciso di mantenere nel dataset la variabile `FUZZY 1`. Lo stesso ragionamento è stato applicato per tutte le coppie di variabili; in nessun caso l'indice di concordanza è risultato inferiore all'81%, per cui in tutti i casi abbiamo mantenuto solo la variabile relativa al primo dei due clinici.

```
DB <- DB[, -c(14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 51, 52)]
```

Ricodifica delle variabili

Come da consegna, le variabili `ELM`, `LEAK_STE` ed `EZ` vengono ricodificate come segue:

- Membrana limitante esterna (`ELM`):
  - `I` → `ELM_P_W=Y + ELM_I_W=Y`
  - `NI` → `ELM_P_W=Y + ELM_I_W=N`
  - `N` → `ELM_P_W=N`
  - `D` → `ELM_P_W=D`
- Leakage and staining (`LEAK_STE`):
  - `L` → `LEAKAGE=Y`
  - `S` → `STAINING=Y`

  - N → LEAKAGE=N + STAINING=N
  - D → LEAKAGE=D + STAINING=D

- Zona Ellissoide (EZ):
  - I → EZ_P_W=Y + EZ_I_W=Y
  - NI → EZ_P_W=Y + EZ_I_W=N
  - N → EZ_P_W=N
  - D → EZ_P_W=D

```
# Ricodifica variabile ELM
DB$ELM <- NA
for (j in 1:122){
  if (DB[j, 19] == "Y"){
    if (DB[j, 20] == "Y"){
      DB[j, 34] <- "I"
    } else if (DB[j, 20] == "N"){
      DB[j, 34] <- "NI"
    }
  } else if (DB[j, 19] == "N"){
    DB[j, 34] <- "N"
  } else if (DB[j, 19] == "D"){
    DB[j, 34] <- "D"
  }
}

# Ricodifica variabile LEAK_STE
DB$LEAK_STE <- NA
for (l in 1:122){
  if (DB[l, 28] == "Y"){
    DB[l, 35] <- "L"
  } else if (DB[l, 29] == "Y"){
    DB[l, 35] <- "S"
  } else if (DB[l, 28] == "N" & DB[l, 29] == "N"){
    DB[l, 35] <- "N"
  } else if (DB[l, 28] == "D" & DB[l, 29] == "D"){
    DB[l, 35] <- "D"
  }
}

# Ricodifica variabile EZ
DB$EZ <- NA
for (j in 1:122){
  if (DB[j, 22] == "Y"){
    if (DB[j, 23] == "Y"){
      DB[j, 36] <- "I"
    } else if (DB[j, 23] == "N"){
      DB[j, 36] <- "NI"
    }
  } else if (DB[j, 22] == "N"){
    DB[j, 36] <- "N"
  } else if (DB[j, 22] == "D"){
    DB[j, 36] <- "D"
  }}
```

A questo punto sono state eliminate dal database le variabili usate per la ricodifica.

```
DB <- DB[, -c(19, 20, 22, 23, 28, 29)]
```

Infine, è utile ricodificare la variabile risposta `FINAL DIAGNOSIS` come variabile dummy che assume valore pari a 1 se il soggetto è affetto da CNV, mentre assume valore pari a 0 se il soggetto è affetto da un'altra patologia.

```
DB$DIAGNOSI <- NA
DB$DIAGNOSI[DB$`FINAL DIAGNOSIS` == "CNV"] <- 1
DB$DIAGNOSI[DB$`FINAL DIAGNOSIS` != "CNV"] <- 0
table(DB$DIAGNOSI)
```

```
##
##  0  1
## 39 83
```

La variabile `FINAL DIAGNOSIS` non è stata comunque rimossa dal dataset in quanto, come vedremo in seguito, per l'applicazione di alcune tecniche di machine learning è richiesto che sia nel formato precedente.

Coercizione e rinomina delle variabili

E' bene trasformare tutte le variabili del database (eccetto `AGE` e `REF`), che sono state importate in formato character, in variabili categoriali.

```
DB[,-c(2,4)] <- lapply(DB[,-c(2,4)], factor)
```

Per quanto riguarda la variabile `REF`, relativa alla rifrazione, questa è stata importata in formato character, ma deve essere trasformata in variabile numerica.

```
DB$REF <- as.integer(DB$REF)
```

Come ultimo vengono modificati i nomi delle variabili in modo che non diano problemi ad essere letti dalle funzioni di R nelle successive elaborazioni (vengono tolti eventuali spazi e simboli speciali).

```
colnames(DB)[c(6,9:24,26,27)] <- c("RECENT_METAMORPHOPSIAS",
                                   "MACULAR_CHRA_ATROPHY",
                                   "PAPILLARY_CHORIORETINAL_ATROPHY",
                                   "ADJACENT_CHRA_ATROPHY",
                                   "INSIDE_CHRA_ATROPHY",
                                   "FUZZY",
```

```
                                "RETINAL_thickening",
                                "IRF",
                                "SRF",
                                "ERM",
                                "ELM_PRESENT_adjacent_to_the_lesion",
                                "EZ_PRESENT_adjacent_to_the_lesion",
                                "HD",
                                "SHADOW",
                                "PERSISTENT_CHOROID_on_SD_OCT",
                                "PERSISTENCE_OF_RPE_layer_within_the_lesion",
                                "SOPRA_LC...57",
                                "INCOMPLETE_MH",
                                "FINAL_DIAGNOSIS")
```

Al termine della fase di cleaning il database presenta 122 osservazioni relative a 31 variabili.

2 - TECNICHE DI MACHINE LEARNING

L'obiettivo del lavoro consiste in un'analisi descrittiva rispetto all'outcome, ovvero rispetto alla variabile `DIAGNOSI`, attraverso l'utilizzo delle informazioni contenute nel database e ai fini della pratica clinica. Al termine del processo di cleaning, il database contiene un numero elevato di variabili e non è detto che siano tutte significative ai fini dell'analisi descrittiva. Per questo motivo vengono applicate alcune tecniche di Machine Learning in modo da selezionare le variabili più significative. Nello specifico, abbiamo applicato due tecniche di Machine Learning: il Random Forest e il Boosting.

Random Forest

I Random Forest sono modelli non parametrici, introdotti da Breiman nel 2001 e utilizzati sia per la regressione che per la classificazione dei dati. Sono tra i metodi più diffusi di Machine Learning poiché si tratta di modelli di apprendimento automatico, molto flessibili, potenti e di facile comprensione; inoltre richiedono poche condizioni da porre sul modello che ha generato i dati. I Random Forest sono basati sugli alberi di decisione o regressione e consentono di trattare più problemi, regressione e classificazione a due o più classi, fornendo stimatori del classificatore di Bayes, che riduce al minimo l'errore di classificazione, o della funzione di regressione.

Nello specifico la costruzione degli alberi si suddivide in 3 fasi principali:

1. Si creano più copie bootstrap del dataset di training.
2. Si crea per ogni sottoinsieme di dati un albero decisionale, selezionando casualmente le variabili da includere nel modello. Per ogni albero si selezionano le variabili più significative secondo una regola decisionale posta all'inizio.
3. Si ripetono i passaggi precedenti n volte, generando così n alberi decisionali a partire da diversi campioni casuali del dataset originale.

Le previsioni finali si basano sulla media delle previsioni calcolate su ogni singolo albero o, nel caso di variabili categoriali, si seleziona la modalità più frequente. In questo modo si combinano tutti gli alberi per creare un solo modello.
Per applicare il metodo del Random Forest abbiamo utilizzato due pacchetti: `randomForest` e `VSURF`.

Funzione `randomForest`
I passaggi che devono essere effettuati per applicare tale funzione sono i seguenti:

- Definizione del seme;
- Definizione del dataset di training `train1`;
- Definizione del dataset di test `miopi.test1`;
- Utilizzo della funzione `randomForest` sul database, utilizzando le righe definite dal dataset di training con `subset`, il numero di variabili che vengono campionate in maniera casuale ad ogni divisione con `mtry` e il numero di alberi cresciuti con `ntree`. Nello specifico abbiamo settato il numero di alberi a 500 e il numero di variabili da selezionare ad ogni nodo come $p/3 = 10$.

Si tenga conto del fatto che per l'applicazione della funzione `randomForest` la variabile di outcome deve essere una variabile dummy. Di conseguenza abbiamo considerato come variabile di outcome la variabile `DIAGNOSI`.

```
DB_RF <- DB
DB_RF <- DB_RF[,-27]

require(randomForest)
set.seed(123)
train1 = sample(1:nrow(DB_RF), nrow(DB_RF)/2)
miopi.test1=DB_RF[-train1,"DIAGNOSI"]
RF.miopi=randomForest(DIAGNOSI~., data=DB_RF, subset=train1, mtry=10, ntree = 500)
```

A questo punto si applica la funzione `importance` all'oggetto creato con la funzione `randomForest`; tale funzione estrae l'oggetto `importance` contenuto al suo interno. Quest'oggetto viene poi convertito in dataframe e riordinato in ordine decrescente in base ai valori di `MeanDecreaseGini`. Questo permette di selezionare le variabili in base alla loro importanza tramite l'*indice di Gini*, che calcola la probabilità di una specifica caratteristica selezionata in maniera casuale di essere classificata in maniera non corretta. Questa misura dell'importanza della variabile è quindi basata sul decremento totale dell'impurità del nodo che risulta dallo split su quella variabile mediato da tutti gli alberi.

```
require(tidyverse)
var_imp_RF <- importance(RF.miopi)
var_imp_RF <- as.data.frame(var_imp_RF)
var_imp_RF <- var_imp_RF %>% arrange(desc(MeanDecreaseGini))
```

Funzione `VSURF`
Il VSURF è una procedura di selezione delle variabili basata sui Random Forest, sviluppata nel 2015 per gestire grandi moli di dati. Si tratta di un pacchetto molto versatile in grado di trattare dati di grandi dimensioni per la regressione o la classificazione. Questa tecnica si sviluppa in 3 passaggi:

1. *Thresholding step*: eliminazione delle variabili irrilevanti dal set di dati;
2. *Interpretation step*: selezione di tutte le variabili legate alla variabile risposta a scopo interpretativo;
3. *Prediction step*: affina la selezione eliminando la ridondanza nell'insieme delle variabili selezionato al secondo passaggio, a scopo di previsione.

Il pacchetto restituisce due sottoinsiemi di variabili, uno con variabili rilevanti per l'interpretazione e un altro che elimina le variabili ridondanti e che si concentra maggiormente sull'obiettivo di previsione. Si tenga

conto del fatto che per l'applicazione della funzione `VSURF` la variabile di outcome deve essere una variabile categoriale; di conseguenza abbiamo considerato come variabile di outcome la variabile `FINAL_DIAGNOSIS`.

```
DB_VSURF <- DB
DB_VSURF <- DB_VSURF[,-31]

library("VSURF")
library("mlbench")
set.seed(1234)
v_miopi <- VSURF(FINAL_DIAGNOSIS ~ ., data = DB_VSURF, na.action = na.omit)
```

```
## Thresholding step
## Estimated computational time (on one core): 1.2 sec.
##   |                                                                              |
## Interpretation step (on 17 variables)
## Estimated computational time (on one core): between 3.4 sec. and  3.4 sec.
##   |                                                                              |
## Prediction step (on 4 variables)
## Maximum estimated computational time (on one core): 0 sec.
##   |                                                                              |
```

Boosting
Il Boosting è una tecnica di Machine Learning che rientra nella categoria dell'Apprendimento Ensemble. Tale metodo funziona in modo simile al Random Forest, con la differenza che gli alberi di decisione sono selezionati sequenzialmente, ossia ogni albero viene creato utilizzando le informazioni provenienti da altri alberi. Nello specifico, dato un modello, l'albero si adatta ai residui anziché all'outcome; in questo modo ogni nuovo albero aiuta a correggere gli errori commessi dall'albero precedente. Il Boosting quindi non implica il campionamento bootstrap, ma si adatta a una versione modificata del dataset originale, permettendo ad alberi di forma diversa di aggiornare i residui.
I parametri che vengono considerati nella creazione degli alberi sono:

- Il numero degli alberi (`n.trees`), che di default è pari a 100.
- La profondità di ogni albero; l'opzione `interaction.depth=4` si riferisce alla profondità massima di interazioni della variabile e il valore 4 ad un modello fino a interazioni quadruple.
- Lo shrinkage (tasso di apprendimento)

Si tenga conto del fatto che per l'applicazione del Boosting la variabile di outcome deve essere una variabile dummy; di conseguenza abbiamo considerato come variabile di outcome la variabile `DIAGNOSI`.

```
DB_BOOST <- DB
DB_BOOST <- DB_BOOST[,-27]
```

Dopo aver impostato il dataset, lo suddividiamo in `train` e `test`, per permettere di valutare i risultati ottenuti col Boosting sul dataset di train, e in seguito effettuare previsioni sul dataset di test. Definiamo inoltre la k-fold cross validation, con k pari a 10.

```
set.seed (1234)
train = sample(1:nrow(DB_BOOST), nrow(DB_BOOST)/2)
DB_BOOST.train <- DB_BOOST[train ,]
DB_BOOST.test <- DB_BOOST[-train ,]

library(caret)
cvcontrol <- trainControl(method="repeatedcv", number = 10,
                          allowParallel=TRUE)
```

A questo punto, possiamo effettuare il Boosting sul dataset di train.

```
train.gbm <- train(as.factor(DIAGNOSI) ~ .,
                   data=DB_BOOST.train,
                   method="gbm",
                   verbose=F,
                   trControl=cvcontrol)
train.gbm
```

```
## Stochastic Gradient Boosting
##
## 61 samples
## 29 predictors
##  2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 56, 55, 55, 55, 55, 55, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8990476  0.7076605
##   1                  100      0.8990476  0.7076605
##   1                  150      0.8338095  0.5847393
##   2                   50      0.9014286  0.7093411
##   2                  100      0.8680952  0.6331507
##   2                  150      0.8538095  0.6097393
##   3                   50      0.8657143  0.6326605
##   3                  100      0.8680952  0.6331507
##   3                  150      0.8538095  0.6097393
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  2, shrinkage = 0.1 and n.minobsinnode = 10.
```

Sensitività e specificità delle diverse tecniche di ML

Per calcolare sensitività e specificità delle diverse tecniche di Machine Learning, utilizziamo i dataset di test.

Per il Random Forest, i valori sono i seguenti:

```
DB_RF.test <- DB_RF[-train1,]
yhat.RF = predict(RF.miopi,newdata=DB_RF.test)
confusionMatrix(as.factor(DB_RF.test$DIAGNOSI),yhat.RF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14  2
##          1  4 41
##
##                Accuracy : 0.9016
##                  95% CI : (0.7981, 0.963)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.0002149
##
##                   Kappa : 0.7557
##
##  Mcnemar's Test P-Value : 0.6830914
##
##             Sensitivity : 0.7778
##             Specificity : 0.9535
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.9111
##              Prevalence : 0.2951
##          Detection Rate : 0.2295
##    Detection Prevalence : 0.2623
##       Balanced Accuracy : 0.8656
##
##        'Positive' Class : 0
##
```

In particolare, la selezione mediante il Random Forest presenta una sensitività pari al 77.7% e una specificità pari al 95.3%.

Per il Boosting, invece, i valori sono i seguenti:

```
gbm.classTest <-  predict(train.gbm,
                          newdata = DB_BOOST.test,
                          type="raw")
gbm.classTest
```

```
##  [1] 1 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1
## [39] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1
## Levels: 0 1
```

```
confusionMatrix(as.factor(DB_BOOST.test$DIAGNOSI),gbm.classTest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 11  9
##          1  4 37
##
##                Accuracy : 0.7869
##                  95% CI : (0.6632, 0.8814)
##     No Information Rate : 0.7541
##     P-Value [Acc > NIR] : 0.3355
##
##                   Kappa : 0.4834
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.7333
##             Specificity : 0.8043
##          Pos Pred Value : 0.5500
##          Neg Pred Value : 0.9024
##              Prevalence : 0.2459
##          Detection Rate : 0.1803
##    Detection Prevalence : 0.3279
##       Balanced Accuracy : 0.7688
##
##        'Positive' Class : 0
##
```

La selezione mediante il Boosting presenta una sensitività pari al 73.3% e una specificità pari all'80.4%.

RISULTATI

SELEZIONE DELLE VARIABILI

Variabili selezionate con `randomForest`

Le variabili più significative selezionate con il metodo Random Forest, in particolare con la funzione `randomForest`, sono le seguenti:

```
var_imp_RF
```

```
##                                         MeanDecreaseGini
## FUZZY                                         8.13617860
## LEAK_STE                                      7.32243171
## RECENT_METAMORPHOPSIAS                        2.93236928
## AGE                                           1.58230437
## RETINAL_thickening                            1.16824621
```

```
## PERSISTENCE_OF_RPE_layer_within_the_lesion       0.89951013
## IRF                                              0.86336466
## REF                                              0.83570353
## EZ                                               0.70455034
## ELM                                              0.68510076
## ERM                                              0.32910306
## SHADOW                                           0.31363632
## EYE                                              0.30335822
## ELM_PRESENT_adjacent_to_the_lesion               0.24739536
## EZ_PRESENT_adjacent_to_the_lesion                0.24120863
## SCHISIS                                          0.21863220
## ADJACENT_CHRA_ATROPHY                            0.19530978
## HD                                               0.18903155
## PSE                                              0.18875441
## INCOMPLETE_MH                                    0.15917470
## MACULAR_CHRA_ATROPHY                             0.15286524
## SEX                                              0.14110277
## HAEMO                                            0.13674470
## INSIDE_CHRA_ATROPHY                              0.13332884
## SRF                                              0.05944154
## PERSISTENT_CHOROID_on_SD_OCT                     0.04505791
## SOPRA_LC                                         0.04375416
## POSITION                                         0.04056114
## PAPILLARY_CHORIORETINAL_ATROPHY                  0.01962685
```

Ovvero, tra tutti gli alberi del Random Forest, risulta che le variabili `LEAK_STE` e `FUZZY` sono le più importanti, seguite da `RECENT_METAMORPHOPSIAS`, `AGE` e `RETINAL_thickening`.

Variabili selezionate con `VSURF`

Un risultato leggermente diverso si ottiene con l'applicazione della funzione `VSURF`:

```
summary(v_miopi)
```

```
##
##  VSURF computation time: 2.8 secs
##
##  VSURF selected:
##  17 variables at thresholding step (in 1.4 secs)
##  4 variables at interpretation step (in 1.2 secs)
##  2 variables at prediction step (in 0.2 secs)
```

```
number <- c(1:26,28:30)
number[v_miopi$varselect.thres]
```

```
##  [1] 13 29  6 14  8 21  2 20  9 16 18 24 23 19 15 11 30
```

```
colnames(DB_VSURF[,c(number[v_miopi$varselect.thres])])
```

```
##  [1] "FUZZY"
##  [2] "LEAK_STE"
##  [3] "RECENT_METAMORPHOPSIAS"
##  [4] "RETINAL_thickening"
##  [5] "HAEMO"
##  [6] "SHADOW"
##  [7] "AGE"
##  [8] "HD"
##  [9] "MACULAR_CHRA_ATROPHY"
## [10] "SRF"
## [11] "ELM_PRESENT_adjacent_to_the_lesion"
## [12] "SOPRA_LC...57"
## [13] "PERSISTENCE_OF_RPE_layer_within_the_lesion"
## [14] "EZ_PRESENT_adjacent_to_the_lesion"
## [15] "IRF"
## [16] "ADJACENT_CHRA_ATROPHY"
## [17] "EZ"
```

Le variabili selezionate mediante questo metodo sono `FUZZY`, `LEAK_STE`, `RECENT_METAMORPHOSIAS` e, a seguire, `RETINAL_thickening`.

Variabili selezionate con il Boosting

Infine, si ottengono risultati simili anche con il metodo Boosting. Le variabili esplicative vengono suddivise in base alle loro categorie.

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
var_imp_boost <- varImp(train.gbm, scale = FALSE)
var_imp_boost
```

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 48)
##
##                         Overall
## FUZZYN                  12.6192
## EYER                     5.8853
## LEAK_STES                5.6515
## AGE                      3.5269
## FUZZYY                   2.1987
## EZN                      0.9623
## IRFN                     0.8797
## SHADOWY                  0.8078
## PSEY                     0.7198
## REF                      0.5975
## ELMN                     0.5405
```

```
## HAEMOY                   0.4721
## RETINAL_thickeningY      0.4044
## MACULAR_CHRA_ATROPHYY    0.3357
## SEXM                     0.3099
## SRFN                     0.2538
## INCOMPLETE_MHY           0.0000
## RECENT_METAMORPHOPSIASN  0.0000
## EZI                      0.0000
## SOPRA_LC...57Y           0.0000
```

Le variabili più importanti selezionate con il metodo Boosting sono le seguenti: `FUZZYN`, `EYER`, `LEAK_STES`, `AGE` e `FUZZYY`.

DISCUSSIONE

I metodi utilizzati hanno portato alla selezione di diverse variabili, di cui solo alcune risultano in comune tra le varie tecniche. Dal momento che il Random Forest ha generato valori di sensitività e specificità più elevati, l'analisi descrittiva successiva è stata effettuata sulle variabili che sono risultate più importanti per questa tecnica.

1 - ANALISI DESCRITTIVA

Dall'analisi effettuata è emerso che le variabili significative sono essenzialmente cinque: `LEAK_STE`, `FUZZY`, `RECENT_METAMORPHOSIAS`, `RETINAL_thickening` e `AGE`. Procediamo ad analizzare singolarmente le variabili selezionate.

Variabile `LEAK_STE`

```
library(table1)
library(kableExtra)
table1(~LEAK_STE | DIAGNOSI, data=DB)
```

	0	1	Overall
	(N=39)	(N=83)	(N=122)
LEAK_STE			
L	2 (5.1%)	64 (77.1%)	66 (54.1%)
N	6 (15.4%)	0 (0%)	6 (4.9%)
S	31 (79.5%)	19 (22.9%)	50 (41.0%)

La variabile `LEAK_STE` è una variabile categoriale che classifica i 122 pazienti inclusi nello studio in funzione della rilevazione di *leakage* (perdita di sangue o fluido nella retina), di *staining* (comparsa di aree fluorescenti nell'epitelio, successivamente all'installazione di fluoresceina) o dell'assenza di entrambi questi fattori.
Sia il leakage che lo staining sono indicativi di danno alla retina, ma non è detto che la rilevazione durante visita oculistica di queste caratteristiche in un paziente sia associata alla presenza di CNV. Infatti, lo staining è familiare tra i portatori di lenti a contatto, ma nella maggior parte dei casi non genera conseguenze clinicamente significative.
Dalla tabella si evidenzia che tutti coloro a cui è stata diagnosticata la CNV presentano un problema o di leakage o di staining. Nello specifico, nel 77.1% dei pazienti a cui è stata diagnosticata la CNV

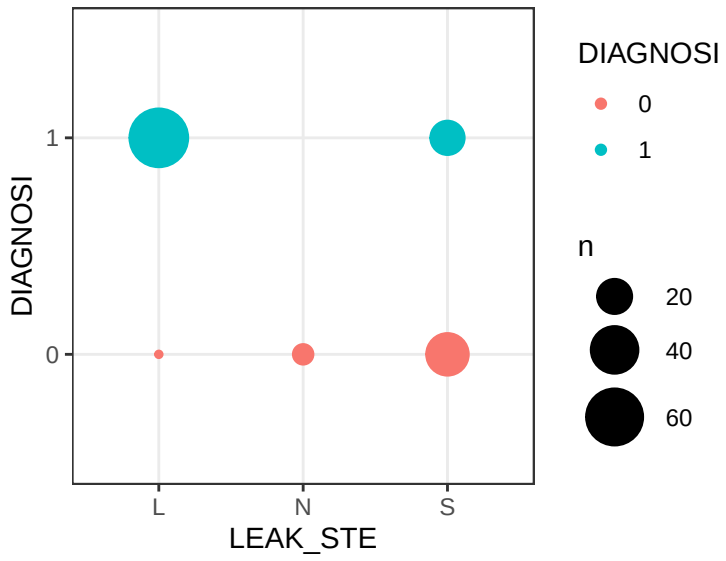
viene rilevata anche la presenza di leakage, mentre nel 22.9% dei pazienti affetti da CNV viene rilevata la presenza di staining. L'associazione tra la variabile `LEAK_STE` e la variabile `DIAGNOSI` viene valutata dal test Chi-quadrato di Pearson:

```
chisq <- chisq.test(table(DB$LEAK_STE, DB$DIAGNOSI))
pvalue <- chisq$p.value
pvalue
```

```
## [1] 1.608132e-13
```

Il test Chi-quadrato mette a verifica l'ipotesi nulla secondo cui le proporzioni di pazienti che presentano leakage o staining nelle categorie della variabile `DIAGNOSI` siano tra loro uguali. Scegliendo un livello di significatività pari a 0.05, il p-value associato a questo test risulta molto più basso; ciò comporta il rifiuto dell'ipotesi nulla e la conclusione che la diversa classificazione degli individui in funzione della variabile `LEAK_STE` è significativamente associata alla diagnosi di CNV.
Tali dati vengono rappresentati nel seguente bubble plot.

```
library(ggplot2)
descrLeak <- dplyr::count(DB, DIAGNOSI, LEAK_STE)
ggplot(descrLeak, aes(x=LEAK_STE, y=DIAGNOSI, size = n)) +
       geom_point(alpha=1, aes(color = DIAGNOSI)) +
       theme_bw() +
       coord_fixed(ratio = 1.5) +
       scale_size(range = c(1,10))
```



Da un'analisi qualitativa del grafico, sembrerebbe che la diagnosi di CNV sia maggiormente associata al fattore leakage, piuttosto che al fattore staining.

Variabile FUZZY

```
table1(~DB$FUZZY | DIAGNOSI, data=DB)
```

	0	1	Overall
	(N=39)	(N=83)	(N=122)
DB$FUZZY			
D	1 (2.6%)	11 (13.3%)	12 (9.8%)
N	36 (92.3%)	16 (19.3%)	52 (42.6%)
Y	2 (5.1%)	56 (67.5%)	58 (47.5%)

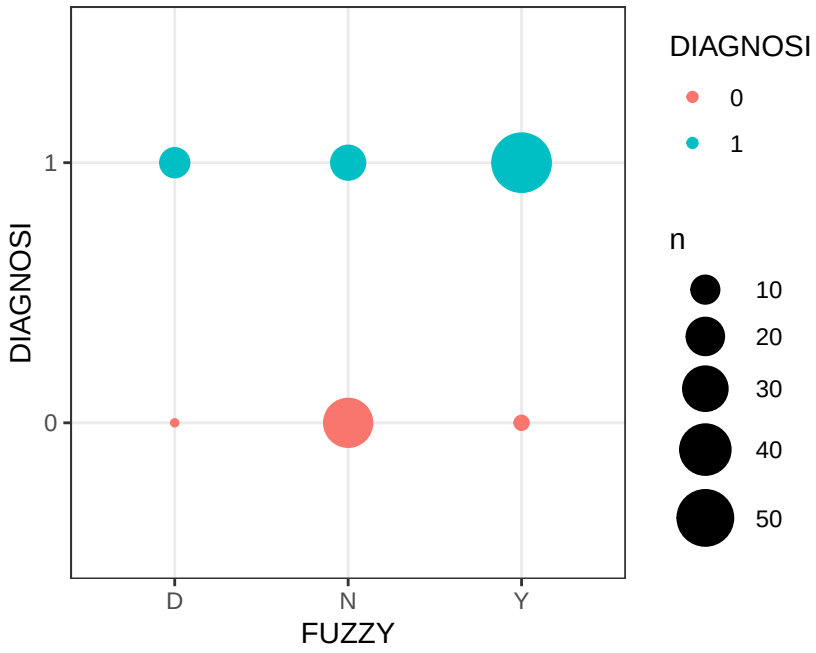
La variabile FUZZY è una variabile categoriale che classifica i 122 pazienti inclusi nello studio in funzione del fatto che presentino o meno delle lesioni irregolari a livello della retina (*lesioni fuzzy*). Dalla tabella si evidenzia che nel 67.5% dei pazienti a cui è stata diagnosticata la CNV viene rilevata la presenza di lesioni irregolari a livello della retina, mentre nel 19.3% dei pazienti affetti da CNV non viene rilevata la presenza di tali lesioni.
Sembrerebbe dunque che la diagnosi di CNV sia associata alla presenza di lesioni fuzzy a livello della retina. Tale associazione è confermata dal test Chi-quadrato di Pearson:

```
chisq <- chisq.test(table(DB$FUZZY, DB$DIAGNOSI))
pvalue <- chisq$p.value
pvalue
```

```
## [1] 2.577723e-13
```

Il test Chi-quadrato mette a verifica l'ipotesi nulla secondo cui le proporzioni di pazienti che presentano lesioni fuzzy nelle categorie della variabile DIAGNOSI siano tra loro uguali. Scegliendo un livello di significatività pari a 0.05, il p-value associato a questo test risulta molto più basso; ciò comporta il rifiuto dell'ipotesi nulla e la conclusione che la diversa classificazione degli individui in funzione della variabile FUZZY è significativamente associata alla diagnosi di CNV.
Tali dati vengono rappresentati nel seguente bubble plot.

```
descrFuzzy <- dplyr::count(DB, DIAGNOSI, FUZZY)
ggplot(descrFuzzy, aes(x=FUZZY, y=DIAGNOSI, size = n)) +
       geom_point(alpha=1, aes(color = DIAGNOSI)) +
       theme_bw() +
       coord_fixed(ratio = 1.5) +
       scale_size(range = c(1,10))
```



Variabile `RECENT_METAMORPHOPSIAS`

```
table1(~DB$RECENT_METAMORPHOPSIAS | DIAGNOSI, data=DB)
```

	0	1	Overall
	(N=39)	(N=83)	(N=122)
DB$RECENT_METAMORPHOPSIAS			
D	1 (2.6%)	0 (0%)	1 (0.8%)
N	23 (59.0%)	6 (7.2%)	29 (23.8%)
Y	15 (38.5%)	77 (92.8%)	92 (75.4%)

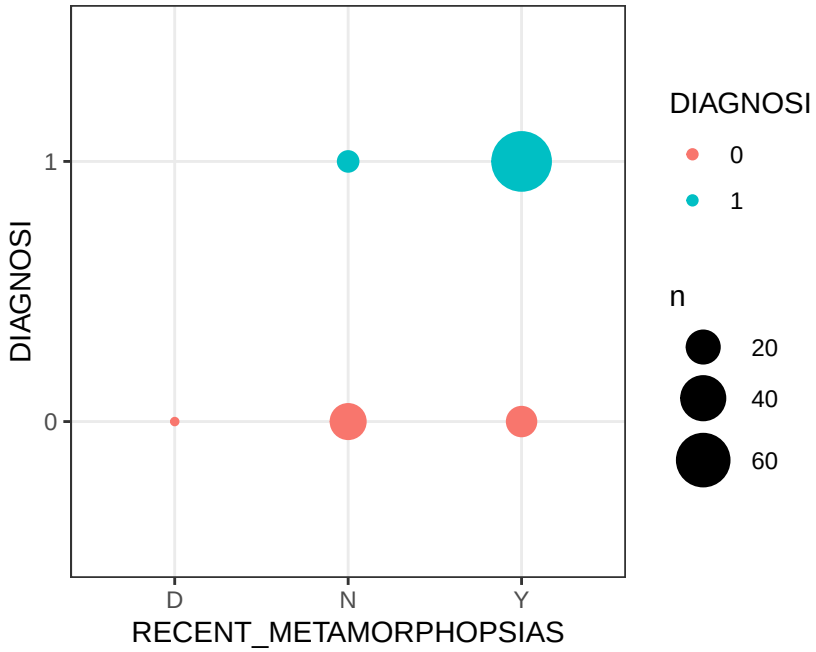
La variabile `RECENT_METAMORPHOPSIAS` è una variabile categoriale che classifica i 122 pazienti inclusi nello studio in persone che hanno avuto o no una metamorfopsia recente. La metamorfopsia è un difetto visivo che fa sì che gli oggetti lineari, come le linee di una griglia, appaiano curvi o arrotondati. È causata da problemi alla retina dell'occhio e, in particolare, alla macula. Dalla tabella si evidenzia che nel 92.8% dei pazienti a cui è stata diagnosticata la CNV viene rilevata tale condizione, mentre nel 7.2% dei pazienti affetti da CNV no.
Sembrerebbe dunque che la diagnosi di CNV sia associata ad una recente metamorfopsia. Tale associazione è confermata dal test Chi-quadrato di Pearson:

```
chisq <- chisq.test(table(DB$RECENT_METAMORPHOPSIAS, DB$DIAGNOSI))
pvalue <- chisq$p.value
pvalue
```

```
## [1] 6.228268e-10
```

Il test Chi-quadrato mette a verifica l'ipotesi nulla secondo cui le proporzioni di pazienti che hanno avuto metamorfopsia di recente nelle categorie della variabile `DIAGNOSI` siano tra loro uguali. Scegliendo un livello di significatività pari a 0.05, il p-value associato a questo test risulta molto più basso; ciò comporta il rifiuto dell'ipotesi nulla e la conclusione che la diversa classificazione degli individui in funzione della variabile `RECENT_METAMORPHOSIAS` è significativamente associata alla diagnosi di CNV.
Tali dati vengono rappresentati nel seguente bubble plot.

```
descrRec <- dplyr::count(DB, DIAGNOSI, RECENT_METAMORPHOPSIAS)
ggplot(descrRec, aes(x=RECENT_METAMORPHOPSIAS, y=DIAGNOSI, size = n)) +
       geom_point(alpha=1, aes(color = DIAGNOSI)) +
       theme_bw() +
       coord_fixed(ratio = 1.5) +
       scale_size(range = c(1,10))
```



Variabile `RETINAL_thickening`

```
table1(~DB$RETINAL_thickening | DIAGNOSI, data=DB)
```

	0	1	Overall
	(N=39)	(N=83)	(N=122)
DB$RETINAL_thickening			
D	7 (17.9%)	1 (1.2%)	8 (6.6%)
N	11 (28.2%)	1 (1.2%)	12 (9.8%)
Y	21 (53.8%)	81 (97.6%)	102 (83.6%)

La variabile `RETINAL_thickening` è una variabile categoriale che classifica i 122 pazienti inclusi nello studio

in funzione del fatto che presentino o meno un'ispessimento della retina (la causa più comune di questa condizione consiste in danni vascolari a livello del tessuto). Dalla tabella si evidenzia che nel 97.6% dei pazienti a cui è stata diagnosticata la CNV viene rilevato un'ispessimento della retina, mentre nell'1.2% dei pazienti affetti da CNV non viene rilevata tale condizione.
Sembrerebbe dunque che la diagnosi di CNV sia associata alla rilevazione di ispessimento della retina. Tale associazione è confermata dal test Chi-quadrato di Pearson:

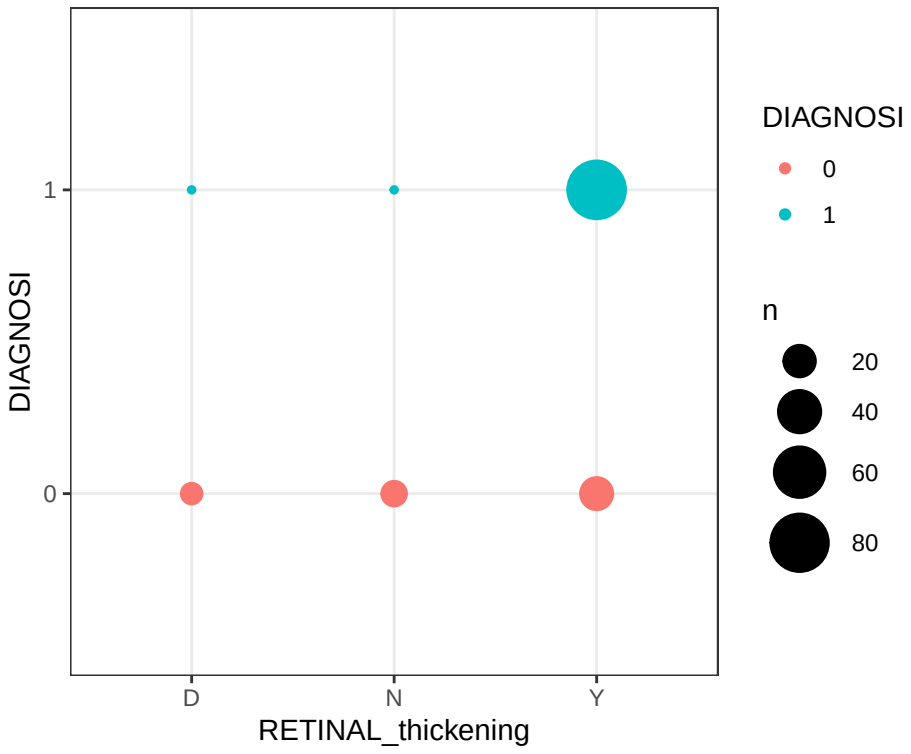
```
chisq <- chisq.test(table(DB$RETINAL_thickening, DB$DIAGNOSI))
pvalue <- chisq$p.value
pvalue
```

```
## [1] 8.866632e-09
```

Il test Chi-quadrato mette a verifica l'ipotesi nulla secondo cui le proporzioni di pazienti che presentano ispessimento della retina nelle categorie della variabile `DIAGNOSI` siano tra loro uguali. Scegliendo un livello di significatività pari a 0.05, il p-value associato a questo test risulta molto più basso; ciò comporta il rifiuto dell'ipotesi nulla e la conclusione che la diversa classificazione degli individui in funzione della variabile `RETINAL_thickening` è significativamente associata alla diagnosi di CNV.

Tali dati vengono rappresentati nel seguente bubble plot.

```
descrRet <- dplyr::count(DB, DIAGNOSI, RETINAL_thickening)
ggplot(descrRet, aes(x=RETINAL_thickening, y=DIAGNOSI, size = n)) +
       geom_point(alpha=1, aes(color = DIAGNOSI)) +
       theme_bw() +
       coord_fixed(ratio = 1.5) +
       scale_size(range = c(1,10))
```



Variabile AGE

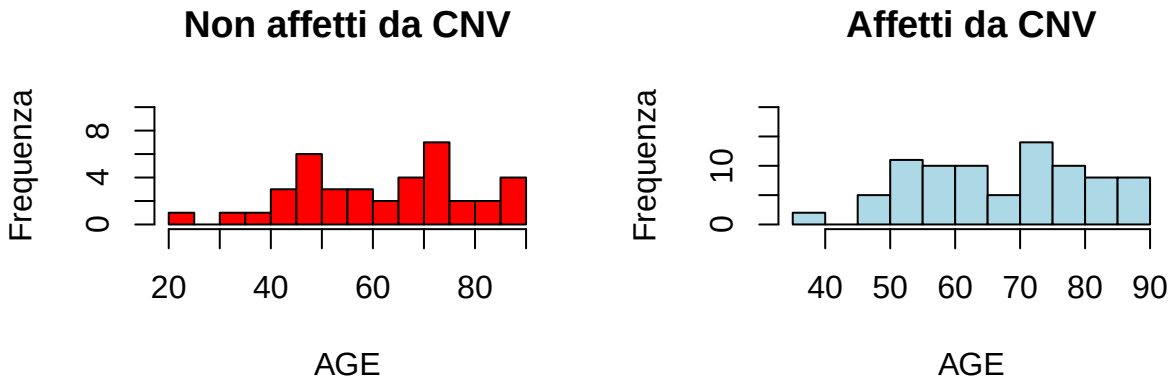
```
table1(~DB$AGE | DIAGNOSI, data=DB)
```

	0	1	Overall
	(N=39)	(N=83)	(N=122)
DB$AGE			
Mean (SD)	62.5 (16.8)	67.7 (13.0)	66.0 (14.5)
Median [Min, Max]	65.0 [22.0, 89.0]	68.0 [39.0, 89.0]	67.0 [22.0, 89.0]

Infine, la variabile AGE è una variabile quantitativa che esprime l'età dei pazienti inclusi nella studio. Dalla tabella si evidenzia che l'età media dei pazienti a cui è stata diagnosticata la CNV è di 67.7 anni (con un minimo di 39 e un massimo di 89), mentre l'età media dei pazienti a cui non è stata diagnosticata la CNV è leggermente inferiore, pari a 62.5 anni (con un minimo di 22 e un massimo di 89).

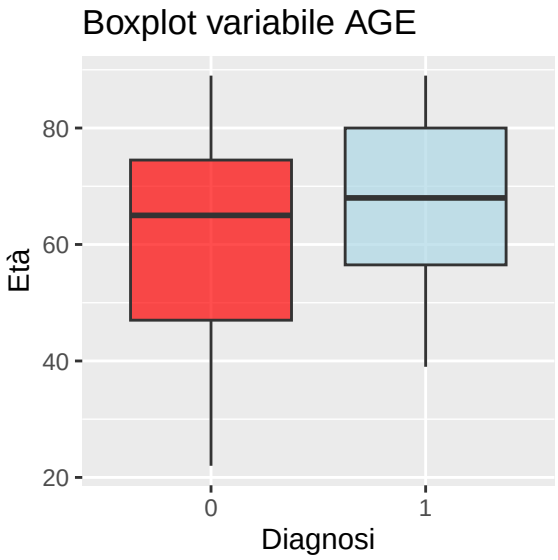
Visualizziamo i dati graficamente. Il primo istogramma mostra la distribuzione per età dei pazienti affetti da altre patologie oculistiche; mentre il secondo istogramma mostra la distribuzione per età dei pazienti affetti da CNV.

```
par(mfrow=c(1,2))
hist(DB$AGE[DB$DIAGNOSI == 0],
     col = 'red',
     ylim = c(0,10),
     main = "Non affetti da CNV",
     xlab = "AGE",
     ylab = "Frequenza",
     breaks = 10)
hist(DB$AGE[DB$DIAGNOSI == 1],
     col = 'light blue',
     main = "Affetti da CNV",
     xlab = "AGE",
     ylab = "Frequenza",
     breaks = 10,
     ylim = c(0,20))
```



Le stesse distribuzioni possono essere visualizzate tramite box-plot.

```
ggplot(DB, aes(x = DIAGNOSI, y = AGE)) +
  geom_boxplot(notch = FALSE, fill = c("red","lightblue"), alpha = 0.7) +
  labs(y = "Età", x = "Diagnosi", title = "Boxplot variabile AGE")
```



Osservando i grafici notiamo che la distribuzione della variabile `AGE` nelle categorie della variabile `DIAGNOSI` non è approssimabile ad una distribuzione Normale (la mediana della distribuzione della variabile `AGE` per i pazienti non affetti da CNV non si colloca al centro del box-plot, ad esempio). Per questo motivo, per valutare mediante un test d'ipotesi la possibile associazione tra la variabile `AGE` e la diagnosi di CNV abbiamo applicato il test di Wilcoxon.

```
wilcox.test(AGE ~ DIAGNOSI, data = DB,
       alternative = 'two.sided',
       conf.level = .95)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  AGE by DIAGNOSI
## W = 1337, p-value = 0.1227
## alternative hypothesis: true location shift is not equal to 0
```

Il test Wilcoxon mette a verifica l'ipotesi nulla secondo cui le età medie dei pazienti nelle categorie della variabile `DIAGNOSI` siano tra loro uguali. Scegliendo un livello di significatività pari a 0.05, il p-value associato a questo test risulta più alto, portando quindi ad accettare l'ipotesi nulla di uguaglianza delle età medie nelle due categorie di pazienti.

Questo non è ciò che ci si aspetterebbe da una variabile selezionata tra le più importanti nel modello: ci aspetteremmo infatti che le età siano significativamente diverse nei due gruppi.

La non significatività del test potrebbe essere dovuta a diversi fattori, tra cui:

- la numerosità del campione è molto bassa: i due gruppi posti a confronto hanno infatti numerosità pari rispettivamente a 81 (per gli individui affetti da CNV) e 39 (per gli individui affetti da altre patologie). Nel caso fosse possibile, sarebbe utile ripetere il test su campioni più grandi
- il test di Wilcoxon effettua un'analisi univariata della relazione tra la variabile `DIAGNOSI` e la variabile `AGE`: potrebbero esserci dei fattori di confondimento dovuti alla presenza di altre variabili che non vengono prese in considerazione.

2- VANTAGGI E SVANTAGGI

Random Forest

In generale gli alberi decisionali sono costrutti molto "naturali", in particolare quando sono esplicativi e le variabili sono categoriche (ancora meglio quando sono binarie); inoltre sono molto facili da analizzare e interpretare. I modelli sono invarianti rispetto alle trasformazioni dei predittori nello spazio e il trattamento dei valori mancanti è più semplice.
Nello specifico, i Random Forest combinano la semplicità degli alberi decisionali con la flessibilità e la potenza di un modello di insieme, rendendo il modello più robusto. Infatti, uno dei principali svantaggi degli alberi decisionali è che sono molto inclini ad un adattamento eccessivo ai dati (overfitting), cioè funzionano bene sui dati di training, ma spesso non sono abbastanza flessibili per fare previsioni su altri dataset di validazione; questo porta ad avere un bias basso ma varianza molto alta. Per eliminare il problema si potrebbe ridurre il numero di alberi, ma si ridurrebbe di conseguenza anche il loro potere predittivo.
I Random Forest sono un metodo alternativo che calcola la media di più alberi decisionali utilizzando diverse variabili selezionate in maniera casuale. Questa tecnica permette di ridurre la varianza del modello previsionale, grazie alla riduzione del forte legame tra l'albero e i dati di training, e di aggirare il problema della correlazione tra i dati.
Gli svantaggi di questa tecnica sono che aumenta il BIAS, il che indica che il modello non sempre riesce ad individuare dei pattern nei dati; inoltre spesso sono necessarie grandi moli di dati e non sempre si arriva a trovare il modello migliore.

Boosting

Per quanto riguarda il Boosting, questa tecnica generalmente richiede più tempo del Random Forest a causa del fatto che gli alberi sono costruiti in modo sequenziale, tuttavia risultati di benchmark hanno dimostrato che sono 'better learners' rispetto ai Random Forest.
Un altro difetto è che sono inclini all'overfitting. Per eliminare questo problema si può ricorrere alla costruzione di alberi più generalizzati, utilizzando una combinazione di parametri come lo shrinkage e la profondità dell'albero.

3 - METODI NON UTILIZZATI

Oltre ai metodi utilizzati, esistono anche altri metodi utili per effettuare la selezione delle variabili che sono stati presi in considerazione ma successivamente scartati per diversi motivi.

PCA

L'*analisi delle componenti principali* (PCA) è una tecnica di Machine Learning che viene utilizzata per la selezione delle variabili e che permette di ridurre il numero delle variabili di un dataset, restituendo delle

combinazioni lineari delle variabili originali (componenti principali), che presentano maggiore variabilità.
In generale, la PCA cerca il miglior spazio di rappresentazione (di dimensione ridotta) che consenta una visualizzazione ottimale nello spazio R^I della nuvola N_K; tale nuvola viene proiettata su un sottospazio dello spazio R^I, scelto in modo tale da ridurre al minimo la distorsione. Il sottospazio viene selezionato in modo tale che le distanze tra i punti proiettati sia il più vicino possibile alle distanze tra i punti originali.
L'analisi delle componenti principali è il metodo di riduzione delle variabili più indicato solo quando le variabili sono quantitative. Siccome il dataset che abbiamo analizzato è formato principalmente da variabili categoriali, a seguito dell'applicazione della PCA si ha una dispersione troppo elevata tra le variabili, che non riescono ad essere accorpate in combinazioni lineari coerenti.
Inoltre, dal momento che la PCA si basa sulla varianza in comune tra le variabili, ha senso includere nel modello solo variabili che risultano almeno moderatamente correlate con le altre. Abbiamo calcolato quindi l'indice KMO (Kaiser-Meyer-Olkin), che rappresenta il grado in cui ciascuna variabile osservata è prevista dalle altre variabili nel set di dati e indica l'idoneità del dataset per l'analisi fattoriale. L'indice è risultato prossimo a 0.5, il che ci ha portato a ritenere il dataset non idoneo all'analisi PCA. Per questi motivi abbiamo deciso di non inserire tale tecnica nell'analisi.

Bagging

Abbiamo deciso di escludere dal progetto anche la tecnica del Bagging, una procedura che permette di ridurre la varianza di un metodo statistico e che aggrega i valori medi predetti per ogni replica bootstrap. Nello specifico, vengono effettuate tante repliche bootstrap del dataset e all'interno di ognuna viene fatto girare un albero, andando poi a calcolare il valore predetto come media dei valori predetti di tutti gli alberi costruiti nelle repliche.
Tale metodo porta ad una maggiore precisione nella previsione rispetto all'utilizzo di un singolo albero, ma utilizza tutte le covariate per costruire ogni albero. In questo modo, se esiste un predittore molto forte, verrà preso in tutte le repliche bootstrap, oscurando il ruolo delle altre variabili. Quindi, in alternativa abbiamo ritenuto più conveniente utilizzare il Random Forest che permette di costruire una serie di alberi su campioni bootstrap del training set ma, a differenza del Bagging, seleziona casualmente ogni volta un certo numero di predittori.

Funzione `tuneRF`

La funzione `tuneRF` è una funzione del pacchetto `randomForest` che, partendo da un valore iniziale di `mtry` (può essere specificato o si può utilizzare un valore di default), permette di cercare il valore ottimale di `mtry` per `randomForest`, utilizzando per la selezione la stima di Out-of-Bar error (OOB), e di vedere quali sono le variabili selezionate.
La funzione ha la seguente sintassi:

```
tuneRF(x, y,
       mtryStart = , #valore iniziale di mtry
       ntreeTry = , #numero di alberi utilizzati nel tuning step
       stepFactor= , #valore per cui mtry viene aumentato o diminuito ad ogni iterazione
       improve=0.05, #valore dell'aumento di OOB necessario affinchè la ricerca continui
       )
```

Per avere dei risultati confrontabili, è auspicabile utilizzare in questa funzione lo stesso numero di alberi utilizzati con la funzione `randomForest`. In questa funzione il numero di alberi utilizzato è 500, in quando è un valore che assicura un valore di OOB (e degli altri errori) stabile (fisso restando il seme). Specificando questo numero di alberi nell'opzione `nTreetry` della funzione `tuneRF` non viene restituito alcun output in quanto l'algoritmo non procede con la ricerca del numero ottimale di `mtry`. Utilizzando un numero di alberi

molto più basso (ad esempio pari a 10) l'algoritmo restituisce dei risultati, tuttavia OOB avrebbe valori troppo instabili. Per questo motivo si è preferito non utilzzare questa procedura.

CONCLUSIONI

Attraverso le analisi effettuate sul database `DB`, a seguito di un importante processo di cleaning per rendere i dati consistenti, abbiamo caratterizzato i pazienti a cui viene diagnosticata la neovascolarizzazione coroideale (CNV) in supporto alla pratica clinica.
In particolare, mediante l'analisi descrittiva delle variabili significative selezionate dalle tecniche di Machine Learning, è possibile concludere che i pazienti a cui viene diagnosticata la CNV sono in media più anziani rispetto ai pazienti a cui vengono diagnosticate altre patologie oculistiche; quindi l'età gioca un ruolo importante nel processo di diagnosi della patologia in esame.
Allo stesso modo, la rilevazione di danni a livello della retina quali il leakage, lo staining, le lesioni fuzzy e l'ispessimento della retina dovrebbero essere considerate dai clinici come campanelli d'allarme per la diagnosi di CNV in quanto la loro presenza è positivamente associata alla patologia in esame.
Infine, anche la rilevazione di una metamorfosi a livello delle cellule del cristallino che provoca problemi nella messa a fuoco è sintomo di CNV.

Occorre tenere conto del fatto che il campione analizzato presenta una numerosità molto bassa, pari a 122 pazienti: con tale dimensione campionaria non è possibile giungere a conclusioni certe e generalizzabili ad una più ampia popolazione, ma solo generare spunti per degli studi più approfonditi.
Inoltre, non sono disponibili informazioni sul metodo di campionamento utilizzato.
Occorrerebbe quindi ripetere le analisi utilizzando un campione più ampio e ottenuto mediante un metodo di campionamento che assicuri (per quanto possibile) di minimizzare l'effetto dei fattori di confondimento.

...