PEC2 - TFM - Master Bioinformática y Bioestadística - UOC

Jordi Cabral

18/11/2019

Contents

Predicciones de interacción genética mediante métodos de Machine Learning

El presente documento corresponde a la parte del código de programación de la PEC2 del TFM, con los primeros cálculos de los diferentes algoritmos

Preparación del entorno

Empezamos cargando las librerías que vamos a necesitar

```
library(knitr)
library(ggplot2)
library(caret)
library(class)
library(gmodels)
library(kableExtra)
library(neuralnet)
library(kernlab)
library(rpart)
library(rpart.plot)
library(randomForest)
library(rattle)
library(RColorBrewer)
```

Preparamos los directorios

```
workingDir <- getwd()
dataDir <- file.path(workingDir,"../Data/")
```

Comprobamos los ficheros existentes en el directorio de datos

```
dir(path = dataDir)
```

```
[1] "allpairsfeta.rds"              "SGA10_BremKruglyak2005_v1.rds"
[3] "SGA16_BremKruglyak2005_v1.rds"
```

Cargar y leer los datos

Cargamos los datos

```
SGA10 <- readRDS(paste0(dataDir, "SGA10_BremKruglyak2005_v1.rds"))
SGA16 <- readRDS(paste0(dataDir, "SGA16_BremKruglyak2005_v1.rds"))
allpaired <- readRDS(paste0(dataDir, "allpairsfeta.rds"))
```

Exploramos y visualizamos superficialmente los datos, viendo su estructura, los primeros registros, y un resumen global

```
dim(SGA10)
```

```
[1] 227   5
```

```
head(SGA10)
```

```
                  GENE1   GENE2  SGAsco       PCC        NPC
YER114C_YPL045W YPL045W YER114C -0.0072 0.4946259 0.04637893
YLR085C_YMR167W YLR085C YMR167W -0.2766 0.4092974 0.04794095
YOR237W_YPL091W YPL091W YOR237W -0.0201 0.2745026 0.01513016
YHR031C_YPR120C YPR120C YHR031C -0.3233 0.2381275 0.01984171
YMR052W_YOL009C YOL009C YMR052W -0.3202 0.4037952 0.01966951
YDL088C_YDR524C YDR524C YDL088C -0.0812 0.4832784 0.02292296
```

```
summary(SGA10)
```

```
    GENE1              GENE2               SGAsco
```

```
 Length:227         Length:227         Min.   :-0.92210
 Class :character   Class :character   1st Qu.:-0.14750
 Mode  :character   Mode  :character   Median :-0.06810
                                       Mean   :-0.13711
                                       3rd Qu.:-0.03765
                                       Max.   :-0.00720
      PCC                NPC
 Min.   :-0.04256   Min.   :0.01352
 1st Qu.: 0.19316   1st Qu.:0.02127
 Median : 0.28407   Median :0.02784
 Mean   : 0.34374   Mean   :0.03239
 3rd Qu.: 0.44519   3rd Qu.:0.03732
 Max.   : 0.93761   Max.   :0.13079
```

```
str(SGA10)
```

```
'data.frame':   227 obs. of  5 variables:
 $ GENE1 : chr  "YPL045W" "YLR085C" "YPL091W" "YPR120C" ...
 $ GENE2 : chr  "YER114C" "YMR167W" "YOR237W" "YHR031C" ...
 $ SGAsco: num  -0.0072 -0.2766 -0.0201 -0.3233 -0.3202 ...
 $ PCC   : num  0.495 0.409 0.275 0.238 0.404 ...
 $ NPC   : num  0.0464 0.0479 0.0151 0.0198 0.0197 ...
```

```
dim(SGA16)
```

```
[1] 757   5
```

```
head(SGA16)
```

```
                      GENE1     GENE2  SGAsco       PCC        NPC
YDR131C_YDR311W   YDR311W   YDR131C -0.0571 0.5847008 0.04834784
YDR108W_YIL039W   YIL039W   YDR108W -0.1693 0.4339379 0.03350580
YDR228C_YLR277C   YDR228C   YLR277C -0.1522 0.5943217 0.02597431
YJR093C_YPR114W   YJR093C   YPR114W -0.1016 0.2035627 0.02536385
YDL220C_YOR378W   YDL220C   YOR378W -0.1578 0.4063060 0.03295474
YMR075C.A_YNL148C YNL148C YMR075C.A -0.0814 0.2913731 0.02318996
```

```
str(SGA16)
```

```
'data.frame':   757 obs. of  5 variables:
 $ GENE1 : chr  "YDR311W" "YIL039W" "YDR228C" "YJR093C" ...
 $ GENE2 : chr  "YDR131C" "YDR108W" "YLR277C" "YPR114W" ...
 $ SGAsco: num  -0.0571 -0.1693 -0.1522 -0.1016 -0.1578 ...
 $ PCC   : num  0.585 0.434 0.594 0.204 0.406 ...
 $ NPC   : num  0.0483 0.0335 0.026 0.0254 0.033 ...
```

```
dim(allpaired)
```

```
[1] 17461095       14
```

```
head(allpaired)
```

```
               V1    V2 Homology PhysicalInt CommonReg Colocalization
Q0045_Q0050 Q0045 Q0050        1           0         0              0
Q0045_Q0055 Q0045 Q0055        1           0         0              0
Q0045_Q0060 Q0045 Q0060        1           0         0              0
Q0045_Q0065 Q0045 Q0065        1           0         0              0
Q0045_Q0070 Q0045 Q0070        1           0         0              0
Q0045_Q0075 Q0045 Q0075        0           0         0              0
```

```
            Phenotype Ohnology Complexes SameFunction SameProtein
Q0045_Q0050         0        0         0            0           0
Q0045_Q0055         0        0         0            0           0
Q0045_Q0060         0        0         0            0           0
Q0045_Q0065         0        0         0            0           0
Q0045_Q0070         0        0         0            0           0
Q0045_Q0075         0        0         0            0           0
                 fcBP      fcCC      fcMF
Q0045_Q0050 0.2941176 0.3243243 0.4848485
Q0045_Q0055 0.3452381 0.4324324 0.8181818
Q0045_Q0060 0.3918919 0.4324324 0.8709677
Q0045_Q0065 0.3918919 0.4324324 0.8709677
Q0045_Q0070 0.3866667 0.4324324 0.8709677
Q0045_Q0075 0.1578947 0.3243243 0.1388889
```

```
tail(allpaired)
```

```
                    V1      V2 Homology PhysicalInt CommonReg
YPR201W_YPR202W YPR201W YPR202W        0           0         0
YPR201W_YPR203W YPR201W YPR203W        0           0         0
YPR201W_YPR204W YPR201W YPR204W        0           0         0
YPR202W_YPR203W YPR202W YPR203W        0           0         1
YPR202W_YPR204W YPR202W YPR204W        0           0         0
YPR203W_YPR204W YPR203W YPR204W        0           0         0
                Colocalization Phenotype Ohnology Complexes SameFunction
YPR201W_YPR202W              0         0        0         0            0
YPR201W_YPR203W              0         0        0         0            0
YPR201W_YPR204W              0         0        0         0            0
YPR202W_YPR203W              0         0        0         0            0
YPR202W_YPR204W              0         0        0         0            0
YPR203W_YPR204W              0         0        0         0            0
                SameProtein       fcBP       fcCC       fcMF
YPR201W_YPR202W           0 0.06250000 0.08333333 0.07692308
YPR201W_YPR203W           0 0.06250000 0.08333333 0.07692308
YPR201W_YPR204W           0 0.04166667 0.08333333 0.02564103
YPR202W_YPR203W           0 1.00000000 1.00000000 1.00000000
YPR202W_YPR204W           0 0.11111111 1.00000000 0.03703704
YPR203W_YPR204W           0 0.11111111 1.00000000 0.03703704
```

```
str(allpaired)
```

```
'data.frame':   17461095 obs. of  14 variables:
 $ V1            : chr  "Q0045" "Q0045" "Q0045" "Q0045" ...
 $ V2            : chr  "Q0050" "Q0055" "Q0060" "Q0065" ...
 $ Homology      : chr  "1" "1" "1" "1" ...
 $ PhysicalInt   : chr  "0" "0" "0" "0" ...
 $ CommonReg     : chr  "0" "0" "0" "0" ...
 $ Colocalization: chr  "0" "0" "0" "0" ...
 $ Phenotype     : chr  "0" "0" "0" "0" ...
 $ Ohnology      : chr  "0" "0" "0" "0" ...
 $ Complexes     : chr  "0" "0" "0" "0" ...
 $ SameFunction  : chr  "0" "0" "0" "0" ...
 $ SameProtein   : chr  "0" "0" "0" "0" ...
 $ fcBP          : num  0.294 0.345 0.392 0.392 0.387 ...
 $ fcCC          : num  0.324 0.432 0.432 0.432 0.432 ...
```

```
 $ fcMF          : num  0.485 0.818 0.871 0.871 0.871 ...
```

Transformación de los datos

Eliminamos las variables identificativas de los genes individuales

```
allpaired$V1 <- NULL
allpaired$V2 <- NULL
```

Convertimos a tipo numérico todas las variables del dataset *allpaireid*

```
allpaired[] <- lapply(allpaired, as.numeric)
str(allpaired)
```

```
'data.frame':   17461095 obs. of  12 variables:
 $ Homology      : num  1 1 1 1 1 0 0 0 0 0 ...
 $ PhysicalInt   : num  0 0 0 0 0 0 1 1 1 0 ...
 $ CommonReg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Colocalization: num  0 0 0 0 0 0 0 0 0 0 ...
 $ Phenotype     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Ohnology      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Complexes     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SameFunction  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SameProtein   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ fcBP          : num  0.294 0.345 0.392 0.392 0.387 ...
 $ fcCC          : num  0.324 0.432 0.432 0.432 0.432 ...
 $ fcMF          : num  0.485 0.818 0.871 0.871 0.871 ...
```

Generamos dos nuevos data frames filtrando sólo los pares existentes en los SGAs

```
allpaired_10 <- allpaired[row.names(allpaired) %in% row.names(SGA10), ]
dim(allpaired_10)
```

```
[1] 190  12
```

Vemos que en lugar de 227 registros hay 190, debido a que algunos genes han cambiado de identificador

```
allpaired_16 <- allpaired[row.names(allpaired) %in% row.names(SGA16), ]
dim(allpaired_16)
```

```
[1] 635  12
```

Vemos que en lugar de 757 registros, hay 635, debido a que algunos genes han cambiado de identificador

Generamos un nuevo dataset, uniendo los campos de SGA y allpaired, para cada uno de los experimientos

```
SGA10_full <- merge(SGA10, allpaired_10, by=0)
rownames(SGA10_full) <- SGA10_full$Row.names
SGA10_full$Row.names <- NULL
dim(SGA10_full)
```

```
[1] 190  17
```

```
head(SGA10_full)
```

```
                  GENE1   GENE2  SGAsco       PCC        NPC Homology
YAL005C_YLL024C YAL005C YLL024C -0.6751 0.7712090 0.09606661        1
YAL010C_YJL100W YJL100W YAL010C -0.0360 0.2724282 0.02458174        0
```

```
YAL010C_YOR090C YOR090C YAL010C -0.1459 0.2136824 0.03680466        0
YAL010C_YPR121W YAL010C YPR121W -0.0455 0.2723902 0.01942903        0
YAL011W_YPL213W YPL213W YAL011W -0.2238 0.1785020 0.02189094        0
YAL024C_YHR077C YAL024C YHR077C -0.0531 0.4628442 0.05343258        0
                PhysicalInt CommonReg Colocalization Phenotype Ohnology
YAL005C_YLL024C           0         1               1         0        0
YAL010C_YJL100W           0         0               0         0        0
YAL010C_YOR090C           0         0               0         0        0
YAL010C_YPR121W           0         0               0         0        0
YAL011W_YPL213W           0         1               0         0        0
YAL024C_YHR077C           0         0               0         0        0
                Complexes SameFunction SameProtein       fcBP       fcCC
YAL005C_YLL024C         0            1           1 0.58015267 0.95000000
YAL010C_YJL100W         0            0           0 0.06329114 0.29032258
YAL010C_YOR090C         0            0           0 0.02500000 0.39130435
YAL010C_YPR121W         0            0           0 0.02222222 0.02380952
YAL011W_YPL213W         0            0           0 0.22857143 0.34883721
YAL024C_YHR077C         0            0           0 0.03067485 0.31578947
                     fcMF
YAL005C_YLL024C 1.00000000
YAL010C_YJL100W 0.03703704
YAL010C_YOR090C 0.07142857
YAL010C_YPR121W 0.08333333
YAL011W_YPL213W 0.12500000
YAL024C_YHR077C 0.14285714
```

```
SGA16_full <- merge(SGA16, allpaired_16, by=0)
rownames(SGA16_full) <- SGA16_full$Row.names
SGA16_full$Row.names <- NULL
dim(SGA16_full)
```

```
[1] 635  17
```

```
head(SGA16_full)
```

```
                  GENE1   GENE2  SGAsco        PCC        NPC Homology
YAL001C_YMR308C YAL001C YMR308C -0.1404 0.57943228 0.03871764        0
YAL005C_YLL024C YAL005C YLL024C -0.7370 0.77120903 0.09606661        1
YAL010C_YDR367W YAL010C YDR367W -0.1024 0.09345734 0.01553714        0
YAL010C_YJL100W YJL100W YAL010C -0.0419 0.27242823 0.02458174        0
YAL011W_YPL213W YPL213W YAL011W -0.4322 0.17850196 0.02189094        0
YAL013W_YPL055C YAL013W YPL055C -0.1845 0.54066298 0.02274498        0
                PhysicalInt CommonReg Colocalization Phenotype Ohnology
YAL001C_YMR308C           0         0               1         0        0
YAL005C_YLL024C           0         1               1         0        0
YAL010C_YDR367W           0         0               0         0        0
YAL010C_YJL100W           0         0               0         0        0
YAL011W_YPL213W           0         1               0         0        0
YAL013W_YPL055C           0         0               0         0        0
                Complexes SameFunction SameProtein       fcBP      fcCC
YAL001C_YMR308C         0            0           0 0.18248175 0.3829787
YAL005C_YLL024C         0            1           1 0.58015267 0.9500000
YAL010C_YDR367W         0            0           0 0.02631579 0.3962264
YAL010C_YJL100W         0            0           0 0.06329114 0.2903226
YAL011W_YPL213W         0            0           0 0.22857143 0.3488372
```

```
YAL013W_YPL055C         0            0           0 0.25196850 0.2941176
                      fcMF
YAL001C_YMR308C 0.06451613
YAL005C_YLL024C 1.00000000
YAL010C_YDR367W 0.25000000
YAL010C_YJL100W 0.03703704
YAL011W_YPL213W 0.12500000
YAL013W_YPL055C 0.07692308
```

Eliminamos las variables "descriptivas" de los genes

```
SGA10_full$GENE1 <- NULL
SGA10_full$GENE2 <- NULL

SGA16_full$GENE1 <- NULL
SGA16_full$GENE2 <- NULL
```

Reproduciendo el modelo de regresión lineal del TFG Aina Rill de la bibliografía, para verificar que partimos de la base correcta

Modelo 1 (Tabla 1)

```
mod1 <- lm(SGAsco ~. -NPC -fcMF , SGA10_full)
summary(mod1)
```

```
Call:
lm(formula = SGAsco ~ . - NPC - fcMF, data = SGA10_full)

Residuals:
     Min       1Q   Median       3Q      Max
-0.49977 -0.04257  0.02682  0.06667  0.41775

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.06514    0.02438  -2.671 0.008261 **
PCC             0.06176    0.05360   1.152 0.250823
Homology        0.04954    0.06646   0.745 0.457038
PhysicalInt     0.05644    0.08797   0.642 0.521982
CommonReg      -0.01943    0.03538  -0.549 0.583582
Colocalization -0.11032    0.04828  -2.285 0.023504 *
Phenotype      -0.06806    0.06928  -0.982 0.327226
Ohnology       -0.20133    0.08714  -2.310 0.022018 *
Complexes       0.15479    0.08802   1.759 0.080385 .
SameFunction   -0.15191    0.04264  -3.563 0.000472 ***
SameProtein    -0.28151    0.07911  -3.559 0.000479 ***
fcBP           -0.16859    0.07486  -2.252 0.025549 *
fcCC           -0.10993    0.05058  -2.173 0.031071 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1271 on 177 degrees of freedom
Multiple R-squared:  0.5301,    Adjusted R-squared:  0.4982
F-statistic: 16.64 on 12 and 177 DF,  p-value: < 2.2e-16
```

```
(RMSE.lm1 <- sqrt(mean(mod1$residuals^2)))
```

```
[1] 0.1226716
```

Modelo 2 (Tabla 2)

```
mod2 <- lm(SGAsco ~ Ohnology + Complexes + SameProtein +
               SameFunction + fcBP + fcCC + NPC, SGA10_full)
summary(mod2)
```

```
Call:
lm(formula = SGAsco ~ Ohnology + Complexes + SameProtein + SameFunction +
    fcBP + fcCC + NPC, data = SGA10_full)

Residuals:
     Min       1Q   Median       3Q      Max
-0.52410 -0.03507  0.02359  0.06385  0.35944

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.003326   0.024082   0.138 0.890296
Ohnology     -0.207477   0.071485  -2.902 0.004161 **
Complexes     0.166463   0.071605   2.325 0.021190 *
SameProtein  -0.253821   0.070119  -3.620 0.000382 ***
SameFunction -0.114987   0.041052  -2.801 0.005645 **
fcBP         -0.118014   0.061202  -1.928 0.055379 .
fcCC         -0.124799   0.048765  -2.559 0.011305 *
NPC          -1.754990   0.579212  -3.030 0.002802 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1251 on 182 degrees of freedom
Multiple R-squared:  0.5321,    Adjusted R-squared:  0.5141
F-statistic: 29.57 on 7 and 182 DF,  p-value: < 2.2e-16
```

```
(RMSE.lm2 <- sqrt(mean(mod2$residuals^2)))
```

```
[1] 0.1224099
```

Los resultados, si bien no son idénticos, son muy parecidos.

Tomaremos estos valoes de RMSE como referencia para evaluar el rendimiento de los modelos basados en machine learning.

Algoritmos Machine Learning

Prepararemos los datos tanto para el dataset *SGA10* como *SGA16*, pero inicialmente realizaremos los cálculos sólo sobre los datos de *SGA10*, para simplificar la comparación, ya que es sobre el que tenemos la referencia de regresión lineal.

Selección de datos para *training* y *test*

Dividimos los dataset en dos, para entrenamiento y validación.

```
bound_10 <- floor(nrow(SGA10_full)*0.67)
bound_16 <- floor(nrow(SGA16_full)*0.67)

seed <- c(12345)

set.seed(seed)
row_train_10 <- sample(seq_len(nrow(SGA10_full)), size = bound_10)

set.seed(seed)
row_train_16 <- sample(seq_len(nrow(SGA16_full)), size = bound_16)

SGA10_train <- SGA10_full[row_train_10, ]
SGA10_test <- SGA10_full[-row_train_10, ]

SGA16_train <- SGA16_full[row_train_16, ]
SGA16_test <- SGA16_full[-row_train_16, ]
```

Algoritmo k-NN

Empezaremos por el algoritmo k-NN (Nearest Neighbors), que aunque se usa habitualmente en modelos de clasificación, también puede ofrecer buenos resulados de regresión.

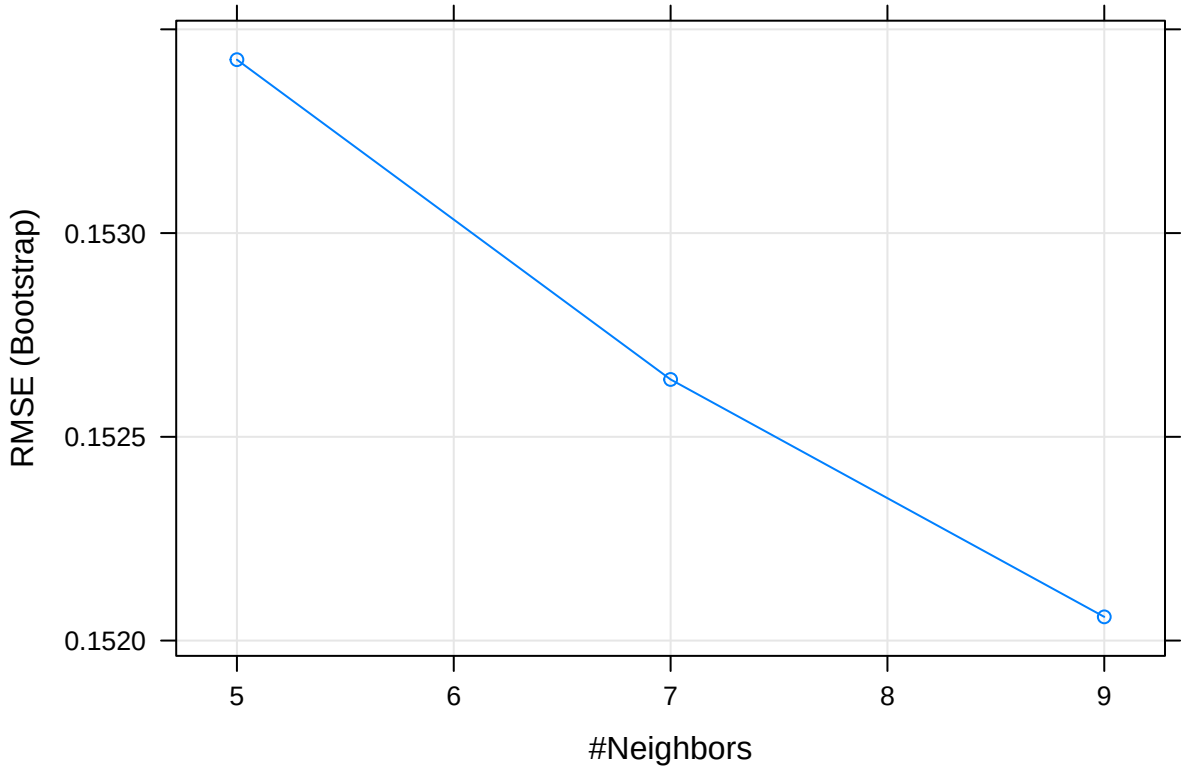
Transformación de los datos

No es necesario, al disponer ya de los datos normalizados y en datasets de *training* y *test* de pasos anteriores.

Entrenamiento del modelo

Para el coeficiente PCC:

```
set.seed(seed)
knn_model_PCC_1 <- train(SGAsco~. -NPC, data=SGA10_train, method="knn")

plot(knn_model_PCC_1)
```




```
k_PCC_1 <- rownames(knn_model_PCC_1$bestTune)
knn_model_PCC_1$results[k_PCC_1,]
```

```
  k      RMSE  Rsquared        MAE     RMSESD RsquaredSD      MAESD
3 9 0.1520582 0.2686059 0.09507469 0.02505281  0.1068463 0.01175275
```

Realizar la predicción

```
pred_PCC_1 <- knn_model_PCC_1 %>% predict(SGA10_test)
```

Evaluar el rendimiento del modelo

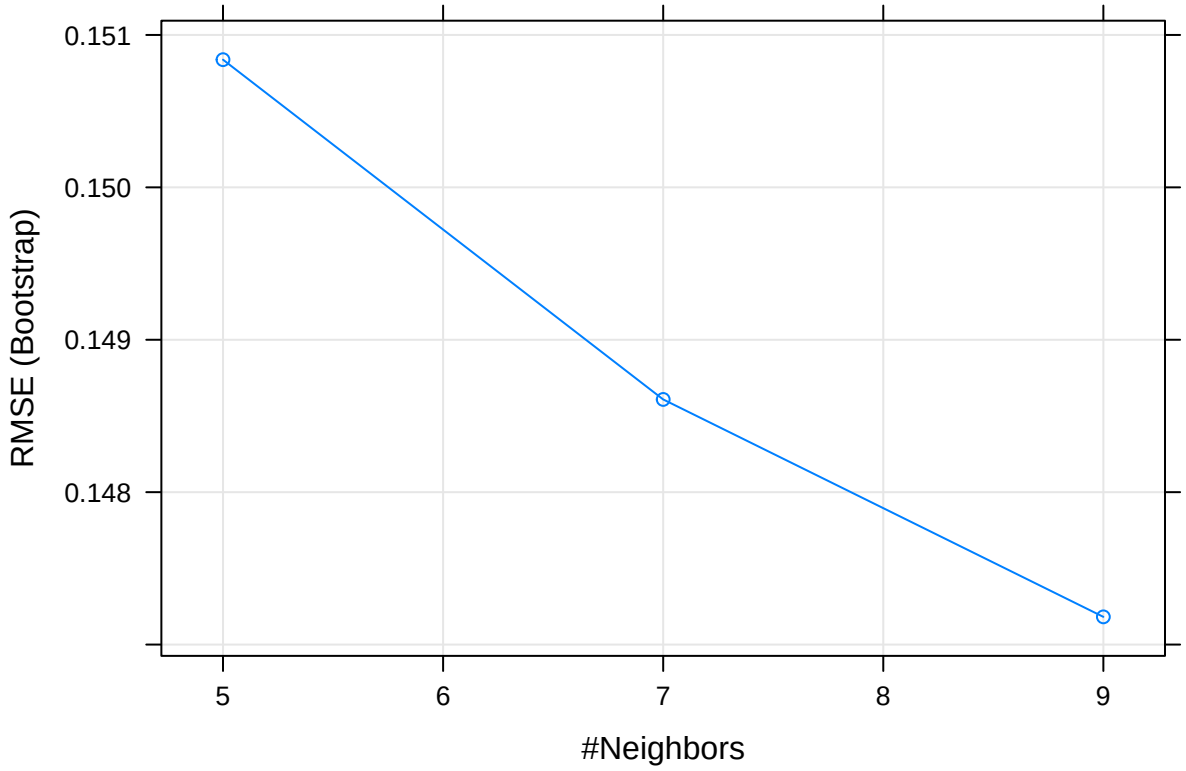
Generamos una tabla para poder visualizar un resumen de las diferentes combinaciones:

```
tabla_knn <- data.frame(Option = c("Default", "Param"), PCC = NA, NPC = NA)

(tabla_knn[1,2] <- RMSE(pred_PCC_1, SGA10_test$SGAsco))
```

```
[1] 0.1465124
```

Para la variable NPC

```
set.seed(seed)
knn_model_NPC_1 <- train(SGAsco~. -PCC, data=SGA10_train, method="knn")

plot(knn_model_NPC_1)
```




```
k_NPC_1 <- rownames(knn_model_NPC_1$bestTune)
knn_model_NPC_1$results[k_NPC_1,]
```

```
  k      RMSE  Rsquared        MAE     RMSESD RsquaredSD      MAESD
3 9 0.1471818 0.3067904 0.09299212 0.02663945  0.1221129 0.01359124
```

Realizar la predicción

```
pred_NPC_1 <- knn_model_NPC_1 %>% predict(SGA10_test)
```

Evaluar el rendimiento del modelo

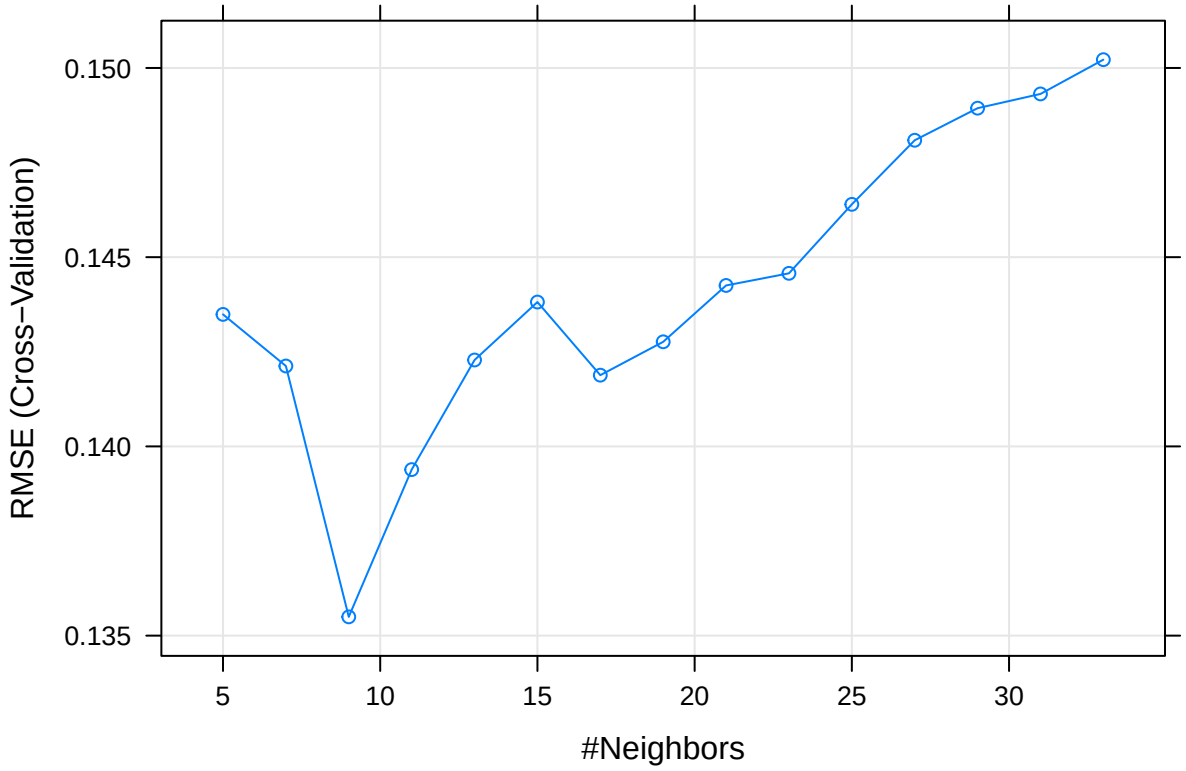
```
(tabla_knn[1,3] <- RMSE(pred_NPC_1, SGA10_test$SGAsco))
```

```
[1] 0.1417423
```

Mejorar el rendimiento del modelo

Podemos probar cambiando alguno de los parámetros por defecto

```
set.seed(seed)
knn_model_PCC_2 <- train(SGAsco~. -NPC, data=SGA10_train, method="knn",
                         trControl = trainControl("cv", number = 10),
                         preProcess = c("center","scale"),
                         tuneLength = 15)
plot(knn_model_PCC_2)
```



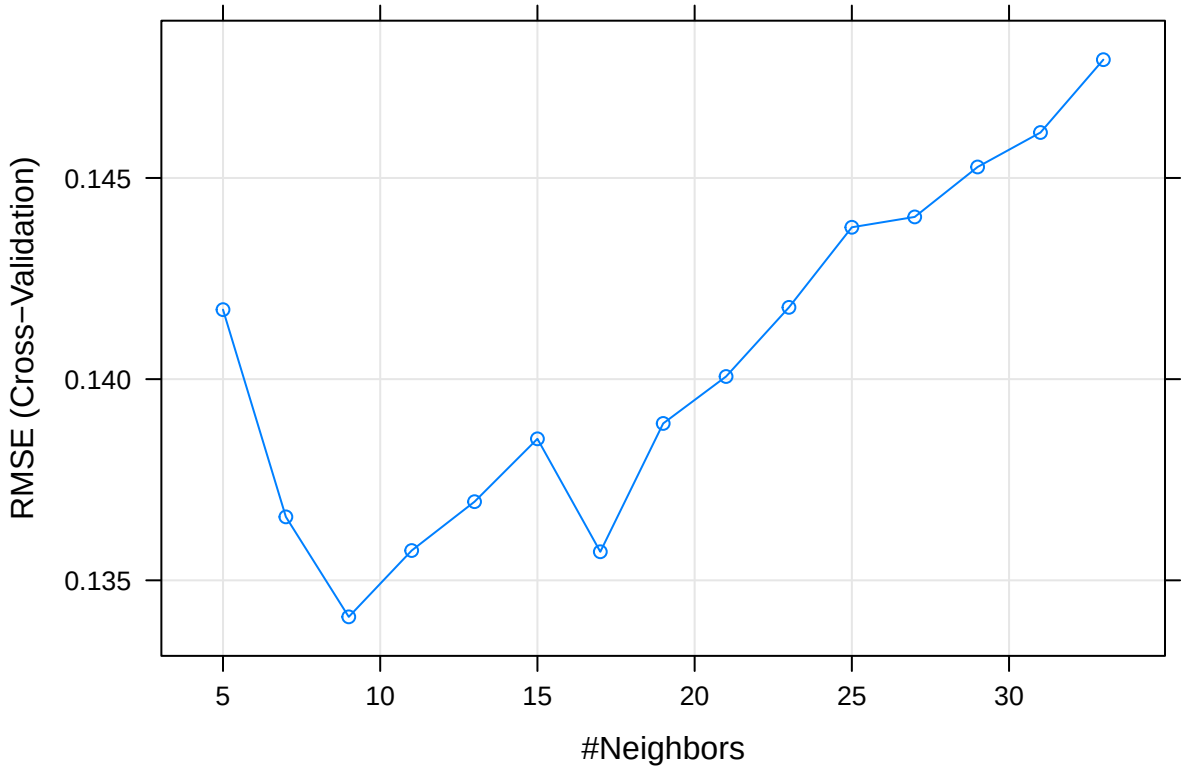

```
k_PCC_2 <- rownames(knn_model_PCC_2$bestTune)
knn_model_PCC_2$results[k_PCC_2,]
```

```
  k     RMSE  Rsquared        MAE     RMSESD RsquaredSD      MAESD
3 9 0.135496 0.5182416 0.09255648 0.05163305  0.2710733 0.02537471
```

```
pred_PCC_2 <- knn_model_PCC_2 %>% predict(SGA10_test)
(tabla_knn[2,2] <- RMSE(pred_PCC_2, SGA10_test$SGAsco))
```

```
[1] 0.1488071
```

```
set.seed(seed)
knn_model_NPC_2 <- train(SGAsco~. -PCC, data=SGA10_train, method="knn",
                         trControl = trainControl("cv", number = 10),
                         preProcess = c("center","scale"),
                         tuneLength = 15)
plot(knn_model_NPC_2)
```




```
k_NPC_2 <- rownames(knn_model_NPC_2$bestTune)
knn_model_NPC_2$results[k_NPC_2,]
```

```
  k      RMSE  Rsquared        MAE    RMSESD RsquaredSD      MAESD
3 9 0.1340918 0.5420777 0.09006229 0.0497747    0.30274 0.02297983
```

```
pred_NPC_2 <- knn_model_NPC_2 %>% predict(SGA10_test)
(tabla_knn[2,3] <- RMSE(pred_NPC_2, SGA10_test$SGAsco))
```

```
[1] 0.1522432
```

Resumen resultados algoritmo knn:

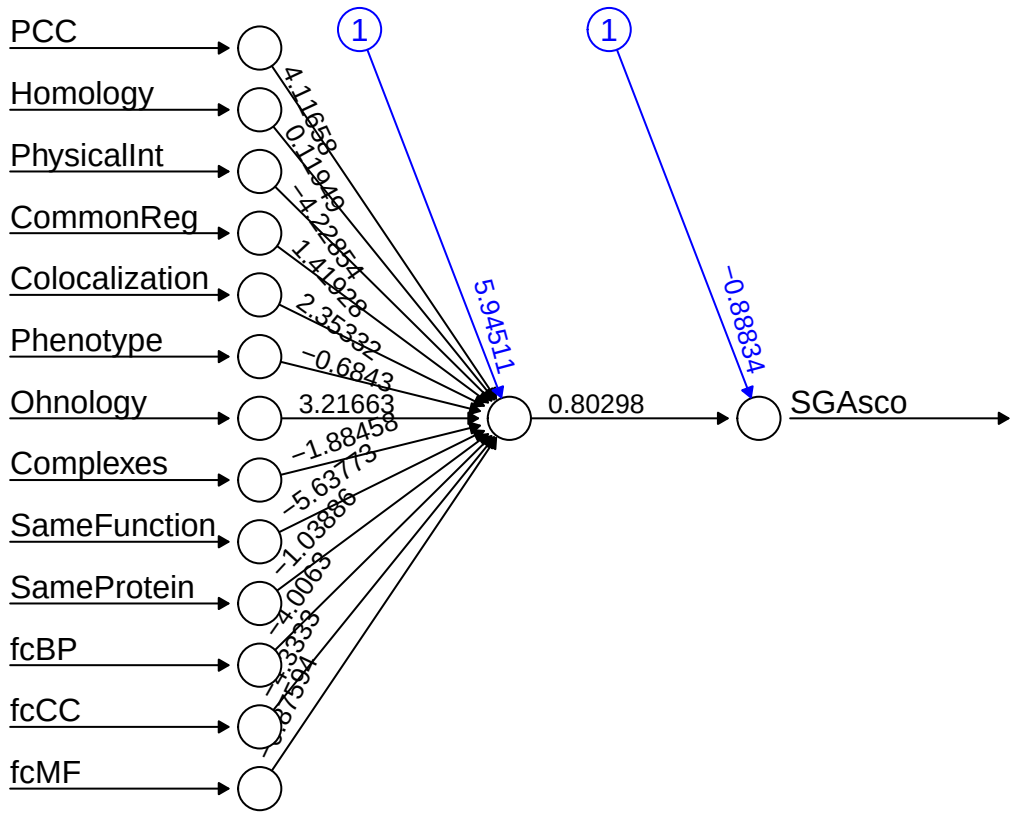
```
kable(tabla_knn) %>%
  kable_styling(full_width = F, position = "left")
```

Option	PCC	NPC
Default	0.1465124	0.1417423
Param	0.1488071	0.1522432

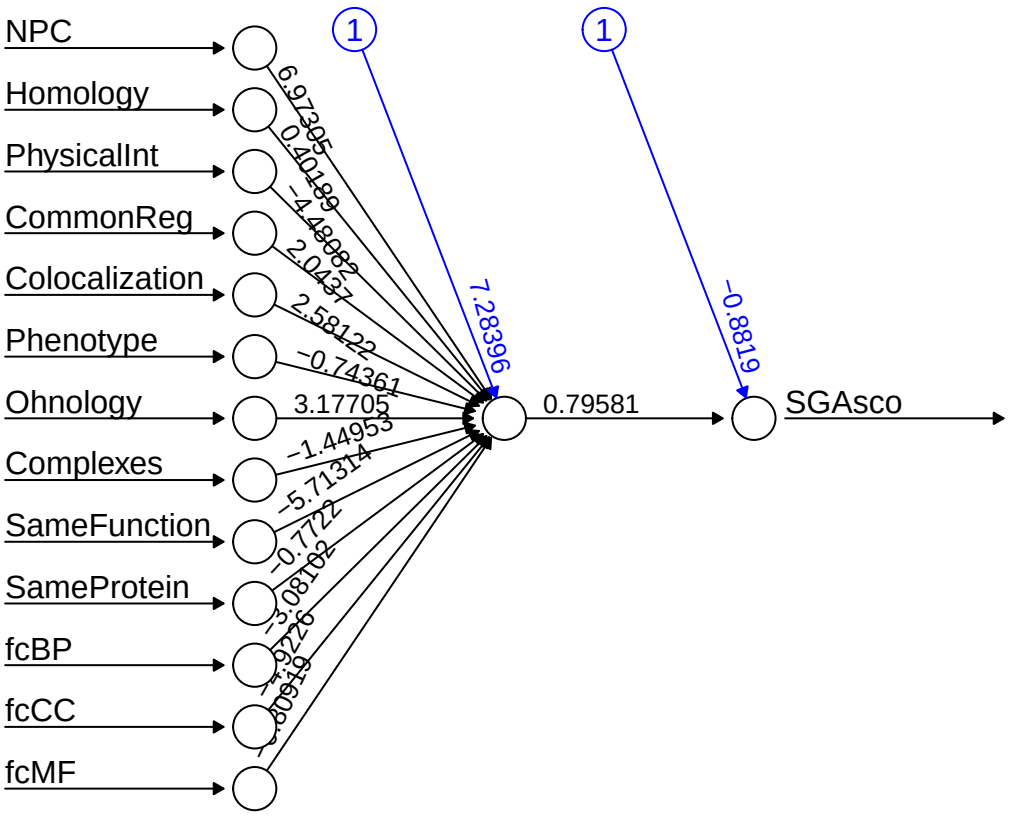
Algoritmo Artificial Neural Network

Entrenamiento del modelo

```
set.seed(seed)
ann_model_1_PCC <- neuralnet(SGAsco ~. -NPC, data=SGA10_train)
plot(ann_model_1_PCC, rep="best")
```




```
set.seed(seed)
ann_model_1_NPC <- neuralnet(SGAsco ~. -PCC, data=SGA10_train)
plot(ann_model_1_NPC, rep="best")
```



Evaluar el rendimiento del modelo

```
tabla_ann <- data.frame(hidden = c(1,2,3), PCC = NA, NPC = NA)

ann_results_1_PCC <- compute(ann_model_1_PCC, SGA10_test)
ann_pred_1_PCC <- ann_results_1_PCC$net.result
(tabla_ann[1,2] <- RMSE(ann_pred_1_PCC, SGA10_test$SGAsco))
```

```
[1] 0.2053081
```

```
ann_results_1_NPC <- compute(ann_model_1_NPC, SGA10_test)
ann_pred_1_NPC <- ann_results_1_NPC$net.result
(tabla_ann[1,3] <- RMSE(ann_pred_1_NPC, SGA10_test$SGAsco))
```

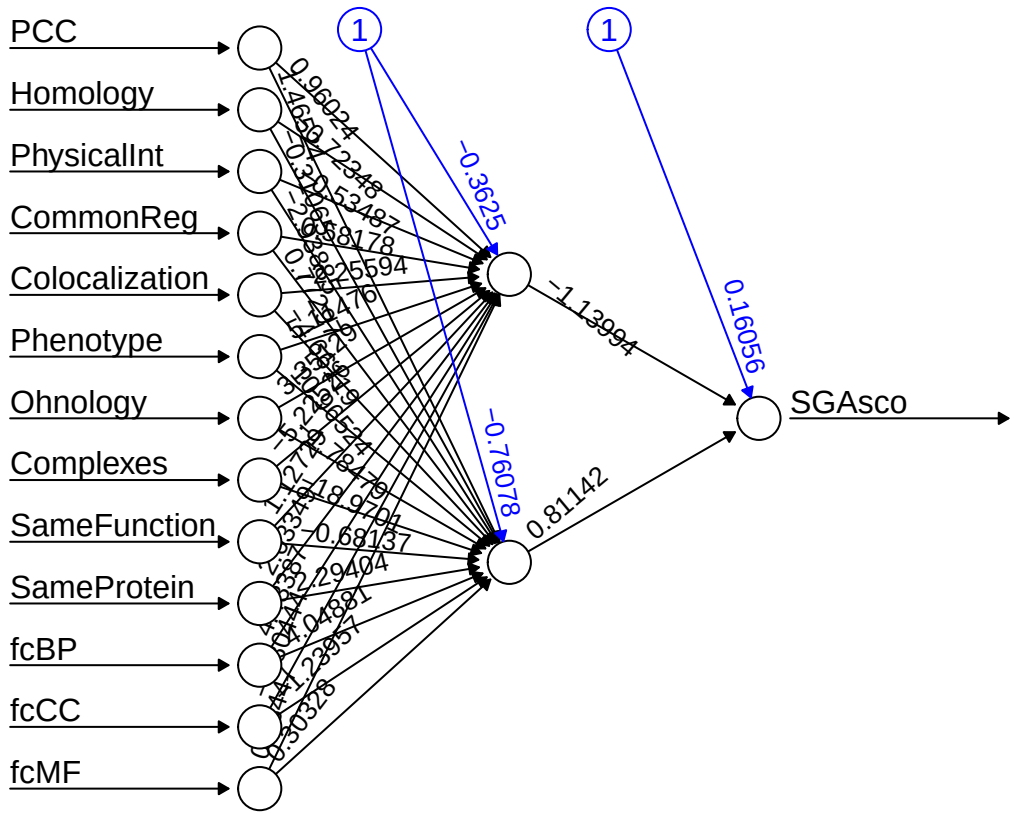
```
[1] 0.2159961
```

Mejorar el rendimiento del modelo

Podemos intentar mejorar el rendimiento incrementando el número de *hidden nodes*.

2 *Hidden nodes*

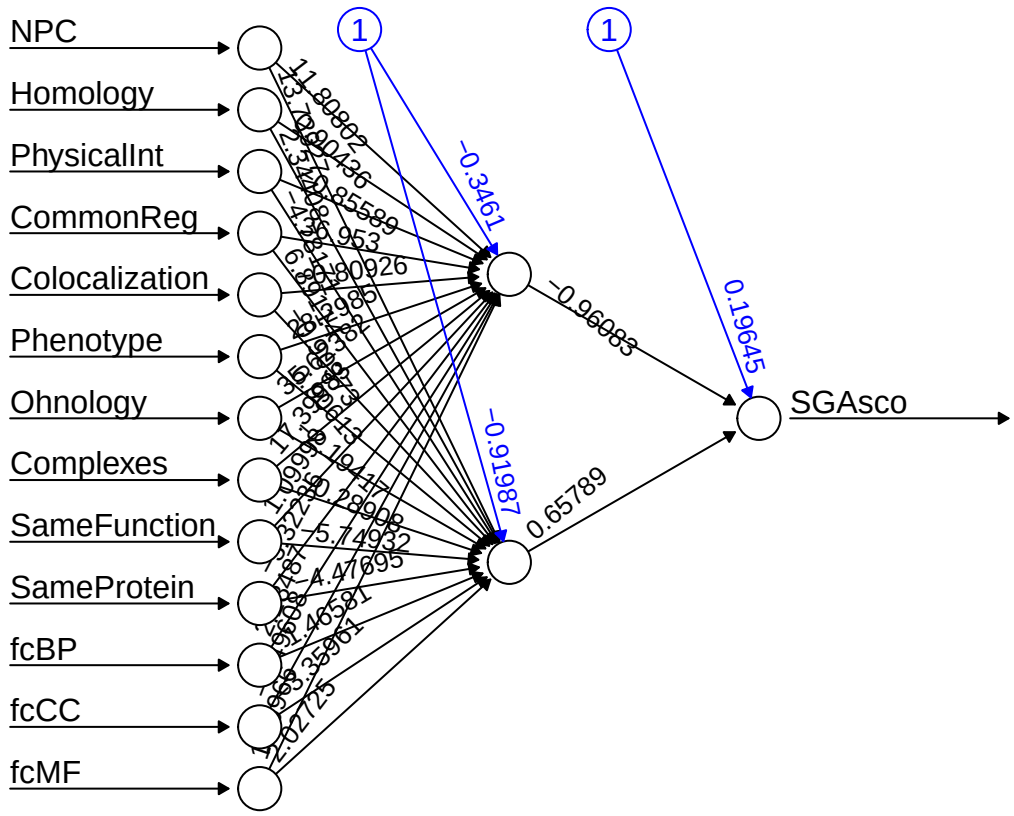
```
set.seed(seed)
ann_model_2_PCC <- neuralnet(SGAsco ~. -NPC, data=SGA10_train, hidden = 2)
plot(ann_model_2_PCC, rep="best")
```




```
ann_results_2_PCC <- compute(ann_model_2_PCC, SGA10_test)
ann_pred_2_PCC <- ann_results_2_PCC$net.result
(tabla_ann[2,2] <- RMSE(ann_pred_2_PCC, SGA10_test$SGAsco))
```

```
[1] 0.1745996
```

```
set.seed(seed)
ann_model_2_NPC <- neuralnet(SGAsco ~. -PCC, data=SGA10_train, hidden = 2)
plot(ann_model_2_NPC, rep="best")
```

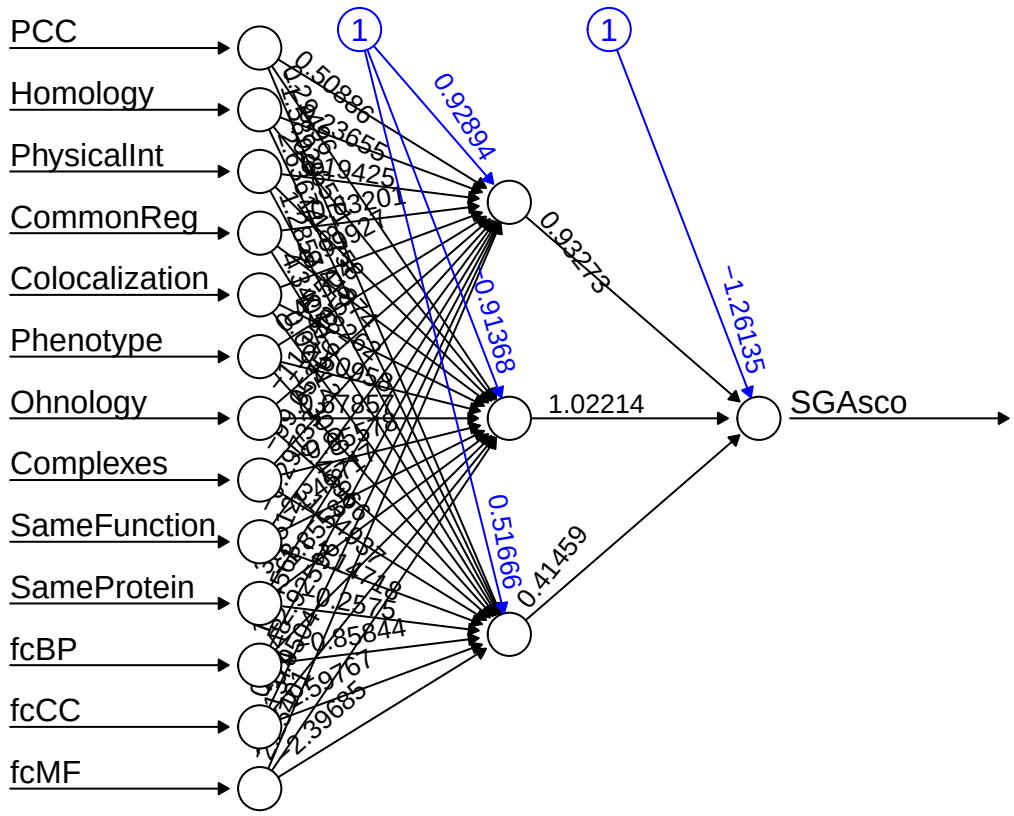



```
ann_results_2_NPC <- compute(ann_model_2_NPC, SGA10_test)
ann_pred_2_NPC <- ann_results_2_NPC$net.result
(tabla_ann[2,3] <- RMSE(ann_pred_2_NPC, SGA10_test$SGAsco))
```

```
[1] 0.1867948
```

3 ***Hidden nodes***

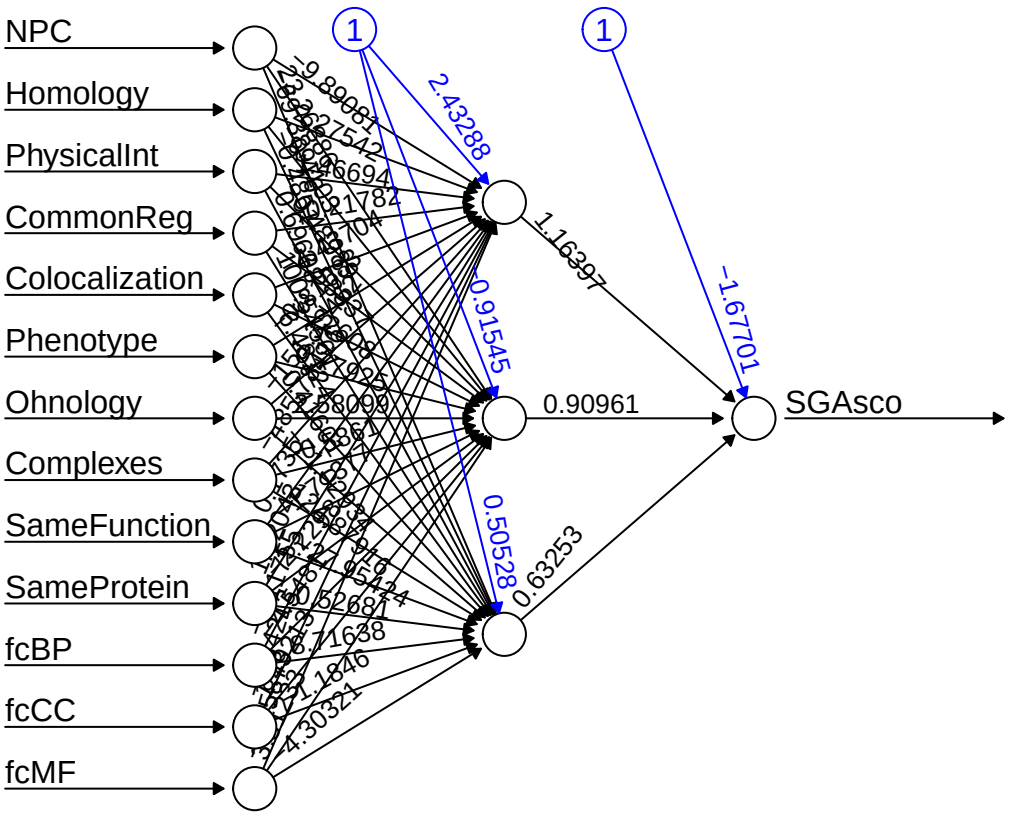
```
set.seed(seed)
ann_model_3_PCC <- neuralnet(SGAsco ~. -NPC, data=SGA10_train, hidden = 3)
plot(ann_model_3_PCC, rep="best")
```




```
ann_results_3_PCC <- compute(ann_model_3_PCC, SGA10_test)
ann_pred_3_PCC <- ann_results_3_PCC$net.result
(tabla_ann[3,2] <- RMSE(ann_pred_3_PCC, SGA10_test$SGAsco))
```

```
[1] 0.1942752
```

```
set.seed(seed)
ann_model_3_NPC <- neuralnet(SGAsco ~. -PCC, data=SGA10_train, hidden = 3)
plot(ann_model_3_NPC, rep="best")
```




```
ann_results_3_NPC <- compute(ann_model_3_NPC, SGA10_test)
ann_pred_3_NPC <- ann_results_3_NPC$net.result
(tabla_ann[3,3] <- RMSE(ann_pred_3_NPC, SGA10_test$SGAsco))
```

```
[1] 0.2170448
```

Resumen resultados algoritmo Artificial Neural Networks

```
kable(tabla_ann) %>%
  kable_styling(full_width = F, position = "left")
```

hidden	PCC	NPC
1	0.2053081	0.2159961
2	0.1745996	0.1867948
3	0.1942752	0.2170448

Algoritmo Support Vector Machine

Entrenamiento del modelo

Empezamos entrenando el modelo con el *kernel* lineal (`vanilladot`)

```
set.seed(seed)
svm_model_PCC_1 <- ksvm(SGAsco ~. -NPC, data=SGA10_train, kernel = "vanilladot")
```

```
 Setting default kernel parameters
```

```
set.seed(seed)
svm_model_NPC_1 <- ksvm(SGAsco ~. -PCC, data=SGA10_train, kernel = "vanilladot")
```

```
 Setting default kernel parameters
```

Evaluar el rendimiento del modelo

```
tabla_svm <- data.frame(kernel = c("vanilladot", "rbfdot", "laplacedot" ), PCC = NA, NPC = NA)

svm_pred_PCC_1 <- predict(svm_model_PCC_1, SGA10_test)
(tabla_svm[1,2] <- RMSE(svm_pred_PCC_1, SGA10_test$SGAsco))
```

```
[1] 0.1653479
```

```
svm_pred_NPC_1 <- predict(svm_model_NPC_1, SGA10_test)
(tabla_svm[1,3] <- RMSE(svm_pred_NPC_1, SGA10_test$SGAsco))
```

```
[1] 0.1654833
```

Mejorar el rendimiento del modelo

Podemos intentar mejorar el rendimiento cambiando el tipo de *kernel* que utiliza el modelo.

Con el *kernel* `rbfdot` (*Radial Basis kernel "Gaussian"*)

```
set.seed(seed)
svm_model_PCC_2 <- ksvm(SGAsco ~. -NPC, data=SGA10_train, kernel = "rbfdot")
svm_pred_PCC_2 <- predict(svm_model_PCC_2, SGA10_test)
(tabla_svm[2,2] <- RMSE(svm_pred_PCC_2, SGA10_test$SGAsco))
```

```
[1] 0.1832892
```

```
set.seed(seed)
svm_model_NPC_2 <- ksvm(SGAsco ~. -PCC, data=SGA10_train, kernel = "rbfdot")
svm_pred_NPC_2 <- predict(svm_model_NPC_2, SGA10_test)
(tabla_svm[2,3] <- RMSE(svm_pred_NPC_2, SGA10_test$SGAsco))
```

```
[1] 0.1861551
```

Con el *kernel* `laplacedot` (*Laplacian kernel*)

```
set.seed(seed)
svm_model_PCC_3 <- ksvm(SGAsco~. -NPC, data=SGA10_train, kernel = "laplacedot")
svm_pred_PCC_3 <- predict(svm_model_PCC_3, SGA10_test)
(tabla_svm[3,2] <- RMSE(svm_pred_PCC_3, SGA10_test$SGAsco))
```

```
[1] 0.1691735
```

```
set.seed(seed)
svm_model_NPC_3 <- ksvm(SGAsco ~. -PCC, data=SGA10_train, kernel = "laplacedot")
svm_pred_NPC_3 <- predict(svm_model_NPC_3, SGA10_test)
(tabla_svm[3,3] <- RMSE(svm_pred_NPC_3, SGA10_test$SGAsco))
```

```
[1] 0.1790633
```

Resumen resultados algoritmo SVM:

```
kable(tabla_svm) %>%
  kable_styling(full_width = F, position = "left")
```

kernel	PCC	NPC
vanilladot	0.1653479	0.1654833
rbfdot	0.1832892	0.1861551
laplacedot	0.1691735	0.1790633

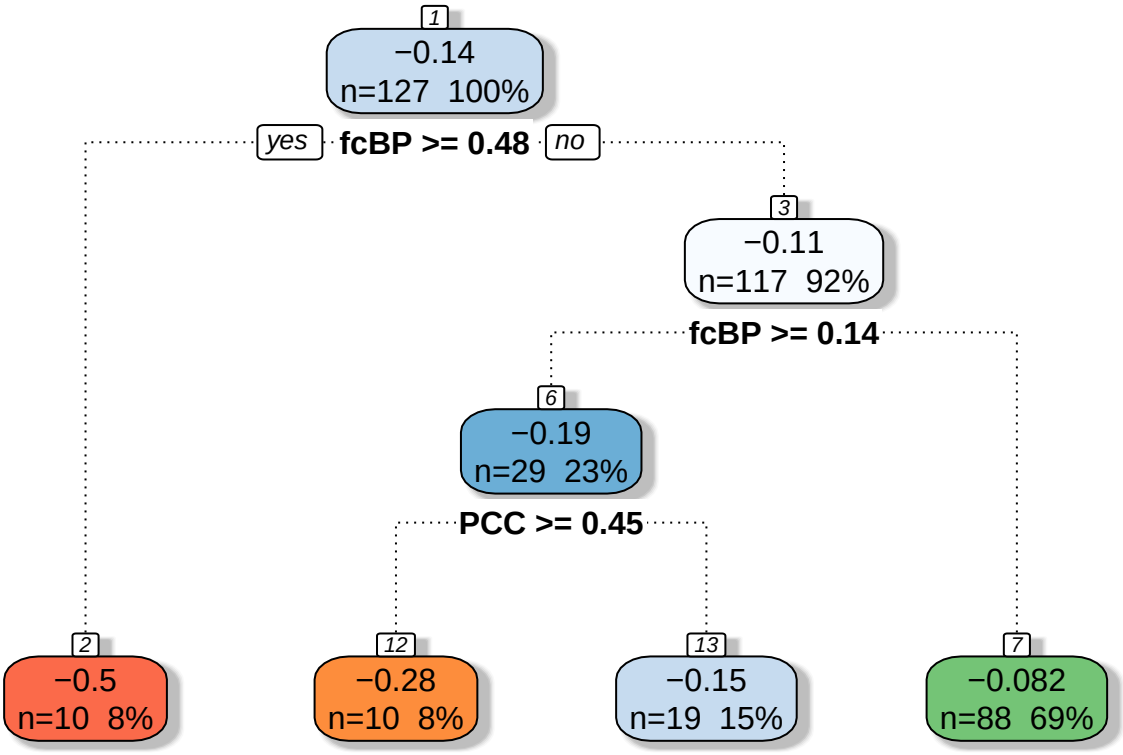
Algoritmo Árbol de Decisión

Entrenamiento del Modelo

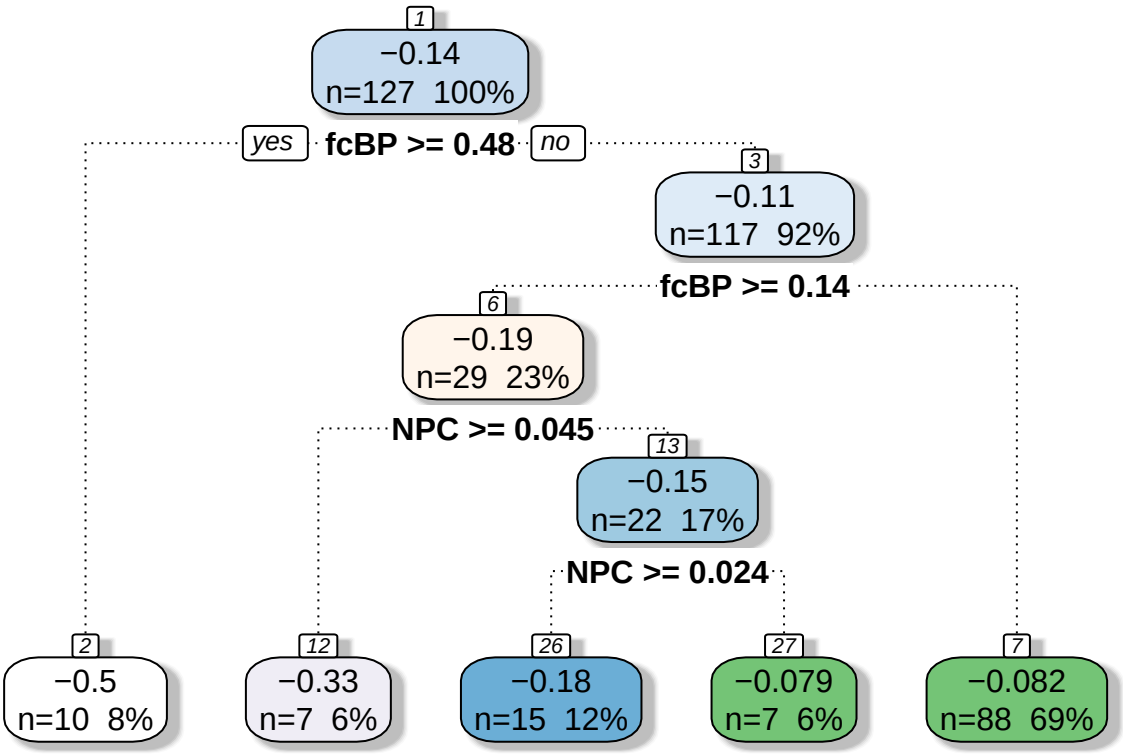
```
set.seed(seed)
ad_model_PCC_1 <- rpart(SGAsco ~. -NPC, data=SGA10_train,)

set.seed(seed)
ad_model_NPC_1 <- rpart(SGAsco ~. -PCC, data=SGA10_train)
```

Mostramos un gráfico del modelo

```
fancyRpartPlot(ad_model_PCC_1, caption = NULL)
```




```
fancyRpartPlot(ad_model_NPC_1, caption = NULL)
```



Evaluar el rendimiento del modelo

```
tabla_ad <- data.frame(Option = c("Default", "Param"), PCC = NA, NPC = NA)

ad_pred_PCC_1 <- predict(ad_model_PCC_1, SGA10_test)
(tabla_ad[1,2] <- RMSE(ad_pred_PCC_1, SGA10_test$SGAsco))
```

```
[1] 0.165828
```

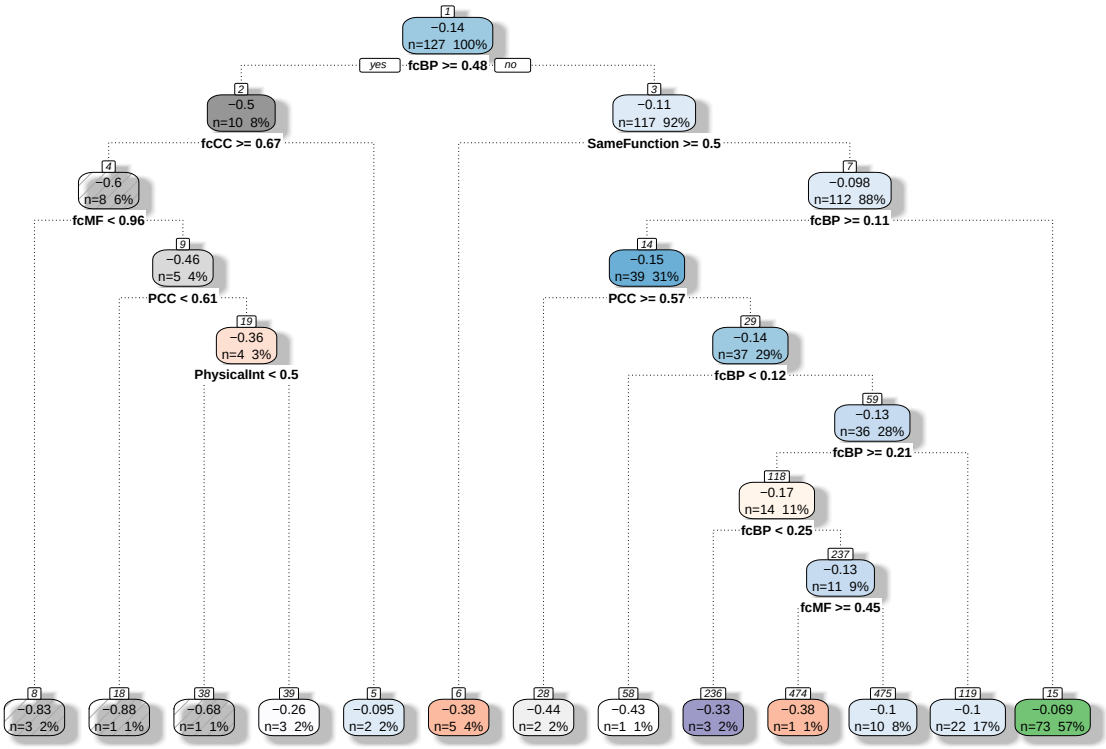
```
ad_pred_NPC_1 <- predict(ad_model_NPC_1, SGA10_test)
(tabla_ad[1,3] <- RMSE(ad_pred_NPC_1, SGA10_test$SGAsco))
```

```
[1] 0.1667751
```

Mejorar el rendimiento del modelo

Podemos probar a cambiar algún parámetro por defecto para comprobar si mejora el rendimiento del modelo

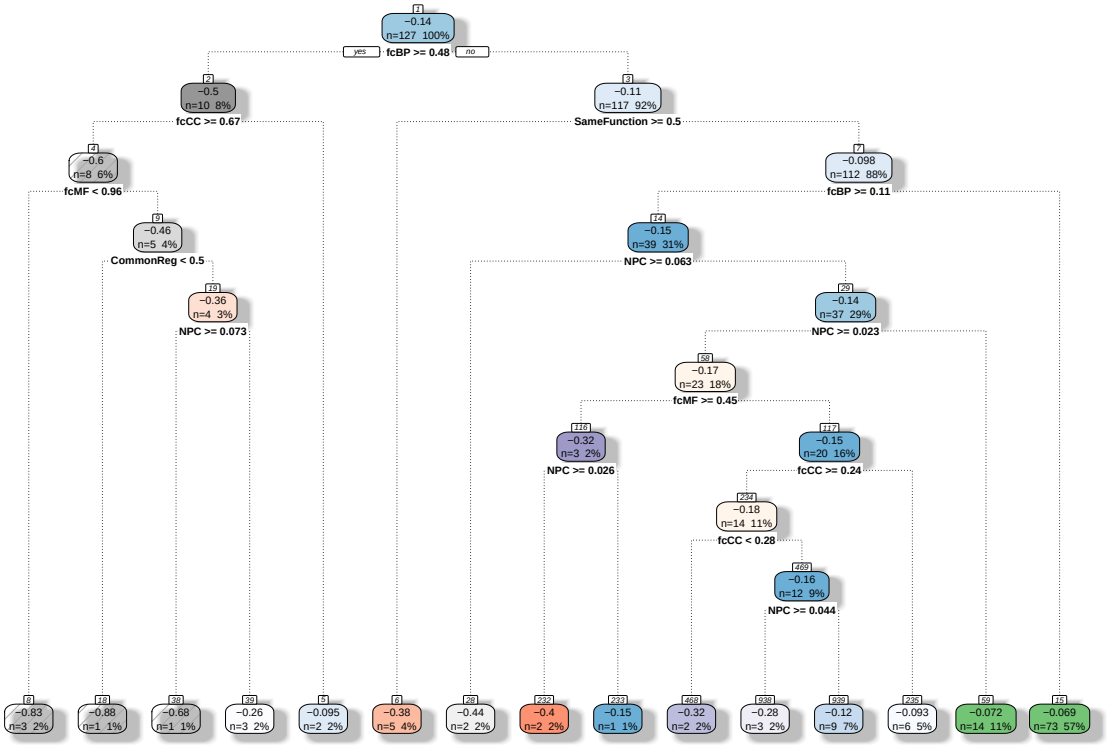
```
set.seed(seed)
ad_model_PCC_2 <- rpart(SGAsco ~. -NPC, data=SGA10_train,
                        minsplit = 2, minbucket = 1)
fancyRpartPlot(ad_model_PCC_2, caption = NULL)
```




```
ad_pred_PCC_2 <- predict(ad_model_PCC_2, SGA10_test)
(tabla_ad[2,2] <- RMSE(ad_pred_PCC_2, SGA10_test$SGAsco))
```

```
[1] 0.1418558
```

```
set.seed(seed)
ad_model_NPC_2 <- rpart(SGAsco ~. -PCC, data=SGA10_train,
                        minsplit = 2, minbucket = 1)
fancyRpartPlot(ad_model_NPC_2, caption = NULL)
```




```
ad_pred_NPC_2 <- predict(ad_model_NPC_2, SGA10_test)
(tabla_ad[2,3] <- RMSE(ad_pred_NPC_2, SGA10_test$SGAsco))
```

```
[1] 0.1376482
```

Resumen resultados algoritmo Arbol de decisión

```
kable(tabla_ad) %>%
  kable_styling(full_width = F, position = "left")
```

Option	PCC	NPC
Default	0.1658280	0.1667751
Param	0.1418558	0.1376482

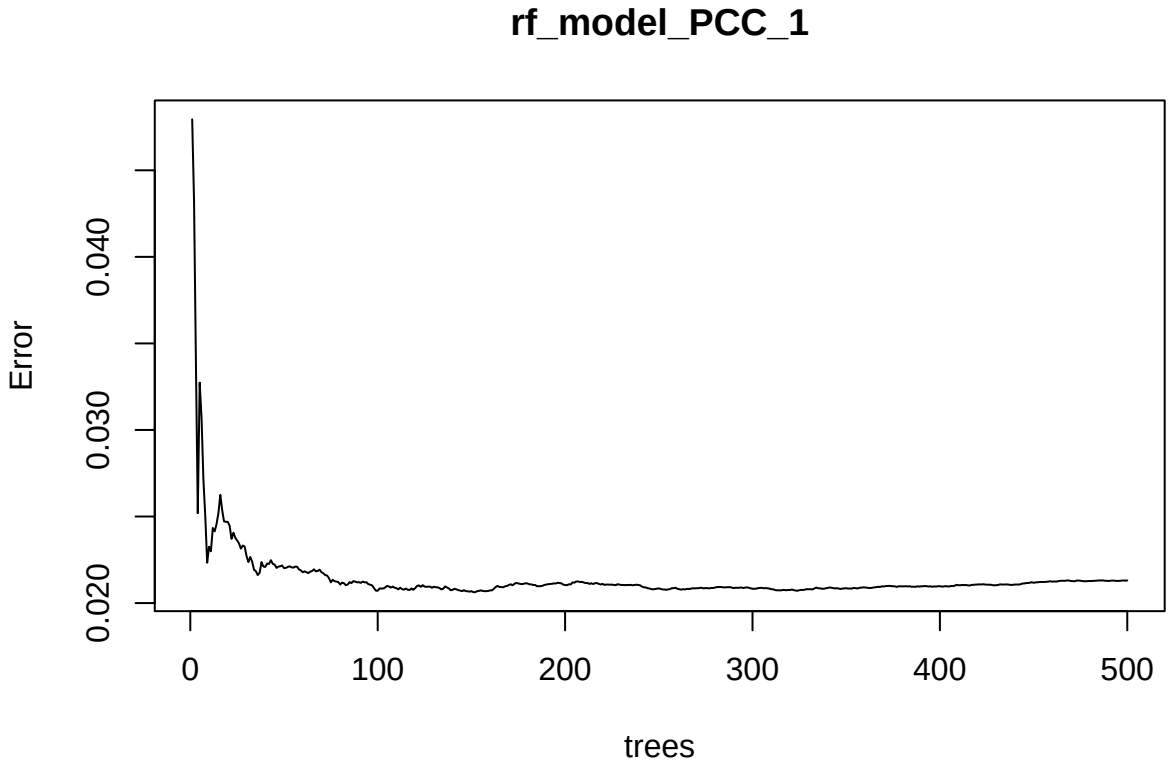
Algoritmo Random Forest

Entrenamiento del Modelo

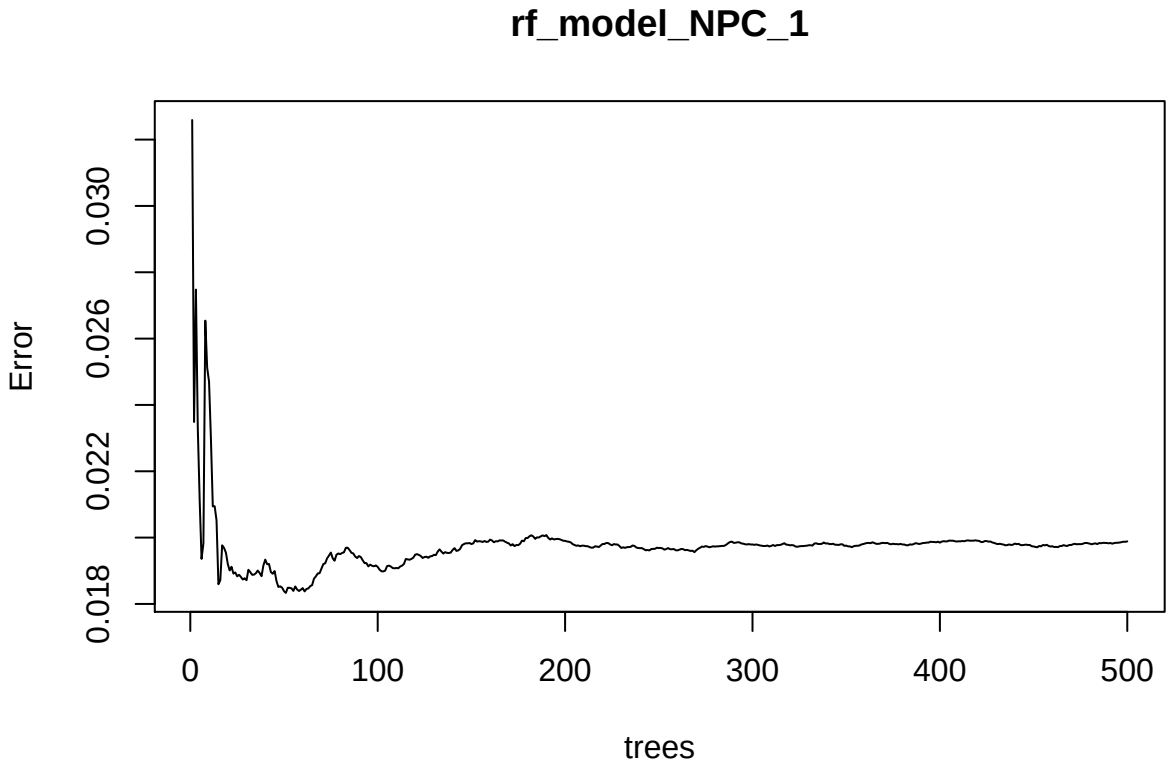
```
set.seed(seed)
rf_model_PCC_1<- randomForest(SGAsco ~. -NPC, data=SGA10_train)

set.seed(seed)
rf_model_NPC_1<- randomForest(SGAsco ~. -PCC, data=SGA10_train)
```

Mostramos un gráfico del modelo

```
plot(rf_model_PCC_1)
```



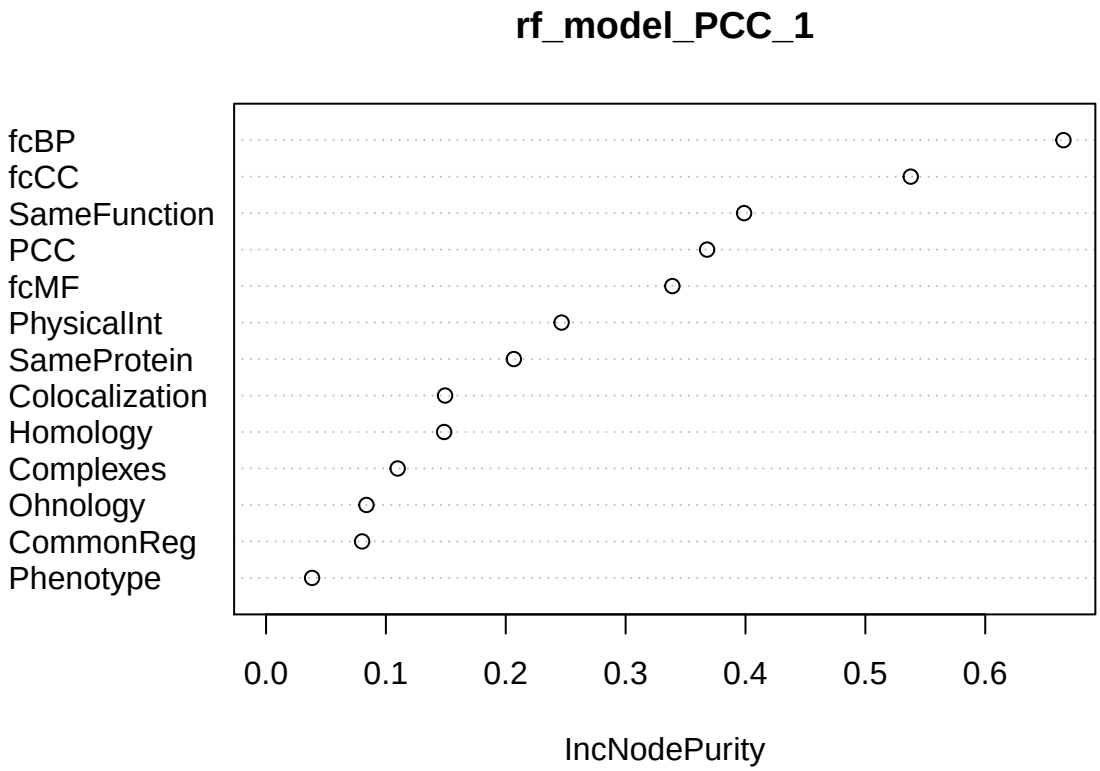

```
plot(rf_model_NPC_1)
```



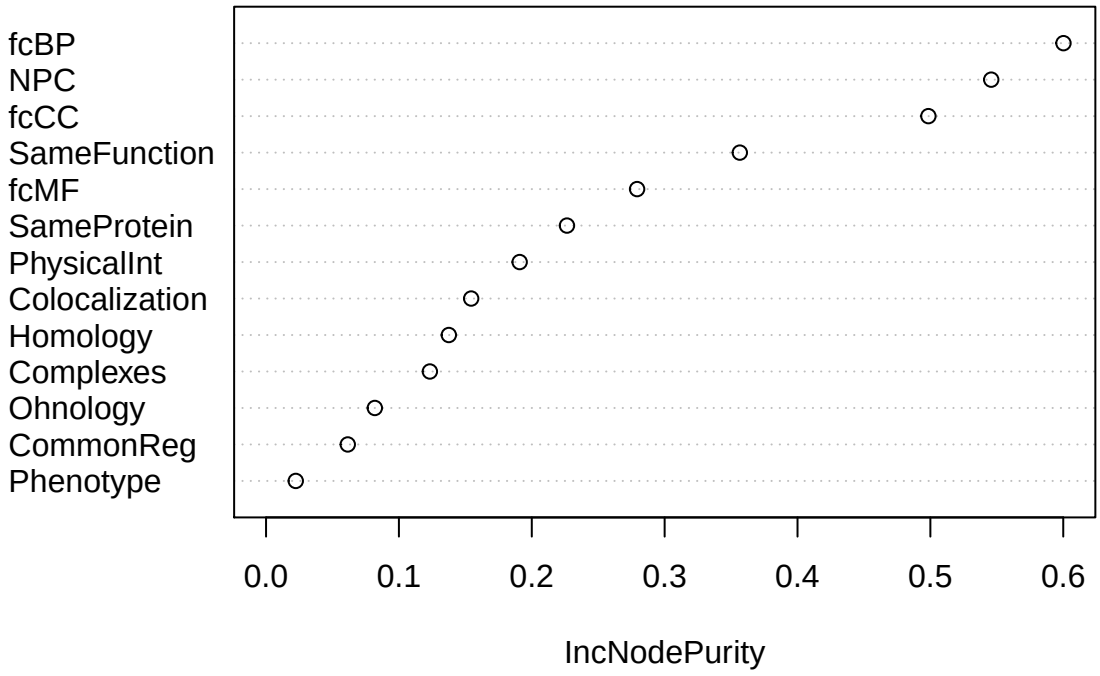
Observamos que el comportamiento es bastante estable a partir de 100 árboles aproximádamente.

Visualizamos la importancia de las variables

```
varImpPlot(rf_model_PCC_1)
```




```
varImpPlot(rf_model_NPC_1)
```



Observamos por ejemplo que la variable *NPC* gana importancia respecto *PCC*.

Evaluar el rendimiento del modelo

```
tabla_rf <- data.frame(Option = c("Default", "Param"), PCC = NA, NPC = NA)

rf_pred_PCC_1 <- predict(rf_model_PCC_1, SGA10_test)
(tabla_rf[1,2] <- RMSE(rf_pred_PCC_1, SGA10_test$SGAsco))
```

```
[1] 0.1420198
```

```
rf_pred_NPC_1 <- predict(rf_model_NPC_1, SGA10_test)
(tabla_rf[1,3] <- RMSE(rf_pred_NPC_1, SGA10_test$SGAsco))
```

```
[1] 0.1431044
```

Mejorar el rendimiento del modelo

Podemos intentar mejorar el rendimiento incrementando el número de árboles.

```
set.seed(seed)
rf_model_PCC_1000 <- randomForest(SGAsco ~. -NPC, data=SGA10_train, ntree=1000)
rf_pred_PCC_1000 <- predict(rf_model_PCC_1000, SGA10_test)
(tabla_rf[2,2] <- RMSE(rf_pred_PCC_1000, SGA10_test$SGAsco))
```

```
[1] 0.1432169
```

```
set.seed(seed)
rf_model_NPC_1000 <- randomForest(SGAsco ~. -PCC, data=SGA10_train, ntree=1000)
rf_pred_NPC_1000 <- predict(rf_model_NPC_1000, SGA10_test)
(tabla_rf[2,3] <- RMSE(rf_pred_NPC_1000, SGA10_test$SGAsco))
```

```
[1] 0.1425534
```

Vemos que el modelo no ha mejorado aumentando el número de árboles del bosque a 1.000, como se intuía en el gráfico anterior.

Resumen resultados algoritmo Random Forest

```
kable(tabla_rf) %>%
  kable_styling(full_width = F, position = "left")
```

Option	PCC	NPC
Default	0.1420198	0.1431044
Param	0.1432169	0.1425534

Recapitulando el resumen de todos los modelos

Regresión lineal

```
RMSE.lm1
```

```
[1] 0.1226716
```

Algoritmo k-nn

```
kable(tabla_knn) %>%
  kable_styling(full_width = F, position = "left")
```

Option	PCC	NPC
Default	0.1465124	0.1417423
Param	0.1488071	0.1522432

Algoritmo Artificial Neural Networks

```
kable(tabla_ann) %>%
  kable_styling(full_width = F, position = "left")
```

hidden	PCC	NPC
1	0.2053081	0.2159961
2	0.1745996	0.1867948
3	0.1942752	0.2170448

Algoritmo Support Vector Machine

```
kable(tabla_svm) %>%
  kable_styling(full_width = F, position = "left")
```

kernel	PCC	NPC
vanilladot	0.1653479	0.1654833
rbfdot	0.1832892	0.1861551
laplacedot	0.1691735	0.1790633

Algoritmo Árbol de decisión

```
kable(tabla_ad) %>%
  kable_styling(full_width = F, position = "left")
```

Option	PCC	NPC
Default	0.1658280	0.1667751
Param	0.1418558	0.1376482

Algoritmo Random Forest

```
kable(tabla_rf) %>%
  kable_styling(full_width = F, position = "left")
```

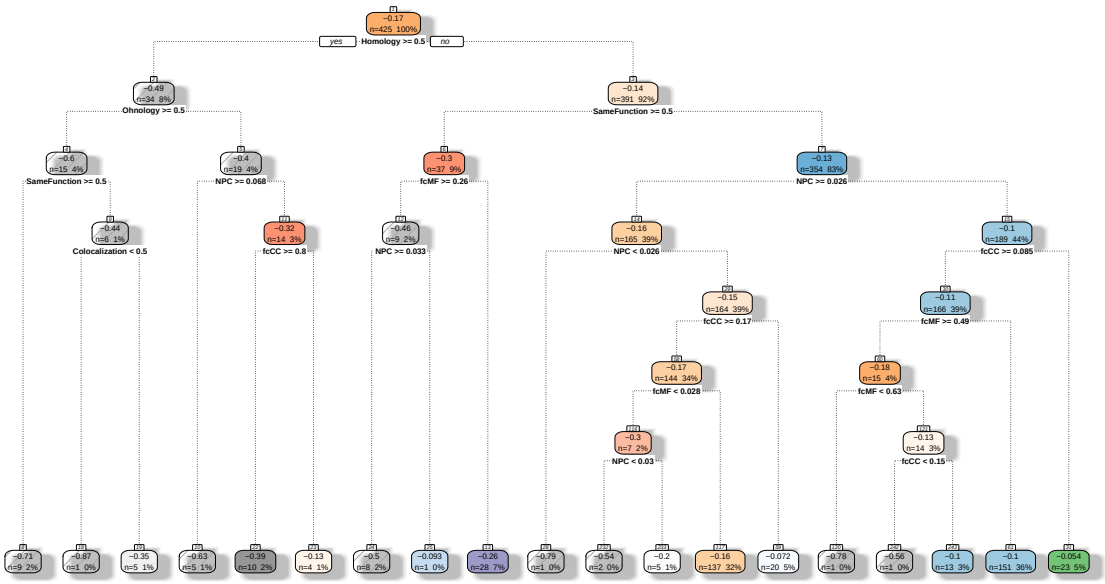
Option	PCC	NPC
Default	0.1420198	0.1431044
Param	0.1432169	0.1425534

Vemos que ninguno mejora el valor del modelo de regresión lineal.

El que ha obtenido el mejor valor es el algoritmo Arbol de Decisión, parametrizado, y con la variable NPC.

Podemos realizar los cálculos con el dataset SGA16 y esos parámetros:

```
set.seed(seed)
ad_model_NPC_16 <- rpart(SGAsco ~. -PCC, data=SGA16_train,
                         minsplit = 2, minbucket = 1)
fancyRpartPlot(ad_model_NPC_16, caption = NULL)
```




```
ad_pred_NPC_16 <- predict(ad_model_NPC_16, SGA16_test)
RMSE(ad_pred_NPC_16, SGA16_test$SGAsco)
```

```
[1] 0.1561926
```

Vemos que aunque incrementemos el número de observaciones, la predicción no mejora.