PEC 3:Determinación de la localización subcelular de proteínas

Cristina Lendinez Gonzalez

17 de junio, 2021

Contents

```
library(mltools)
library(data.table)
library(class)
library(gmodels)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(lattice)
library(ggplot2)
library(knitr)
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:mltools':
##
##     skewness
```

```
library(neuralnet)
library(NeuralNetTools)
#library(kernalab)
library(C50)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(mltools)
```

1 Lectura de los datos, exploración, transformación y obtención de las muestras train y test

Los datos que se van a utilizar en esta PEC vienen adjuntos al enunciado y han sido descargados directamente desde la pagina de la UOC en el aula de Machine Learning,

```
datos<-read.table("./yeast.data")
```

Voy a veer que tipo de variables tengo en mi dataset llamado data

```
str(datos)
```

```
## 'data.frame':    1484 obs. of  10 variables:
##  $ V1 : chr  "ADT1_YEAST" "ADT2_YEAST" "ADT3_YEAST" "AAR2_YEAST" ...
##  $ V2 : num  0.58 0.43 0.64 0.58 0.42 0.51 0.5 0.48 0.55 0.4 ...
##  $ V3 : num  0.61 0.67 0.62 0.44 0.44 0.4 0.54 0.45 0.5 0.39 ...
##  $ V4 : num  0.47 0.48 0.49 0.57 0.48 0.56 0.48 0.59 0.66 0.6 ...
##  $ V5 : num  0.13 0.27 0.15 0.13 0.54 0.17 0.65 0.2 0.36 0.15 ...
##  $ V6 : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ V7 : num  0 0 0 0 0 0.5 0 0 0 0 ...
##  $ V8 : num  0.48 0.53 0.53 0.54 0.48 0.49 0.53 0.58 0.49 0.58 ...
##  $ V9 : num  0.22 0.22 0.22 0.22 0.22 0.22 0.22 0.34 0.22 0.3 ...
##  $ V10: chr  "MIT" "MIT" "MIT" "NUC" ...
```

```
colnames(datos)<-c("secuencia","mcg","gvh", "alm","mit","erl","pox","vac","nuc","class")
```

Hago una tabla de frecuencias para ver la localizacion en la celula.

```
table(datos$class)
```

```
##
## CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
## 463   5  35  44  51 163 244 429  20  30
```

En el enunciado nos piden que englobe en una univa clase Mem los tipos (MEM1, MEM2, MEM3).

```
datos$class <- as.character(datos$class)
datos$class[datos$class == "ME1"] <- "MEM"
datos$class[datos$class == "ME2"] <- "MEM"
datos$class[datos$class == "ME3"] <- "MEM"
```

Lo que voy a hacer es crear el dataset con el que voy a trabajar.

```
data <- subset(datos, subset = class == "CYT" | class == "MEM" | class == "MIT" | class == "NUC")
```

Puedo ver que la primera variable la variable secuencia no me sirve, ya que es una varable explicativa.

```
data <- data[-1]
```

Puedo ver que ya me he quedado solo con las variables numericas, las cuales voy a usar para hacer el anaisis. , empezare haciendo un summary del dataframe llamado **dataframe**. y un table para ver como se dividen las varibles class y erc.

```
table(data$class)
```

```
##
## CYT MEM MIT NUC
## 463 258 244 429
```

```
table(data$erl)
```

```
##
##  0.5    1
## 1385    9
```

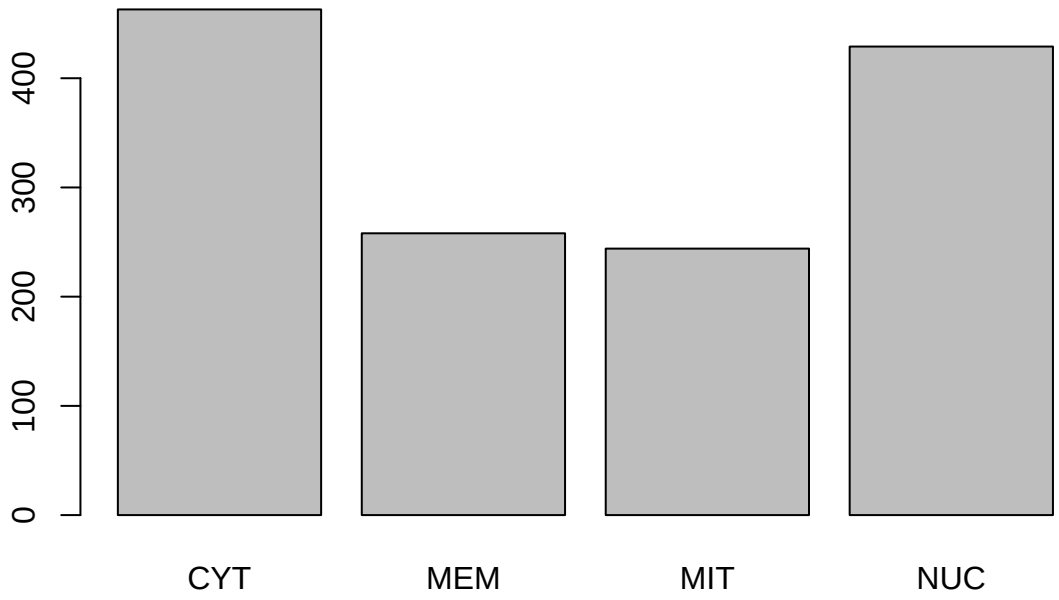
Hago el estadistico básico con la variable summary

```
summary(data)
```

```
##       mcg              gvh              alm              mit
##  Min.   :0.1100   Min.   :0.1300   Min.   :0.2100   Min.   :0.0000
##  1st Qu.:0.4000   1st Qu.:0.4200   1st Qu.:0.4600   1st Qu.:0.1700
##  Median :0.4800   Median :0.4800   Median :0.5100   Median :0.2200
##  Mean   :0.4918   Mean   :0.4928   Mean   :0.5009   Mean   :0.2616
##  3rd Qu.:0.5700   3rd Qu.:0.5600   3rd Qu.:0.5600   3rd Qu.:0.3200
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       erl              pox               vac              nuc
##  Min.   :0.5000   Min.   :0.000000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5000   1st Qu.:0.000000   1st Qu.:0.4800   1st Qu.:0.2200
##  Median :0.5000   Median :0.000000   Median :0.5100   Median :0.2200
##  Mean   :0.5032   Mean   :0.001908   Mean   :0.5002   Mean   :0.2787
##  3rd Qu.:0.5000   3rd Qu.:0.000000   3rd Qu.:0.5300   3rd Qu.:0.3100
##  Max.   :1.0000   Max.   :0.830000   Max.   :0.7300   Max.   :1.0000
##     class
##  Length:1394
##  Class :character
##  Mode  :character
##
##
##
```

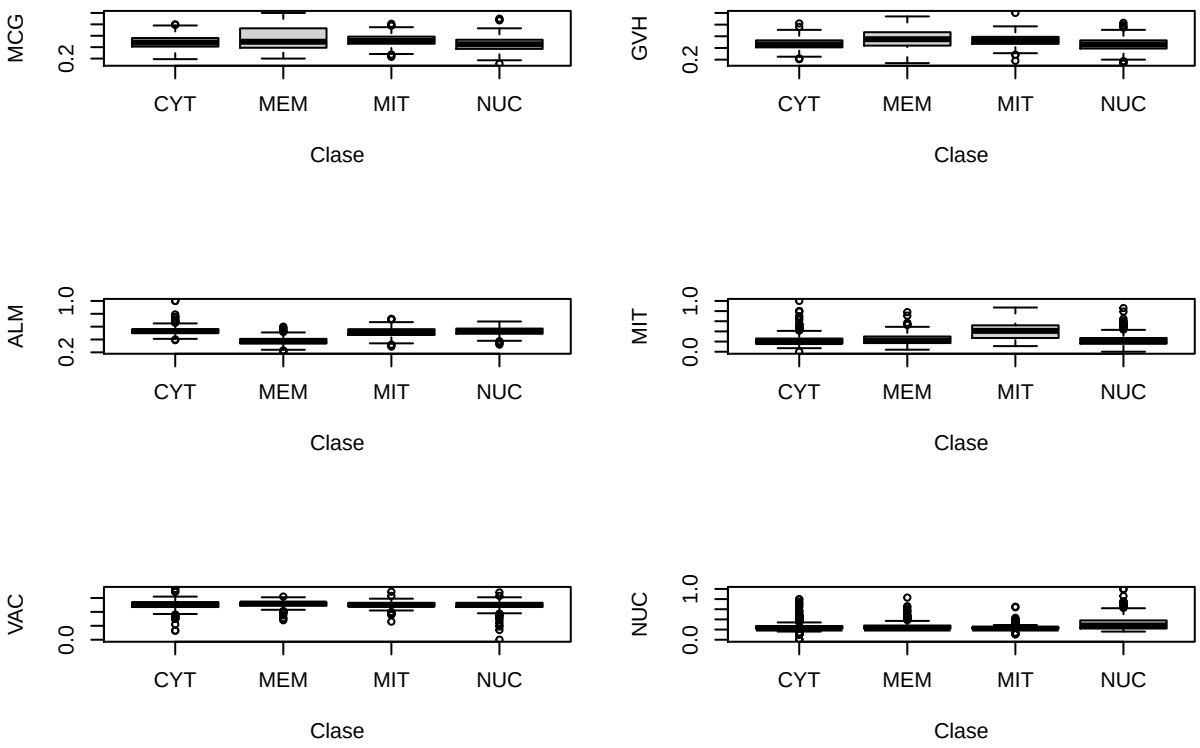
Voy a graficar la variable class, asi podre ver como se distribuyen en un grafico de barras los diferentes tipos de clases.

```
barplot(table(data$class))
```



voy a comparar los diferentes graficos con todas las variables en graficos de cajas.

```
par(mfrow=c(3,2))
boxplot(data$mcg~data$class, xlab="Clase", ylab="MCG")
boxplot(data$gvh~data$class, xlab="Clase", ylab="GVH")
boxplot(data$alm~data$class, xlab="Clase", ylab="ALM")
boxplot(data$mit~data$class, xlab="Clase", ylab="MIT")
boxplot(data$vac~data$class, xlab="Clase", ylab="VAC")
boxplot(data$nuc~data$class, xlab="Clase", ylab="NUC")
```



Como puedo ver todos los valores que tengo oscilan entre 0 y 1 y por eso no tenemos que hacer el one-hote encodign o dummy ya que no hay que normalizar los valores.

1.1 Obtención muestras train y test

Voy a eliminar la variable class, ya que es una variable categorica.

```
data_2 <- as.data.frame(data[-9])
```

veo que observciones y variables tengo, mirando las primeras 5 observaciones

```
head(data_2)
```

```
##    mcg  gvh  alm  mit erl pox  vac  nuc
## 1 0.58 0.61 0.47 0.13 0.5 0.0 0.48 0.22
## 2 0.43 0.67 0.48 0.27 0.5 0.0 0.53 0.22
## 3 0.64 0.62 0.49 0.15 0.5 0.0 0.53 0.22
## 4 0.58 0.44 0.57 0.13 0.5 0.0 0.54 0.22
## 5 0.42 0.44 0.48 0.54 0.5 0.0 0.48 0.22
## 6 0.51 0.40 0.56 0.17 0.5 0.5 0.49 0.22
```

Vamos a generar la parte de training y la parte de test, hago una separacion del 67% y del 33%

```
set.seed(1234)
train<-sample(1:nrow(data_2),round(2*nrow(data_2)/3))
out_training<-data_2[train,]
out_test<-data_2[-train,]
dim(out_training)
```

```
## [1] 929   8
```

```
dim(out_test)
```

```
## [1] 465   8
```

```
#labels
class_training <- data[train,9]
class_test <- data[-train,9]
```

2 Elaboración de los algoritmos

Vamos a analizar la capacidad de predecir los algoritmos que hemos aprendido a lo largo del curso

2.1 Algoritmo K-NN

Vamos a entrenar el algoritmo **KNN** para ver que valores obtengo en estos k(1,3,5,7,11), son los mismos k que usamos en la pec1.

```
ks<-c(1,3,5,7,11)
kNN_all<-data.frame(ks,Accuracy=NA, Kappa=NA, AccuracyLower=NA, AccuracyUpper=NA)

j<-0
for(i in ks){
j<-j+1
set.seed(1234)
prediction<-knn(train=out_training,test=out_test,cl=class_training,k=i)
conf.mat.kNN<-confusionMatrix(table(prediction,class_test))
kNN_all[j,2:5]<-round(conf.mat.kNN$overall[1:4],3)
}
kable(kNN_all,align=c("l","c","c","c","c"),caption=paste("Algoritmo kNN"))
```

Table 1: Algoritmo kNN

ks	Accuracy	Kappa	AccuracyLower	AccuracyUpper
1	0.578	0.422	0.532	0.624
3	0.596	0.447	0.550	0.641
5	0.602	0.456	0.556	0.647
7	0.604	0.457	0.558	0.649
11	0.617	0.475	0.571	0.662

Puedo ver que obtengo unos valores con los diferentes k(1,3,5,7,11), en el que veo que el que mejor precisión tiene es el k11 con un Accuracy de 0.617 .

Voy a generar la matriz de confusión con el k11, ya que es el que mejor valor predictivo me acaba de dar.

```
#con la mejor k
test_predicion<-knn(train=out_training,test=out_test,cl=class_training,k=11)
confusionMatrix(table(class_test,test_predicion))
```

```
## Confusion Matrix and Statistics
##
##           test_predicion
## class_test CYT MEM MIT NUC
##        CYT  94   2  10  51
##        MEM   8  67   2   9
##        MIT  20   6  57   9
##        NUC  54   5   8  63
##
## Overall Statistics
##
##                Accuracy : 0.6043
##                  95% CI : (0.5582, 0.649)
##     No Information Rate : 0.3785
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4567
##
##  Mcnemar's Test P-Value : 0.1157
##
## Statistics by Class:
##
##                      Class: CYT Class: MEM Class: MIT Class: NUC
## Sensitivity              0.5341     0.8375     0.7403     0.4773
## Specificity              0.7820     0.9506     0.9098     0.7988
## Pos Pred Value           0.5987     0.7791     0.6196     0.4846
## Neg Pred Value           0.7338     0.9657     0.9464     0.7940
## Prevalence               0.3785     0.1720     0.1656     0.2839
## Detection Rate           0.2022     0.1441     0.1226     0.1355
## Detection Prevalence     0.3376     0.1849     0.1978     0.2796
## Balanced Accuracy        0.6580     0.8941     0.8250     0.6380
```

2.2 Algoritmo Naive Bayes

Este algoritmo esta basado en el teorema de Bayes.En este algoritmo utilizaré los datos originales quese han utilizado en el caso anterior de los **KNN**. No tenemos que transformar las variables. Entrenare el modelo con laplace=0 y laplace=1

2.2.1 Entrenamiento del model de Naive Bayes

```
set.seed(1234)
NB_0<-naiveBayes(out_training,class_training,type="raw",laplace=0)
NB_1<-naiveBayes(out_training,class_training,type="raw",laplace = 1)
```

2.2.2 Prediciion y evaluacion del modelo Naive Bayes

```
#predicción y evaluación del modelo
predNB_0<-predict(NB_0,out_test,type="class")
predNB_1<-predict(NB_1,out_test,type="class")
evalNB_0<-confusionMatrix(table(predNB_0,class_test))
evalNB_1<-confusionMatrix(table(predNB_1,class_test))
```

Los datos que obtenemos son estos:

```
lp<-data.frame(laplace=c(0,1))
NB_all<-rbind(round(evalNB_0$overall[1:4],3),round(evalNB_1$overall[1:4],3))
NB_all<-cbind(lp,NB_all)
kable(NB_all,align=c("l","c","c","c","c"),caption=paste("Algoritmo Naive Bayes"))
```

Table 2: Algoritmo Naive Bayes

laplace	Accuracy	Kappa	AccuracyLower	AccuracyUpper
0	0.535	0.39	0.489	0.582
1	0.535	0.39	0.489	0.582

##Algoritmo Neural Networks

2.2.3 Transformar los datos

Como tengo que entrenar el modelo del ANN o Algoritmo Neural Networks, tengo que crear unas nuevas variables para poder ponerle nobre a la variable clase.

```
data_ANN<-data[,-9]
data_ANN$CYT<-data$class=="CYT"
data_ANN$MEM<-data$class=="MEM"
data_ANN$MIT<-data$class=="MIT"
data_ANN$NUC<-data$class=="NUC"
names(data_ANN)
```

```
##  [1] "mcg" "gvh" "alm" "mit" "erl" "pox" "vac" "nuc" "CYT" "MEM" "MIT" "NUC"
```

Ahora tengo que hacer lo mismo que con los anteriores, tengo que partir **data_ANN** para generar la parte de entrenamiento y la parte de test.

```
ANN_train <- data_ANN[train,]
ANN_test <- data_ANN[-train,]
```

2.2.4 Entrenar el modelo data_ANN

Ahora se entrenaran 2 modelos. Uno tendra 3 nodos en la capa oculta y el otro 5.

```
library(neuralnet)
xnam<-names(data_ANN[1:8])
(fmla=as.formula(paste("CYT+MEM+MIT+NUC ~", paste(xnam,collapse="+"))))
```

```
## CYT + MEM + MIT + NUC ~ mcg + gvh + alm + mit + erl + pox + vac +
##     nuc
```

```
set.seed(1234)
ANN_mod1<-neuralnet(fmla, data=ANN_train,hidden=1)
# entrenar el modelo con 3 nodos
ANN_mod3<-neuralnet(fmla, data=ANN_train,hidden=3)
# entrenar el modelo con 5 nodos
ANN_mod5<-neuralnet(fmla, data=ANN_train,hidden=5)
```

2.2.5 Predicción y evaluacion del ANN

Muestro el modelo.

```
#Evaluación del modelo con 3 nodos
ANN3results=compute(ANN_mod3,ANN_test[1:8])$net.result
maxidx<-function(arr){
return(which(arr == max(arr)))}
idx=apply(ANN3results,1,maxidx)
prediction=factor(idx,levels=1:4, labels= c("CYT","MEM","MIT","NUC"))
res3<-table(prediction,class_test)
evalANN3<-confusionMatrix(res3)
evalANN3
```

```
## Confusion Matrix and Statistics
##
##           class_test
## prediction CYT MEM MIT NUC
##        CYT  97   8  22  38
##        MEM   4  70   8   9
##        MIT  15   2  53   9
##        NUC  41   6   9  74
##
## Overall Statistics
##
##                Accuracy : 0.6323
##                  95% CI : (0.5866, 0.6762)
##     No Information Rate : 0.3376
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4978
##
##  Mcnemar's Test P-Value : 0.3235
##
## Statistics by Class:
##
##                      Class: CYT Class: MEM Class: MIT Class: NUC
```

```
## Sensitivity              0.6178     0.8140     0.5761     0.5692
## Specificity              0.7792     0.9446     0.9303     0.8328
## Pos Pred Value           0.5879     0.7692     0.6709     0.5692
## Neg Pred Value           0.8000     0.9572     0.8990     0.8328
## Prevalence               0.3376     0.1849     0.1978     0.2796
## Detection Rate           0.2086     0.1505     0.1140     0.1591
## Detection Prevalence     0.3548     0.1957     0.1699     0.2796
## Balanced Accuracy        0.6985     0.8793     0.7532     0.7010
```

Hago la predicción del modelo con 5 nodos.

```
ANN5results=compute(ANN_mod5,ANN_test[1:8])$net.result
maxidx<-function(arr){
return(which(arr == max(arr)))}
idx=apply(ANN5results,1,maxidx)
prediction=factor(idx,levels=1:4, labels= c("CYT","MEM","MIT","NUC"))
res5<-table(prediction,class_test)
evalANN5<-confusionMatrix(res5)
evalANN5
```

```
## Confusion Matrix and Statistics
##
##           class_test
## prediction CYT MEM MIT NUC
##        CYT  96  10  25  49
##        MEM   4  70   8  10
##        MIT  11   2  48   7
##        NUC  46   4  11  64
##
## Overall Statistics
##
##                Accuracy : 0.5978
##                  95% CI : (0.5517, 0.6427)
##     No Information Rate : 0.3376
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4481
##
##  Mcnemar's Test P-Value : 0.01897
##
## Statistics by Class:
##
##                      Class: CYT Class: MEM Class: MIT Class: NUC
## Sensitivity              0.6115     0.8140     0.5217     0.4923
## Specificity              0.7273     0.9420     0.9464     0.8179
## Pos Pred Value           0.5333     0.7609     0.7059     0.5120
## Neg Pred Value           0.7860     0.9571     0.8892     0.8059
## Prevalence               0.3376     0.1849     0.1978     0.2796
## Detection Rate           0.2065     0.1505     0.1032     0.1376
## Detection Prevalence     0.3871     0.1978     0.1462     0.2688
## Balanced Accuracy        0.6694     0.8780     0.7341     0.6551
```

Ahora lo que hago es tabular los datos y seguidamente hare el grafico de la red neuronal.

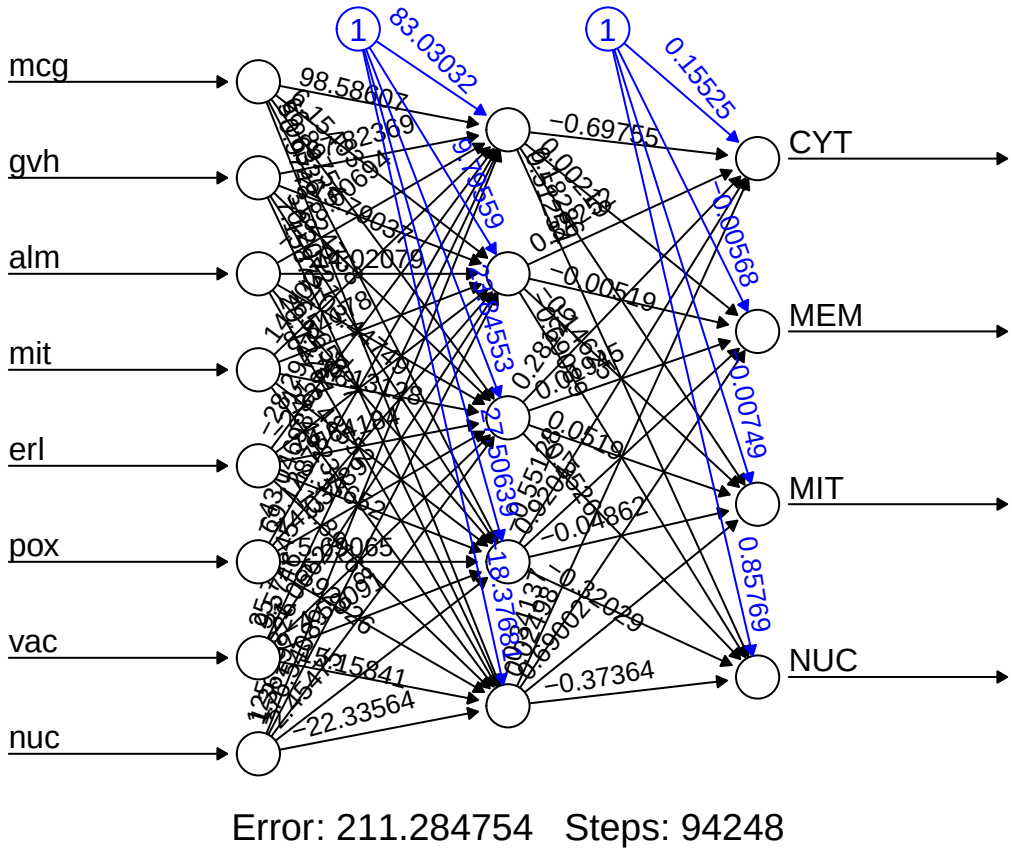
```
Nodos_ANN<-data.frame(Nodos=c(3,5))
ANN_All<-rbind(round(evalANN3$overall[1:4],3), round(evalANN5$overall[1:4],3))
ANN_All<-cbind(Nodos_ANN,ANN_All)
kable(ANN_All,align=c("l","c","c","c","c"),caption=paste("ANN"))
```

Table 3: ANN

Nodos	Accuracy	Kappa	AccuracyLower	AccuracyUpper
3	0.632	0.498	0.587	0.676
5	0.598	0.448	0.552	0.643

lo que observo al ver los datos obtenidos en el ANN son mejores los datos obtenidos en el que tiene 5 nodos, que el que tiene 3 nodos.

```
plot(ANN_mod5,rep="best")
```



2.3 Algoritmo SVM

En este apartado no hace falta que tranformemos los datos.

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
set.seed(1234)
class_train<-as.factor(data[train,9])
SVM_vanilladot<-ksvm(class_train ~.,data=out_training,kernel="vanilladot")
```

```
##  Setting default kernel parameters
```

```
set.seed(1234)
SVM_rbf<-ksvm(class_train ~.,data=out_training,kernel="rbf")
```

2.3.1 Prediccion y evaluacion del modelo de SVM

Ahora hago la predicción del modelo SVM

```
SVM_vanPredicion<-predict(SVM_vanilladot,out_test)
res_van<-table(SVM_vanPredicion,class_test)
svm_vanMat<-confusionMatrix(res_van)
```

```
SVM_rbfPredicion<-predict(SVM_rbf,out_test)
res_rbf<-table(SVM_rbfPredicion,class_test)
svm_rbfMat<-confusionMatrix(res_rbf)
```

Saco el resultado de la predicción del modelo.

```
Modelo<-data.frame(Modelo=c("Lineal","Gaussiano"))
SVM_All<-rbind(round(svm_vanMat$overall[1:4],3),round(svm_rbfMat$overall[1:4],3))
SVM_All<-cbind(Modelo,SVM_All)
kable(SVM_All,align=c("l","c","c","c","c"),caption=paste("Algoritmo SVM"))
```

Table 4: Algoritmo SVM

Modelo	Accuracy	Kappa	AccuracyLower	AccuracyUpper
Lineal	0.624	0.479	0.578	0.668
Gaussiano	0.647	0.514	0.602	0.691

Con los resultados que he obtenido puedo decir que tenemos una ligera mejor prediccion con el medelo “Gausiano”, que con el modelo “lineal”. Ahora lo que tengo que hacer es su matriz de confusion.

```
svm_rbfMat
```

```
## Confusion Matrix and Statistics
##
##                  class_test
## SVM_rbfPredicion CYT MEM MIT NUC
##              CYT 112   7  25  45
##              MEM   3  71   8   5
##              MIT   8   0  47   9
##              NUC  34   8  12  71
##
## Overall Statistics
##
##                Accuracy : 0.6473
##                  95% CI : (0.602, 0.6908)
##     No Information Rate : 0.3376
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5141
##
##  Mcnemar's Test P-Value : 0.001827
##
## Statistics by Class:
##
##                      Class: CYT Class: MEM Class: MIT Class: NUC
## Sensitivity              0.7134     0.8256     0.5109     0.5462
## Specificity              0.7500     0.9578     0.9544     0.8388
## Pos Pred Value           0.5926     0.8161     0.7344     0.5680
## Neg Pred Value           0.8370     0.9603     0.8878     0.8265
## Prevalence               0.3376     0.1849     0.1978     0.2796
## Detection Rate           0.2409     0.1527     0.1011     0.1527
## Detection Prevalence     0.4065     0.1871     0.1376     0.2688
## Balanced Accuracy        0.7317     0.8917     0.7326     0.6925
```

2.4 Algoritmo Classification Tree

Voy a preparar el modelo para entrenar el Algoritmo de arbol de decisión, en este modelo tampoco tengo que hacer ninguna transformacion de los datos como ocurrio en la red neuronal.

2.4.1 Entrenamiento del modelo del árbol de decisión.

En este caso tambien se entrenaran dos modelos. Uno sera C5.0 simple y el otro C5.0 haciendo boosting con 10 trials.

```
set.seed(1234)
CTree_Simple<-C5.0(class_train ~.,data=out_training)
set.seed(1234)
CTree_Boost<-C5.0(class_train ~.,data=out_training, trial=10)
```

2.4.2 Predicción y evaluación del modelo

```
class_test2<-as.factor(data[-train,9])
predicion_Simple<-predict(CTree_Simple,out_test)
evalSimple<-confusionMatrix(predicion_Simple,class_test2)

predicion_Boost<-predict(CTree_Boost,out_test)
evalBoost<-confusionMatrix(predicion_Boost,class_test2)
```

La tabla con los resultados es la siguiente:

```
Modelo<-data.frame(Modelo=c("Simple","Boost"))
CT_All<-rbind(round(evalSimple$overall[1:4],3),round(evalBoost$overall[1:4],3))
CT_All<-cbind(Modelo,CT_All)
kable(CT_All,align=c("l","c","c","c","c"),caption=paste("Algoritmo Classification Tree"))
```

Table 5: Algoritmo Classification Tree

Modelo	Accuracy	Kappa	AccuracyLower	AccuracyUpper
Simple	0.581	0.430	0.534	0.626
Boost	0.613	0.473	0.567	0.657

Como puedo ver el modelo "Boost" es ligeramente mejor que el modelo "lineal".

2.4.3 Algoritmo Random Forest

Vamos a hacer el entrenamiento del **Algoritmo Random Forest**, com hice anteriormente no es necesario transformar los datos en este modelo.

Voy a entrenar el modelo con dos algoritmos diferentes(uno con arbol 50 y otro con arbol 100)

```
set.seed(1234)
datos_tree50<-randomForest(class_train ~., data=out_training,ntree=50)
set.seed(1234)
datos_tree100<-randomForest(class_train ~., data=out_training,ntree=100)
```

2.4.4 Predicción y evaluación del modelo

Vamos a realizar la prediciión y evaluacion del modelo del **Algortimo Random Forest**

```
predicion_50<-predict(datos_tree50,out_test)
evaluacion_50<-confusionMatrix(predicion_50,class_test2)
predicion_100<-predict(datos_tree100,out_test)
evaluacion_100<-confusionMatrix(predicion_100,class_test2)
```

Ahora saco los resultados.

```
Numero_arbol<-data.frame(Modelo=c("50","100"))
RandomF_All<-rbind(round(evaluacion_50$overall[1:4],3),round(evaluacion_100$overall[1:4],3))
RandomF_All<-cbind(Numero_arbol,RandomF_All)
kable(RandomF_All,align=c("l","c","c","c","c"),caption=paste("Algoritmo Random Forest"))
```

Table 6: Algoritmo Random Forest

Modelo	Accuracy	Kappa	AccuracyLower	AccuracyUpper
50	0.662	0.537	0.617	0.705
100	0.662	0.538	0.617	0.705

Con los resultados obtenidos, lo que veo es que el modelo de arbol 50 es ligeramente superior al modelo de arbol 100.

3 Conclusión y Discusion sobre el rendimiento de los modelos

Es esta PEC se han utilizado 6 metodos que han sido estudiados durante el curso (k-Nearest Neighbour, Naive Bayes, Artificial Neural Network, Support Vector Machine, Arbol de Decisión y Random Forest.) Subo uno sin la tabla ahora te subo otro, lo siento muchisisimo, me esta adando muchos problemas.

```
library(kernlab)
```

```
ALL_row<- data.frame(Algoritmo=c("kNN", "Naive Bayes", "ANN","SVM", "C5.0", "RF"),
                     parametros=c("k= 11","laplace= 0","Nodos= 5", "Gausiano","trial= 10","Arbol= 100"))
ALL_sum<-rbind(kNN_all[5,2:5],round(evalNB_0$overall[1:4],3),
round(evalANN5$overall[1:4],3),round(svm_rbfMat$overall[1:4],3),
round(evalBoost$overall[1:4],3),round(evaluacion_100$overall[1:4],3))
ALL_sum<-cbind(ALL_row,ALL_sum)
kable(ALL_sum,align=c("l","c","c","c","c", "c"),caption=paste("Resultado algoritmos optimizados"))
```

Table 7: Resultado algoritmos optimizados

	Algoritmo	parametros	Accuracy	Kappa	AccuracyLower	AccuracyUpper
5	kNN	k= 11	0.617	0.475	0.571	0.662
2	Naive Bayes	laplace= 0	0.535	0.390	0.489	0.582
3	ANN	Nodos= 5	0.598	0.448	0.552	0.643
4	SVM	Gausiano	0.647	0.514	0.602	0.691
51	C5.0	trial= 10	0.613	0.473	0.567	0.657
6	RF	Arbol= 100	0.662	0.538	0.617	0.705

```
#En la tabla puedo ver  que todos los algoritmos sus valores estan entre 0.535 el m s bajo (algoritmo d
```