Detección de tos y no tos a partir de una señal de audio usando distintos clasificadores

Maria José Pinto Bernal
Departamento de Ingeniería Biomédica
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
maria.pinto@mail.escuelaing.edu.co

Diego Fernando Casas Bocanegra
Departamento de Ingeniería Biomédica
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
diego.casa-b@mail.escuelaing.edu.co

***Resumen*—La tos es un síntoma común de muchas enfermedades respiratorias. Muchas publicaciones médicas subrayan que un sistema para la detección automática, objetiva y fiable de los episodios de tos es importante y muy prometedor para detectar la gravedad de la patología, sobretodo en la enfermedad de la tos crónica. Utilizar el procesamiento de señales digitales para calcular los coeficientes espectrales característicos de los eventos de sonido, que luego se clasifican en eventos de tos y de no tos es muy prometedor para el diseño, validación y entrenamiento de clasificadores. Razón por la cual, en el presente artículo se presenta una fase de caracterización en donde se seleccionan 10 características relevantes de una señal de audio. A partir de estas características se diseñaron e implementaron 6 clasificadores: Knn, Naive-Bayes, Arbol de decisión, Regresión Logística Múltiple, RNA y SVM por medio de validación cruzada. Se obtuvo que el clasificador que presenta mayor rendimiento fue el de SVM con un desempeño del 92,5 %**

***Index Terms*—señal de audio, fase de caracterización, clasificador Knn, clasificador Naive-Bayes, arbol de desición, Regresión múltiple, RNA, SVM.**

I. Introducción

La tos es un síntoma común pero complicado de las enfermedades respiratorias. Este síntoma es el más común por el cual las personas solicitan una consulta médica [1]. De hecho, los estudios de población informaron que la prevalencia de tos puede variar entre el 3 % y 40 % [2], [3] Este síntoma se produce dado a una inspiración inicial profunda seguida de caducidad contra una glotis cerrada que se abre [4], [5]. Como resultado se genera una fonación característica, que se compone de dos componentes distintos denominado primero y segundo sonido de la tos [6].

Mientras que el reconocimiento de un solo evento de tos es relativamente fácil, la evaluación de la frecuencia de la tos durante un largo período de tiempo sigue siendo difícil, tanto para fines clínicos y de investigación. Parte del problema es la naturaleza de la tos paroxística que requiere la grabación durante un período de tiempo prolongado con el fin de generar una estimación precisa de la frecuencia de la tos [7]. Por lo tanto, existe la necesidad de desarrollar métodos basados en sonidos de la tos para contar y clasificar los eventos de la tos.

Considerando lo anterior, este artículo se centra en el diseño de un dataset a partir de una fase de caracterización de una señal de audio que presenta eventos de tos y sin tos; y en el diseño, entrenamiento e implementación de seis clasificadores: Knn, Naive-Bayes, Arbol de decisión, Regresión Logística Múltiple, RNA y SVM.

En primer lugar, es preciso mencionar que para la fase de caracterización fue necesario analizar la señal de audio con tos y sin tos previamente procesada en Matlab. Lo anterior, con el fin de seleccionar las características que presentan mayor relevancia e información de la señal. Se seleccionaron 10 características con las cuales se genero el dataset.

Ahora bien, respecto al diseño, entrenamiento e implementación de los clasificadores, se utilizó la validación cruzada por medio del dataset generado. La validación cruzada es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y de prueba. Una vez, generado cada clasificadores se evalúa el desempeño con el fin de establecer cual presenta mayor precisión.

A continuación se presenta la metodología implementada.

II. Metodología

Esta sección describe el procedimiento que fue realizado para la elaboración del dataset y el diseño de los clasificadores mediante una señal de audio obtenida por un voluntario masculino de 30 años de edad.

II-A. Pre-procesamiento

Se desarrolló un software de pre-procesamiento bajo Matlab versión R2018b [8]. En primer lugar, se analiza la señal para identificar los períodos de sonido dentro de las grabaciones. Para ello, se analiza la señal por cada frame para identificar los eventos de sonido, extraerlos, eliminar aquellos fragmentos de señal con silencio y generar una nueva señal de audio. En segundo lugar, se identifican los eventos de inicio y finalización de tos, esto se realiza de forma manual. En tercer lugar, se divide la señal en cada 1024 muestras, y por medio de los eventos encontrados se clasifica cada muestra en dos clase: 'tosó 'sintos'. A partir de esta clasificación se procede a encontrar el centroide de frecuencia de toda la señal y generar un archivo en donde se encuentre esta característica con su clasificación correspondiente (con tos o sin tos).

II-B. Fase de caracterización

La obtención de las características de la señal de audio sin eventos de silencio se realiza bajo RStudio [9] por medio de:

los coeficientes cepstrales en la frecuencia en la escala Mel, la función analyze y análisis de componentes principales. Cabe resaltar que se considera como una característica el centroide de frecuencia encontrado en el pre-procesamiento.

II-B1. Coeficientes Cepstrales en las Frecuencias de Mel: Los Coeficientes Cepstrales en la Escala de Mel (MFCC) representan la amplitud del espectro del habla de manera compacta, esto los ha vuelto la técnica de extracción de características más usada en reconocimiento de voz [10] debido a su bajo costo computacional y a su robustez [11]. El proceso para la elaboración de un vector característico de MFCC [11], [12] son las siguientes:

- Separar la señal en pequeños tramos.
- A cada tramo se le aplica la Transformada Discreta de Fourier (DFT) y esta información es pasada al dominio de Mel mediante el Banco de Filtros.
- Se obtiene el logaritmo de todas las energías de cada frecuencia Mel.
- Se aplica la Transformada de Coseno Discreta (DCT) a cada uno de estos logaritmos.

Los MFCC se encontraron mediante la función **melfcc** en RStudio. Dicha función retorna *n* cantidad de coeficientes de acuerdo a lo que se requiera. Para este estudio se obtuvieron 4 coeficientes era lo mas conveniente, de los cuales se seleccionaron el coeficiente 1 y 4 para el diseño del data set.

II-B2. Función Analyze: La función **analyze** tiene como fin retornar diferentes características de la señal de acuerdo a su frecuencia de muestreo, en este caso de 48 kHz. Dicha función retorna 27 características propias de la señal de audio de las cuales se analizaron y se escogieron las 7 características mas relevantes:

- Amplitud RMS por frame
- Pitch
- Entropia
- Harmonicos
- Pendiente
- Frecuencia
- Ancho

II-C. Dataset

A partir de las 10 características encontradas y seleccionadas en la fase de caracterización se genera el dataset. Cada característica se clasificada ya sea en la clase de *'tos'* o *'sintos'*.

II-D. Clasificadores

Respecto al diseño y validación de los clasificadores, esta sección se divide en tres partes. La primera parte presenta el diseño de los datos. La segunda parte consiste en presentar el diseño y finalidad de cada clasificador. La tercera parte el desempeño de los clasificadores.

Entrenamiento y validación

En primer lugar, se carga el dataset generado y se procede a normalizar los datos (ver ecuación 1).

$$\frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

En segundo lugar, se generan los datos de entrenamiento y prueba. Se seleccionaron aleatoriamente el 70% de los datos para entrenamiento y el 30% restante para prueba por medio de la función **split** como se ilustra a continuación.

```
split<- sample.split(data_norm$clase, SplitRatio = 0.7)
data_train = subset(data_norm , split == TRUE)
data_test = subset(data_norm , split == FALSE)
```

A partir de estos datos, se diseña y entrenan los clasificadores.

Diseño

Cabe resaltar que el diseño de los seis clasificadores: Knn, Naive-Bayes, Arbol de decisión, Regresión Logística Múltiple, RNA y SVM, se realiza bajo el software de RStudio. Una vez, se diseñe cada clasificador se procede a evaluar su desempeño.

II-D1. Clasificador Knn: Es un algoritmo basado en instancia de tipo supervisado de Machine Learning. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Al ser un método sencillo, es ideal para introducirse en el mundo del Aprendizaje Automático. Sirve esencialmente para clasificar valores buscando los puntos de datos más similares (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación.

A partir de los datos de entrenamiento y de validación generados, se hace uso del clasificador Knn. El cual cuenta con cuatro argumentos. El primer argumento hace referencia a los datos de entrenamiento, el segundo argumento los datos de validación, el tercer argumento los datos de clasificación correspondiente a los datos de entrenamiento y el ultimo argumento, a la cantidad de k-cercanos considerados o también conocido como vecinos considerados, donde el número de vecinos es el factor decisivo del clasificador, en este caso es 10.

```
pr = knn(dataset_train,dataset_test,cl=dataset_target_category,k=10)
```

II-D2. Naive-Bayes: Es una técnica de clasificación y predicción supervisada que construye modelos que predicen la probabilidad de posibles resultados. Constituye una técnica supervisada porque necesita tener ejemplos clasificados para que funcione, está basada en el Teorema de Bayes, también conocido como teorema de la probalibidad condicionada.

Para el diseño del clasificador Naive-Bayes, se usan los datos de entrenamiento y validación ya realizados previamente. El cual cuenta con dos argumentos, el primero hace referencia a los datos de entrenamiento y el tercer argumento los datos de clasificación correspondiente a los datos de entrenamiento. Posterior a ellos se realiza la predicción de clasificador por medio de la función *predict* que tiene como argumento de entrada el modelo de clasificación y los datos de validación.

```
m=naiveBayes(data_train,data_train_category)
pr = predict(m,data_test)
```

II-E. Árbol de decisión

Aprendizaje basado en árboles de decisión utiliza un árbol de decisión como un modelo predictivo que mapea observaciones sobre un artículo a conclusiones sobre el valor objetivo del artículo. Es uno de los enfoques de modelado predictivo utilizadas en estadísticas, minería de datos y aprendizaje automático. Los modelos de árbol, donde la variable de destino puede tomar un conjunto finito de valores se denominan árboles de clasificación. En estas estructuras de árbol, las hojas representan etiquetas de clase y las ramas representan las conjunciones de características que conducen a esas etiquetas de clase.

La realización de este modelo de clasificación, se muestra a continuación la función utilizada en R.

```
tree <- ctree(clase ~ ., data = data_train)
```

II-F. Regresión lineal y logística

El método de regresión logística permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa. Una de las principales aplicaciones de la regresión logística es la de clasificación binaria, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la variable empleada como predictor.

```
modelo_glm <- glm(clase ~ Amplitud_RMS + Pitch +
    Centroide_de_frecuencia + Coeficiente_1_de_MFCC
    + Coeficiente_4_de_MFCC + Entropy + Harmonics +
    SpecSlope + F1_freq + F1_width , data = data_
    train ,
                  family = "binomial")

pr=predict(modelo_glm,newdata=data_test,type="
    response")
```

II-G. Redes neuronales

Una Red Neuronal Artificial, RNA, desde ahora es básicamente un modelo simplificado del modo en que los sistemas nerviosos procesan información. Funciona en forma simultánea con un número de unidades simples de procesamiento interconecta-das que emulan a las neuronas (llamadas también nodos), están organizadas en niveles denominados capas. Cada nodo está conectado con otros mediante enlaces de comunicación, cada uno de los cuales tiene asociado un peso. En los pesos se encuentra el çonocimiento"que tiene la RNA acerca de un determinado problema. Algunas de las redes neuronales son herramientas útiles en muchas aplicaciones de predicción en minería de datos debido a su potencia, flexibilidad y facilidad de uso.

Se mostrara el modelo de red neuronal por medio del cual se diseño el clasificador para esta aplicación.

```
model_results <- compute(redes_neuronales, data_test
    [1:10])
predict <- model_results$net.result
```

II-H. Maquina de soporte vectorial SVM

La función kernel utilizada en el entrenamiento y la predicción. Este parámetro se puede establecer en cualquier función, de la clase kernel, que calcula un producto interno en el espacio de características entre dos argumentos vectoriales. Kernlab proporciona las funciones de kernel más populares que se pueden usar configurando el parámetro del kernel en las siguientes cadenas:

- rbfdot: Radial Basis kernel function "Gaussian"
- polydot: Polynomial kernel function
- vanilladot: Linear kernel function
- laplacedot: Laplacian kernel function
- besseldot: Bessel kernel function
- anovadot: ANOVA RBF kernel function
- splinedot: Spline kerne

Después de realizar el modelo con las diferentes funciones de kernel se determino que el mejor desempeño se obtuvo con la función anovadot. A continuación se ilustra la función utilizada en R para el diseño del clasificador.

```
modelo <- ksvm(clase ~.,data=data_train,kernel = "
    anovadot")
predictions <- predict(modelo, data_test)
```

Desempeño

Ahora bien, es importante recalcar que todos los clasificadores se entrenan y testean por el método de validación cruzada, la cual es una técnica utilizada para evaluar los resultados del clasificador y garantizar que son independientes de la partición entre datos de entrenamiento y validación.

Para ello se genera la matriz denominada *tab* por medio de la función *table* que contiene dos argumentos. El primero correspondiente al modelo del clasificador y el segundo a los datos de clasificación correspondiente a los datos de prueba. A partir de estos datos se utiliza los factores de validación cruzada para construir una tabla de contingencia de los recuentos en cada combinación de los niveles de factores.

```
tab = table(clasificador,dataset_test_category)
```

Con el propósito de medir la precisión de los clasificadores se genera una función matemática denominada *accuracy* que tiene como fin calcular la exactitud o el desempeño del clasificador correspondiente.

```
accuracy = function(x){sum(diag(x)/(sum(rowSums(x)))
    ) * 100}
```

Partiendo de de la función generada, se realiza el calculo del desempeño del clasificador correspondiente a partir de la matriz de confusión generada.

```
Precision=accuracy(tab)
```

Finalmente, se imprime en pantalla el resultado del desempeño obtenido por el clasificador.

```
print(paste("Desempe o clasificador", Presicion))
```

Para una mayor profundización en la sección de anexos se adjunta el código implementado.

III. Resultados

Esta sección presenta los resultados obtenidos a partir del procesamiento y caracterización de una señal de audio con eventos de tos y sin tos procedente de un voluntario masculino de 30 años de edad. Los resultados se dividen en tres partes principales; (i) Pre-procesamiento, (ii) Fase de caracterización, y (iii) Desempeño de los clasificadores.

III-A. Pre-procesamiento

Esta parte incluye el pre-procesamiento de la señal realizado bajo Matlab. En la 1 se ilustra en la parte superior la señal inicial del voluntario y en la parte inferior la señal procesada solamente con los eventos donde se presenta sonido. En dichas gráficas se observa la reducción de la señal comparada una con la otra, dado la eliminación de los eventos sin sonido. Así mismo, se ilustra en la señal inicial (parte superior) tres recuadros vino-tintos, los cuales ejemplifican tres eventos sin sonidos de la señal.

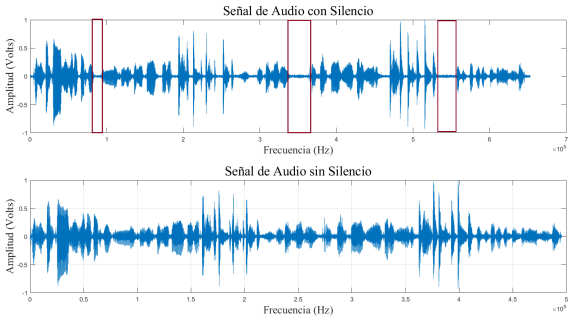

Figura 1: Señal de audio con silencio y sin silencio. En la parte superior se presenta la señal inicial y se presentan tres rectángulos vino-tintos que hace referencia a fragmentos de señal sin sonido. En la parte inferior se presenta la señal con los fragmentos sin sonido removidos.

III-B. Fase de caracterización

Esta segunda parte constituye la fase de caracterización de la señal de audio pre-procesada, es decir, sin eventos sin sonido. En el cuadro I, se ilustra la media y la desviación estándar de las 10 características de la señal, y cada característica debidamente etiquetada, ya sea en la clase de tos o sintos. Cabe resaltar que se obtuvo 313 datos de cada característica mencionada (con su etiquetado correspondiente); con lo cual se conformo el data set.

III-C. Clasificadores

Esta última parte constituye los resultados del desempeño de los seis clasificadores implementados a partir del data set generado. El cuadro II ilustra que el clasificador Knn obtuvo un desempeño del 81,914 % y el clasificador Naive-Bayes un desempeño del 76,595 % a partir del data set.

IV. Análisis

Siguiendo el mismo orden, esta sección presenta la discusión y el análisis de los resultados obtenidos en este estudio. En primer lugar, de acuerdo al pre-procesamiento, se observo como en la señal inicial de audio se identificaron los fragmentos sin sonido, los cuales presentan un nivel de amplitud en voltios muy bajo. Razón por la cual, se estableció un umbral para identificar los eventos con sonido y sin sonido. De igual manera, se observo que aquellos eventos sin sonido presentan un comportamiento lineal. En la señal de audio pre-procesada se observa que dichos comportamientos desaparecen, se reduce la señal y se obtiene finalmente la señal compacta con todos eventos que presentan sonido.

De acuerdo a la fase de caracterización se observo que existe un aumento en la media de los datos de las siguientes características: amplitud RMS, pitch, entropia, frecuencia y ancho, en los eventos que no presentan tos. Mientras que las características: centroide de frecuencia, coeficiente 1 y 4 de MFCC y los armonicos de la señal presentaron mayor valor en la media de los datos, en los eventos de la señal que presentan tos.

Respecto al desempeño de los clasificadores, los resultados mostraron que el clasificador con mejor desempeño fue el modelo de maquina de soporte vectorial (SVM) usando la función de Kernel presento un desempeño del 92,553 %. Es pertinente mencionar que la función de Kernel usada fue la Anova, la cual presenta alta precisión el problemas de regresión multidimensional. No obstante, la elección de la función mas adecuada es relativo y depende en gran medida del problema en cuestión, por consiguiente, se debe evaluar en gran medida cual función presenta mejor rendimiento de acuerdo a la aplicabilidad. Por ejemplo, el método para el cual se selecciono dicha función, fue mediante la aplicación de las otras funciones que en este caso presentaba Rstudio, las cuales eran 9 y de dichas, funciones se seleccionó Anova que fue la que presento mayor desempeño. Por otro lado, el clasificador que presento menor desempeño fue el modelo de regresión logística múltiple 48,936 %. Este resultado indica que para esta aplicabilidad dicho clasificador no es pertinente de utilizar.

De la misma manera, se observo que a excepción del modelo de regresión logística múltiple, los otros clasificadores presentaron un desempeño superior al 80 %, indicando así que son buenos clasificadores. No obstantes, los mas recomendados para utilizar en detección de tos y no tos en una señal de audio son KNN y SVM, dado que presentaron mayor precisión.

V. Conclusiones

Este artículo presento el análisis de una señal de audio con tos y sin tos de un voluntario masculino de 30 años de edad. Este estudio mostró la obtención de 10 características de la señal de audio con las cuales se diseño, entreno y valido seis clasificadores. A modo de comparar el rendimiento de los clasificadores, se diseñaron mediante la misma validación cruzada, con el fin de asegurar los mismos datos de entrenamiento y de testeo. El clasificador que presento

Cuadro I: Dataset. Se ilustra la media y la desviación estándar de las 10 características que conforman el data frame, correctamente etiquetadas al evento sintos o contos

	Amplitud RMS	Pitch	Centroide de frecuencia	Coeficiente 1 de MFCC	Coeficiente 4 de MFCC	Entropy	Harmonics	SpecSlope	F1_freq	F1_width	Clase
sin_mean	0.1066382	195.415	22.67393789	5.7499209	0.2309791	0.210147	5.6685457	-0.001143	586.229	168.3998	sintos
sin_std	0.0563452	66.4472	14.60640164	0.2309779	0.1726714	0.12895	3.2006574	0.0007389	370.845	127.8228	sintos
con_mean	0.0856583	193.077	31.78442105	6.0999284	0.2644305	0.198307	6.0833233	-0.00096	518.408	166.0674	contos
con_std	0.0485543	40.912	16.9693586	0.3188528	0.1714635	0.125339	2.5825908	0.0005922	238.139	128.4406	contos

Cuadro II: Resultado del desempeño de los clasificadores mediante los datos del data set.

Clasificador	Desempeño (%)
Knn	89.361
Naive-Bayes	87.234
Árbol de decisión	80.851
Regresión Logística Múltiple	48.936
Redes Neuronales	81.914
SVM	92.553

mayor desempeño fue el de modelo SVM con un desempeño del 92,5% mediante el uso de la función kernel con anova. No obstante, a excepción del modelo de regresión logística múltiple todos los clasificadores presentaron un desempeño superior al 80% indicando que son buenos clasificadores y podrían ser utilizados.

Aunque los resultados que se presentan en este artículo son prometedores, esta investigación se encuentra todavía en una fase temprana. Por lo tanto, los hallazgos de este trabajo servirán como base para extender nuestros estudios a una escala mayor. Por lo tanto, como un paso futuro, se propone validar estos resultados con un mayor número de señales de audio. Esto permitirá un análisis estadístico más robusto, proporcionando información significativa asociada a los clasificadores.

VI. Anexo - Código Implementado

A continuación se ilustra el código implementado.

```
#Trabajo Segundo Corte

## Librerias
library(ggfortify)
library(class)
library(lattice)
library(matrixStats)
library(MASS)
library(e1071)
library(caret)
library(klaR)
library(readxl)
library(party)
library(readr)
library(neuralnet)
library(caTools)
library(randomForest)
library(kernlab)
library(ggplot2)
library(vcd)
library(plyr)


# Se carga el dataset correspondiente

data_frame <- read_excel("~/Maestria/Renocimiento automatico de patrones/Caracteristicas_Completas.xlsx")
str(data_Frame)

#Normalizac on de los datos.
normalize <- function(x) {return ((x - min(x))/(max(x)- min(x)))}
data_norm <- as.data.frame(lapply(data_frame[,c(1,2,3,4,5,6,7,8,9,10)], normalize))
data_norm$clase<-data_frame$clase
data_norm$clase<-as.factor(data_norm$clase)

#Divis n de los datos de entreamiento y validaci n

split<- sample.split(data_norm $clase, SplitRatio = 0.7)
data_train = subset(data_norm , split == TRUE)
data_test = subset(data_norm , split == FALSE)

# Clasificador KNN

classifier_knn = knn(data_train[,c(1,2,3,4,5,6,7,8,9,10)],data_test[,c(1,2,3,4,5,6,7,8,9,10)],cl=data_train$clase,k=10)
tab = table(classifier_knn,data_test$clase)
accuracy = function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
ty=accuracy(tab)
print(paste("Desempe o KNN",ty))

# Clasificador NaiveBayes

classifier_naiveBayes = train(data_train[,c(1,2,3,4,5,6,7,8,9,10)],data_train$clase,'nb')
pr1 = predict(classifier_naiveBayes,data_test[,c(1,2,3,4,5,6,7,8,9,10)])
tab1 = table(pr1,data_test$clase)
P1=accuracy(tab1)
print(paste("Desempe o NaiveBayes",P1))

#Arboles de desici n

tree <- ctree(clase ~ ., data = data_train)
resultados = predict(tree, data_test[,1:10])
clasificaciones = table(resultados, data_test$clase)
P_arbol = accuracy(clasificaciones)
print(paste("Desempe o Arbol de desici n",P_arbol))


#Regresi n l gica m ltiple

modelo_glm <- glm(clase ~ Amplitud_RMS +  Pitch + Centroide_de_frecuencia + Coeficiente_1_de_MFCC
```

```
    + Coeficiente_4_de_MFCC + Entropy + Harmonics +
    SpecSlope + F1_freq + F1_width , data = data_
    train ,
                    family = "binomial")
umbral = 0.02
summary(modelo_glm)
predicciones <- ifelse(test = modelo_glm$fitted.
    values > umbral,yes = "contos", no = "sintos")
matriz_confusion <- table(modelo_glm$model$clase,
    predicciones,
                              dnn = c("observaciones", "
    predicciones"))
pr=predict(modelo_glm,newdata=data_test,type="
    response")
predicciones <- ifelse(pr > umbral,  yes = "contos",
     no = "sintos")
matriz_confusion_1 <- table(predicciones,data_test$
    clase)
matriz_confusion_1
p_regresion=accuracy(matriz_confusion_1)
print(paste("Desempe o Regresi n m ltiple",p_
    regresion))



## Cambia el nombre de la clase a binario
data_test$clase<-as.character(data_test$clase)
data_train$clase<-as.character(data_train$clase)


data_test$clase[data_test$clase == 'sintos'] <- 0
data_test$clase[data_test$clase == 'contos'] <- 1

data_train$clase[data_train$clase == 'sintos'] <- 0
data_train$clase[data_train$clase == 'contos'] <- 1

data_test$clase <- as.numeric(data_test$clase)
data_train$clase <- as.numeric(data_train$clase)


#Redes Neuronales

redes_neuronales <- neuralnet(clase ~ Amplitud_RMS +
      Pitch + Centroide_de_frecuencia + Coeficiente_
    1_de_MFCC + Coeficiente_4_de_MFCC + Entropy +
    Harmonics + SpecSlope + F1_freq + F1_width, data
     = data_train, hidden = c(10, 5, 7)); # Build
    the model
model_results <- compute(redes_neuronales, data_test
    [1:10])
predict <- model_results$net.result
cor(predict, data_test$clase)
results <- data.frame (actual = as.numeric(data_test
    $clase), prediction = model_results$net.result)
rounded <- sapply(results, round, digits=0)
r <- data.frame(rounded)
attach(r)
tab2 = table (actual,prediction)
p2=accuracy(tab2)
print(paste("Desempe o Redes neuronales",p2))


#SVM

data_test$clase<-as.factor(data_test$clase)
data_train$clase<-as.factor(data_train$clase)
rbf <- rbfdot(sigma=0.1)
rbf
modelo <- ksvm(clase ~.,data=data_train,kernel = "
    anovadot")
modelo
predictions <- predict(modelo, data_test)
head(predictions)
tab<-table(predictions,data_test$clase)
agreement = predictions == data_test$clase
table(agreement)
prop.table(table(agreement))
p3=accuracy(tab)
print(paste("Desempe o SVM",p3))
```

REFERENCIAS

[1] S. M. Schappert and C. W. Burt, "Ambulatory care visits to physician offices, hospital outpatient departments, and emergency departments: United States, 2001-02." *Vital and health statistics. Series 13, Data from the National Health Survey*, no. 159, pp. 1–66, feb 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16471269

[2] K. Lai, H. Shen, X. Zhou, Z. Qiu, S. Cai, K. Huang, Q. Wang, C. Wang, J. Lin, C. Hao, L. Kong, S. Zhang, Y. Chen, W. Luo, M. Jiang, J. Xie, and N. Zhong, "Clinical Practice Guidelines for Diagnosis and Management of Cough-Chinese Thoracic Society (CTS) Asthma Consortium." *Journal of thoracic disease*, vol. 10, no. 11, pp. 6314–6351, nov 2018. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/30622806 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6297434

[3] P. Cullinan, "Persistent cough and sputum: prevalence and clinical characteristics in south east England." *Respiratory medicine*, vol. 86, no. 2, pp. 143–9, mar 1992. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/1615181

[4] A. H. Morice, G. A. Fontana, M. G. Belvisi, S. S. Birring, K. F. Chung, P. V. Dicpinigaitis, J. A. Kastelik, L. P. McGarvey, J. A. Smith, M. Tatar, and J. Widdicombe, "ERS guidelines on the assessment of cough," *European Respiratory Journal*, vol. 29, no. 6, pp. 1256–1276, mar 2007. [Online]. Available: http://erj.ersjournals.com/cgi/doi/10.1183/09031936.00101006

[5] U. R. Abeyratne, V. Swarnkar, R. Triasih, and A. Setyati, "Cough Sound Analysis - A new tool for diagnosing Pneumonia," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, jul 2013, pp. 5216–5219. [Online]. Available: http://ieeexplore.ieee.org/document/6610724/

[6] R. X. A. Pramono, S. A. Imtiaz, and E. Rodriguez-Villegas, "A Cough-Based Algorithm for Automatic Diagnosis of Pertussis," *PLOS ONE*, vol. 11, no. 9, p. e0162128, sep 2016. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0162128

[7] Y. Shi, H. Liu, Y. Wang, M. Cai, and W. Xu, "Theory and Application of Audio-Based Assessment of Cough," *Journal of Sensors*, vol. 2018, pp. 1–10, 2018. [Online]. Available: https://www.hindawi.com/journals/js/2018/9845321/

[8] C. Moler and MathWorks, "Matlab R2018b," 2018. [Online]. Available: https://www.mathworks.com/?requestedDomain=

[9] J. Allaire and Rstudio Inc, "RStudio," 2011. [Online]. Available: https://rstudio.com/

[10] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *Ismir*, vol. 270, 2000, pp. 1–11.

[11] G. A. Martínez Mascorro and G. Aguilar Torres, "Reconocimiento de voz basado en MFCC, SBC y Espectrogramas," *Ingenius*, no. 10, dec 2013. [Online]. Available: http://revistas.ups.edu.ec/index.php/ingenius/article/view/10.2013.02

[12] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of MFCC," *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 582–589, nov 2001. [Online]. Available: http://link.springer.com/10.1007/BF02943243