Recibido: 19 de agosto del 2017
Aprobado: 23 de setiembre del 2017

Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python

Rosa Fátima Medina Merino
rmedinam@ulima.edu.pe
Universidad de Lima. Lima, Perú

Carmen Ismelda Ñique Chacón
carmen_ismelda@hotmail.com
Instituto Nacional de Estadística e Informática. Lima, Perú

Resumen

El presente artículo presenta la aplicación del método no paramétrico *Random Forest* mediante el aprendizaje supervisado, como una extensión de los árboles de clasificación. El algoritmo de *Random Forest* surge como la agrupación de varios árboles de clasificación; básicamente selecciona de manera aleatoria una cantidad de variables con las cuales se construye cada uno de los árboles individuales, y se realizan predicciones con estas variables que posteriormente serán ponderadas a través del cálculo de la clase más votada de los árboles que se generaron, para finalmente hacer la predicción por *Random Forest*. Para la aplicación se trabajó con 3168 registros de voz grabados, para los cuales se presentan los resultados de un análisis acústico, registrándose variables tales como frecuencia, espectro, modulación, entre otras, con lo cual se busca obtener un patrón de identificación y clasificación según género a través de un identificador de voz. El registro de datos utilizado es de acceso libre y puede ser descargado desde la plataforma web de Kaggle a través del enlace <https://www.kaggle.com/primaryobjects/voicegender>. Para el desarrollo del algoritmo del modelo, se recurrió al programa estadístico R. Adicionalmente, se realizaron aplicaciones con Python mediante el desarrollo de algoritmos de clasificación.

Palabras clave: Bosques aleatorios / árboles de clasificación / modelos no paramétricos de clasificación / aprendizaje supervisado / lenguaje R / lenguaje Python

Abstract

Random Forests as an extension of the classification trees with the R and Python programs

This article presents the application of the non-parametric Random Forest method through supervised learning, as an extension of classification trees. The Random Forest algorithm arises as the grouping of several classification trees. Basically it randomly selects a number of variables with which each individual tree is constructed and predictions are made with these variables that will later be weighted through the calculation of the most voted class of these trees that were generated, to finally do the prediction by Random Forest. For the application, we worked with 3168 recorded voices, for which the results of an acoustic analysis are presented, registering variables such as frequency, spectrum, modulation, among others, seeking to obtain a pattern of identification and classification according to gender through a voice identifier. The data record used is in open access and can be downloaded from the Kaggle web platform via <https://www.kaggle.com/primaryobjects/voicegende>r. For the development of the algorithm's model, the statistical program R was used. Additionally, applications were made with Python by the development of classification algorithms.

Keywords: Random Forest / classification trees / non-parametric classification models / supervised learning / R language / Python language

1. Introducción

Una manera de sintetizar información es la que se realiza mediante la conformación y caracterización de grupos o clases, los cuales se conforman de tal manera que los elementos dentro de cada grupo sean lo más homogéneos posibles y que, en cambio, los elementos de diferentes grupos sean lo más diferentes posibles. Trabajar con un árbol de clasificación es una buena alternativa para realizar una clasificación, pero ¿qué sucede cuando se tiene una gran cantidad de datos y variables a la vez?

La gran cantidad de datos que se manejan en la actualidad conlleva a utilizar métodos estadísticos que permitan realizar un análisis de ellos, incluso para trabajar con muchas variables. En la búsqueda de herramientas para manejar semejante cantidad de datos surge la minería de datos, la cual no solo abarca el campo de la estadística, sino también el de las ciencias de la computación. Al hacer interactuar estos grandes campos, surge el aprendizaje supervisado y el no supervisado; dentro del aprendizaje supervisado se tiene el modelo no paramétrico *Random Forest*, el cual es una técnica de clasificación que se basa en un conjunto de árboles de decisiones, y que es ideal para trabajar con una gran cantidad de datos y múltiples variables, ya que selecciona submuestras para elaborar cada árbol. Con ello se garantiza el uso de todas las variables y datos para construir el modelo. Al igual que *Random Forest*, dentro de la minería de datos se encuentran otros modelos para realizar análisis estadísticos en las múltiples áreas de aplicación donde se genere una gran cantidad de datos y se requiera establecer agrupaciones o clasificaciones.

El reconocimiento de voz es una técnica sobre la que se vienen perfeccionando diversos aplicativos con gran cantidad de utilidades, que van desde los sistemas de telecomunicaciones, la activación de circuitos con el simple reconocimiento de voz, entre otras. Por ello, en este artículo se realizará la aplicación de un modelo de clasificación a partir del registro VOICE, de libre acceso, y que cuenta con una cantidad superior a los 3000 registros.

2. Conceptos previos

2.1 Árboles de clasificación

Los árboles de clasificación incorporan un enfoque de clasificación supervisada, la idea surgió de la estructura de un árbol que se compone de una raíz, nodos (las posiciones donde las ramas se dividen), ramas y hojas; de manera similar, un árbol de clasificación se construye a partir de nodos que representan los círculos y las ramas son representadas por los segmentos que conectan los nodos. Un árbol de clasificación se inicia desde la raíz, se extiende hacia abajo y generalmente se dibuja de izquierda a derecha. El nodo inicial se llama nodo *raíz*, mientras los nodos en los extremos de la cadena se les conocen como nodos *hoja*. Dos o más ramas pueden extenderse desde cada nodo interno, es decir, desde un nodo que no es el nodo hoja (Ali, Khan, Ahmad y Maqsood, 2012).

2.2 Árboles de clasificación (CART)

Los árboles de clasificación y regresión (CART) fueron desarrollados por Breiman, Freidman, Olshen y Stone en el libro *Classification and Regression Trees,* publicado en 1984. Para los árboles de clasificación la bondad de una división se cuantifica por una medida de impureza; se dice que una división es pura si, después de la división, todas las instancias de la elección de una rama pertenecen a la misma clase. Para el nodo m, N_m es el número de instancias de entrenamiento que alcanza el nodo m. Para el nodo raíz, esto es N. N_m^i de N_m pertenecen a las clases C_i, donde $\sum_i N_m^i = N_m$. Dado que una instancia alcanza el nodo m, la estimación de la probabilidad de la clase C_i es:

$$\hat{P}(C_i \backslash x, m) \equiv p_m^i = \frac{N_m^i}{N_m} \qquad (1)$$

El nodo m es puro si p_m^i para todo i son 0 o bien 1. Es 0 cuando ninguna de las instancias del nodo m son de Clase C_i, y es 1 si todos estos casos son de C_i. Si la división es pura, no es necesario dividir más y se puede añadir un nodo hoja etiquetado con la clase para la cual p_m^i es 1.

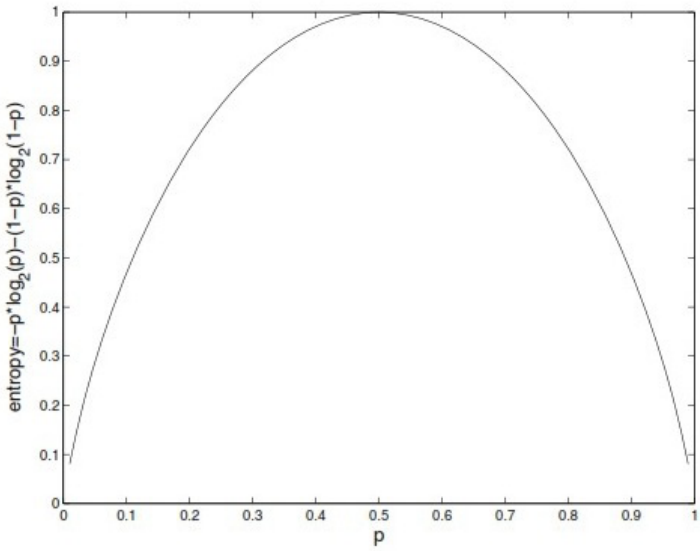
2.2.1 Medición de la impureza de un árbol

Una función posible para medir la impureza es la entropía (figura 1).

$$I_m = -\sum_{i=1}^{K} p_m^i log\ p_m^i \qquad (2)$$

Donde $0log0 \equiv 0$. La entropía en teoría de la información señala el número mínimo de bits necesarios para codificar el código de clase de una instancia.

Figura 1. Función de entropía para dos clases



Fuente: Alpaydin (2010, p. 189)

Pero la entropía no es la única función para medir la impureza del nodo; existen otras, como el índice de diversidad de Gini (Breiman, Friedman, Stone y Olshen, 1984), el cual trata de minimizar la impureza existente en los subconjuntos de casos de entrenamiento generados al ramificar por un atributo determinado. Su función es la siguiente:

$$G(C_i \backslash p_m^i) = 1 - \sum_{k=1}^{N} (p_m^i)^2 \quad (3)$$

Misclassification error (error de clasificación):

$$E = 1 - \max[p_m^i] \quad 4)$$

(Alpaydin, 2010)

2.2.2 Poda de árboles

Cuando un árbol presenta sobreajuste, es decir, tiene muchas ramas, lo cual hace compleja su interpretación y manejo, es necesario, entonces, realizar una poda de este, que consiste en cortar sucesivamente las ramas y nodos terminales hasta lograr un tamaño adecuado para dicho árbol. Existen dos tipos de poda: la prepoda y la pospoda.

La prepoda consiste en no dividir según la clase si se produce una determinada condición; esto es, ya no se sigue construyendo el árbol de decisión a partir del nodo donde se cumple la condición. Una de las formas para lograr este propósito es comparando el error de clasificación que se obtiene en un nodo que ya se ha expandido frente al error del árbol sin expandir; si este es menor que el error del nodo expandido, entonces se permanece con el nodo sin expandir (poda). El segundo tipo de poda, la pospoda, se realiza cuando ya se ha generado todo el árbol de decisión. Una de las maneras para realizar este tipo de poda es generar todo el árbol y luego empezar a analizar qué nodo se puede eliminar para reducir el error de clasificación; a veces en esta forma se llega a hacer de manera recursiva la eliminación de algún nodo, puesto que lo que se desea es no incrementar el error de clasificación.

2.3 Bagging y Boosting

Bagging y *Boosting* se enfocan en usar modelos individuales para desarrollar un mejor método. En *Bagging* se da a los modelos un peso igual. De manera contraria se desarrolla en *Boosting*: en este se dan distintos pesos a los modelos con el objetivo de brindarles mayor ponderación a aquellos que resaltan más.

Para hacer una introducción a *Bagging*, suponga que se tiene un conjunto de datos del mismo tamaño, y se utilizará una técnica de máquina de aprendizaje para construir un árbol de decisión para cada conjunto de datos. Lo que se podría pensar es que cada árbol sea casi idéntico a otro y que, por lo tanto, se obtendrá una misma predicción para un nuevo caso; pero esta suposición no es cierta, más aún si los conjuntos de datos son reducidos. Si los datos de entrenamiento sufrieran algún cambio, entonces se tendría como resultado fácilmente un atributo diferente, lo cual implica que existe la posibilidad de que en los casos de prueba para algunos árboles de decisión se produzcan predicciones correctas y otras no. *Bagging* es un método de aprendizaje estadístico cuyo procedimiento tiene como propósito la reducción de la varianza: este método es útil en el contexto de árboles de decisión.

Se tiene un conjunto de n observaciones independientes $x_1, x_2, \ldots, x_n$, cada una con varianza σ^2/n; entonces, la varianza de la media $\bar{X}$ de las observaciones está dada por σ^2/n; con esto se puede concluir que utilizando un conjunto de observaciones se reduce la varianza. Una

forma natural de reducir la varianza, y por consiguiente aumentar la precisión de la predicción de un método de aprendizaje estadístico, es seleccionar una gran cantidad de conjuntos de entrenamiento de la población y construir un modelo de predicción independiente utilizando cada conjunto de entrenamiento. En otras palabras, se puede calcular, $\widehat{h^1}(x), \widehat{h^2}(x), \ldots, \widehat{h^B}(x)$ utilizando *B* conjuntos de entrenamiento por separado, y un promedio de ellas con el fin de obtener un único modelo de aprendizaje estadístico de varianza pequeña.

$$\widehat{h_{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{h^b}(x) \qquad (5)$$

Por supuesto, esto no es práctico, ya que generalmente no se tiene acceso a múltiples conjuntos de entrenamiento. En su lugar se puede iniciar tomando muestras repetidas a partir del conjunto de datos de entrenamiento (individual). En este enfoque se generan *B* diferentes conjuntos de datos de entrenamiento *bootstrap*. A continuación se entrena con el método en el b^{avo} conjunto *bootstrap* de entrenamiento con el fin de conseguir $\widehat{h^{*b}}(x)$ y, finalmente, el promedio de todas las predicciones, para obtener:

$$\widehat{h_{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{h^{*b}}(x) \qquad (6)$$

A esto se denomina *Bagging* (James, Witten, Hastie y Tibshirani, 2013).

2.4 Random Forest

Random Forest es una combinación de árboles predictivos (clasificadores débiles); es decir, una modificación del *Bagging*, el cual trabaja con una colección de árboles incorrelacionados y los promedia (Hastie, Friedman y Tibshirani, 2001), en el cual se tiene que cada árbol depende de los valores de un vector aleatorio de la muestra de manera independiente y con la misma distribución de todos los árboles en el bosque. La generalización de error para los bosques converge a un límite en cuanto el número de árboles en el bosque sea grande. El error de generalización de un bosque de árboles de clasificación depende de la fuerza de los árboles individuales en el bosque y la correlación entre ellos. El uso de una selección aleatoria de características para dividir cada nodo produce tasas de error que se comparan favorablemente al algoritmo AdaBoost (Freund y Schapire, 1996), pero son más robustos con respecto al ruido (clasificador fuerte). Estimaciones internas supervisan el error, la fuerza y la correlación. Además, estos se utilizan para mostrar la respuesta al aumento del número de características utilizadas

en la división. Los cálculos internos se utilizan, asimismo, para medir la variable de importancia. Las ideas son también aplicables a la regresión.

El elemento común en todos estos procedimientos es que para el k-ésimo árbol se genera un vector aleatorio Θ_k, independiente de los últimos vectores aleatorios $\Theta_1, \ldots, \Theta_{k-1}$ pero con la misma distribución; y un árbol se desarrolla usando el conjunto de entrenamiento y de Θ_k, lo que resulta en un clasificador donde $h(x, \theta_k)$ es un vector de entrada.

Como se ha señalado líneas arriba, el método *Random Forest* se basa en un conjunto de árboles de decisión, es decir, una muestra entra al árbol y es sometida a una serie de test binarios en cada nodo, llamados *split*, hasta llegar a una hoja en la que se encuentra la respuesta. Esta técnica puede ser utilizada para dividir un problema complejo en un conjunto de problemas simples.

En la etapa de entrenamiento, el algoritmo intenta optimizar los parámetros de las funciones de *split* a partir de las muestras de entrenamiento.

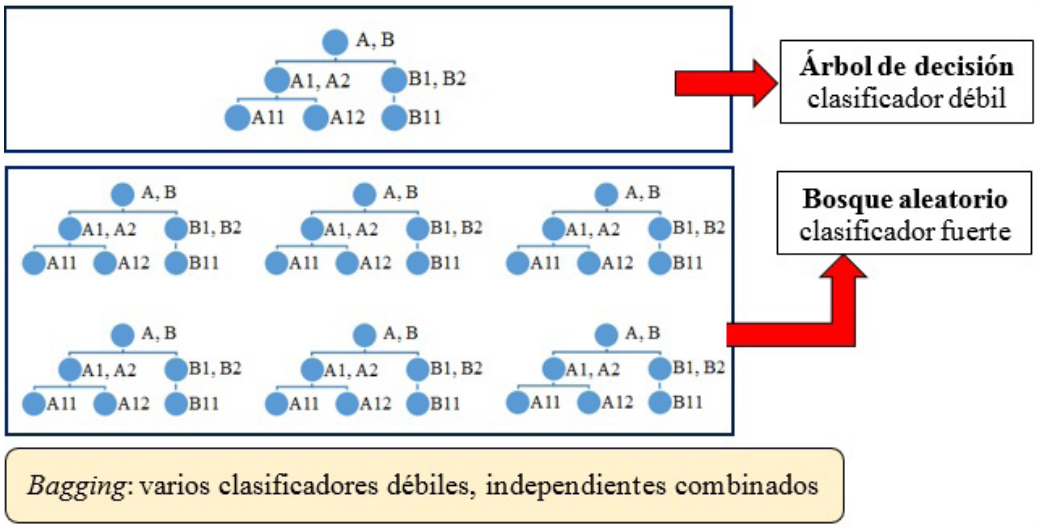
$$\theta_k^{*} = argmax_{\theta\ j \epsilon \tau_j} I_j$$

Para ello se utiliza la siguiente función de ganancia de información:

$$I_j = H(j) - \sum_{i \in 1,2} \frac{|S_j^i|}{|S_j|} H(S_j^i) \qquad (7)$$

Donde S representa el conjunto de muestras que hay en el nodo por dividir, y S^ison los dos conjuntos que se crean de la escisión. La función mide la entropía del conjunto, y depende del tipo de problema que abordamos (Breiman, 2001).

Figura 2. Árbol de decisión vs Bosque aleatorio



Elaboración propia

3. Metodología

Formalmente, se puede definir un bosque aleatorio como un clasificador que consiste en una colección de clasificadores estructurada de árboles $\{h(x,\theta_k), k=1, ...\}$ donde el $\{\theta_k\}$son vectores aleatorios idénticamente distribuidos y cada árbol tiene una participación en la unidad de la clase más popular en el vector de entrada x.

3.1 Criterio de convergencia

Dado un conjunto de clasificadores $h_1(x)$, $h_2(x)$, …, $h_k(x)$ y con el conjunto de entrenamiento se extraerá al azar con la distribución del vector aleatorio Y, X. Se define la función de margen como:

$$mg(X,Y) = av_k I(h_k(X) = Y) - max_{j \neq Y} av_k I(h_k(X) = j) \quad (8)$$

Donde $I(.)$ es la función indicadora (ganancia de información). Cuanto mayor sea el margen, más confianza habrá en la clasificación. La generalización de error viene dada por:

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \quad (9)$$

Los subíndices X, Y indican que la probabilidad está sobre el espacio X, Y.

Para un gran número de árboles se presenta una convergencia para el error, lo cual se demuestra por la "Ley fuerte de los grandes números" y la misma estructura del árbol, según se puede apreciar en el siguiente teorema.

Teorema 1

A medida que el número de árboles aumenta, casi seguramente todas las secuencias θ_1, θ_2, ... ; PE^*convergen a:

$$P_{X,Y}(P_\theta(h(X,\theta) = Y) - max_{j \neq Y} P_\theta(h(X,\theta) = j) < 0) \ldots\ldots (10) \qquad (10)$$

Este resultado explica por qué los bosques aleatorios no sobreajustan a medida que se añaden más árboles, pero producen un valor límite de la generalización de error.

3.2 La fuerza y la correlación

Para los bosques aleatorios, un límite superior se puede derivar para la generalización de error en función de dos parámetros, que son medidas de forma exacta, los clasificadores individuales y de la dependencia entre ellos. La interacción entre estos dos da la base para entender el funcionamiento de *Random Forest*.

Definición 1. La función marginal de un *Random Forest* es:

$$mr(X,Y) = P_\theta(h(X,\theta) = Y) - max_{j \neq Y} P_\theta(h(X,\theta) = j) \qquad (11)$$

Y la fuerza del conjunto de clasificadores $[h(X, \theta)]$es:

$$s = E_{X,Y} mr(X,Y) \qquad (12)$$

Asumiendo s ≥ 0 la desigualdad de Chebyshev

$$PE^* \leq var(mr)/s^2 \qquad (13)$$

Una expresión más reveladora de la varianza de *mr* se deriva de la siguiente:

$$\hat{j}(X,Y) = argmax_{j \neq Y} P_\theta(h(X,\theta) = j) \tag{14}$$

Por tanto:

$$\begin{aligned} mr(X,Y) &= P_\theta(h(X,\theta) = Y) - P_\theta(h(X,\theta) = \hat{j}(X,Y)) \\ &= E_\theta[I(h(X,\theta) = Y) - I\big(h(X,\theta) = \hat{j}(X,Y)\big)] \end{aligned} \tag{15}$$

Definición 2. La función marginal en la rama es:

$$rmg(\theta, X, Y) = I(h(X,\theta) = Y) - \mathrm{I}(h(X,\theta) = \hat{j}(X,Y)) \tag{16}$$

Así $mr(X, Y)$ es la esperanza de rmg(θ, X, Y) con respecto a Θ. Para cualquier función f de la identidad

$$[E_\theta f(\theta)]^2 = E_{\theta,\theta'} f(\theta) f(\theta') \tag{17}$$

Donde se tiene $\theta\theta'$ es independiente con la misma distribución, lo cual implica:

$$mr(X,Y)^2 = E_{\theta,\theta'} rmg(\theta, X, Y) rmg(\theta', X, Y) \tag{a}$$

Usando (I):

$$\begin{aligned} var(mr) &= \bar{\rho}(E_\theta sd(\theta))^2 \\ &\leq \bar{\rho} E_\theta var(\theta) \end{aligned} \tag{b}$$

Donde $\bar{\rho}$ es el valor medio de la correlación; es decir,

$$\bar{\rho} = E_{\theta,\theta'}(\rho(\theta\theta')sd(\theta)sd(\theta'))/E_{\theta,\theta'}(sd(\theta)sd(\theta'))$$

Se escribe:

$$E_\theta = var(\theta) \leq E_\theta \left(E_{X,Y} rmg(\theta, X, Y)\right)^2 - s^2 \leq 1 - s^2 \quad \text{(c)}$$

Juntando (a), (b) y (c)

Definición 3. Una cota superior para la generalización de error viene dada por:

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2 \qquad (18)$$

Esto demuestra que los dos ingredientes que intervienen en el error de generalización de los bosques al azar son la fuerza de los clasificadores individuales en el bosque, y la correlación entre ellos en términos de las funciones de margen de la rama. La relación c/s^2 es la correlación dividida por el cuadrado de la fuerza. Esta relación será mejor cuanto más pequeño sea su valor.

Definición 4. La razón c/s^2 de un clasificador *Random Forest* se define como:

$$c/s^2 = \bar{\rho}/s^2 \qquad (19)$$

La correlación $\bar{\rho}$ se da entre $I(h(X, \theta) = Y)$ y $I(h(X, \theta') = Y)$. En particular, toman valores entre 1 y –1, entonces Y:

$$\bar{\rho} = E_{\theta,\theta'}[\rho(h(.,\theta), h(.,\theta'))] \qquad (20)$$

De modo que $\bar{\rho}$ es la correlación entre dos miembros diferentes del bosque promediada sobre la distribución de $\theta\theta$.

3.3 Importancia de variables

3.3.1 *Mean Decrease Accuracy* (MDA)

Para realizar el modelo *Random Forest* se usael *bootstrap*, con el fin de entrenar cada árbol que se agregará en el bosque. En dicho proceso se deja aproximadamente un tercio de los casos de la muestra; a los casos que no son considerados para entrenar el árbol se les llama *out-of-bag* (OOB). Con ellos se puede estimar un error insesgado de clasificación y también se pueden utilizar para hacer una estimación de la importancia de las variables. El funcionamiento de esta lógica es la siguiente: primero se escoge el error de clasificación *out-of-bag*, después se toma una variable al azar y se permutan sus valores dentro de los datos de entrenamiento, ocasionando que dicha variable escogida descorrelacione lo aprendido por el modelo. Luego se vuelve a calcular el error OOB, para luego compararlo con el error calculado inicialmente. En consecuencia, por lógica, si el error cambia, se afirma que dicha variable es importante. Este proceso se repite con todas las variables y luego estas se ordenan de acuerdo a los cambios que produjeron cada una en los errores OOB.

3.3.2 *Mean Decrease Gini* (MDG)

Otra manera de estimar la importancia de las variables en el modelo *Random Forest* es utilizando el criterio de Gini. Este consiste en seleccionar la variable en cada partición en la construcción de los árboles y que corresponde a una disminución de esta medida. La importancia de una variable en un árbol se mide como la suma de los decrementos atribuidos a esa variable y la importancia final, como la media en todos los árboles. En el acápite 2.2.1 se definió el índice de Gini, que cuando es usado como función de impureza se le conoce como *Gini Importance* o *Mean Decrease Gini*. Esta medida es usada en el programa Python como medición de la importancia de las variables.

4. Resultados

La data VOICE contiene medidas y resultados de la clasificación de 3168 muestras de voz grabadas, recogidas de hablantes de sexo masculino y femenino. Las muestras de voz son preprocesadas mediante análisis acústico en R usando los paquetes *seewave* y *tuneR*, con un rango de frecuencia de 0 –280 Hz analizados. Las medidas incluidas en el archivo <voice.csv> corresponden solo a propiedades acústicas de la voz y el habla. La clasificación concierne a una categoría de género al que corresponden estas características.

4.1 Muestra

Se tienen 3168 muestras de voz grabadas a hombres y mujeres.

4.2 Descripción de los datos

Las variables independientes que se tomarán en cuenta para el análisis del modelo de *Random Forest* serán las siguientes:

Freq_media: Frecuencia media (en kHz)

Sd_frecuencia: Desviación estándar de las frecuencias

Median: Frecuencia de la mediana (en kHz)

Q25: Primer cuartil (en kHz)

Q75: Tercer cuartil (en kHz)

Asimetria: Asimetría.

Entropia_espectral: Entropía espectral

Plenitud_espectral: Plenitud espectral

Frecuencia_modo: frecuencia de moda

Centroide_de_frecuencia: Centroide de frecuencia

Meanfun: Promedio de la frecuencia fundamental medida a través de la señal acústica

Minfun: Frecuencia fundamental mínima medida a través de la señal acústica

Maxfun: Frecuencia fundamental máxima medida a través de la señal acústica

Meandom: Promedio de la frecuencia dominante medida a través de la señal acústica

Mindom: Mínimo de frecuencia dominante medida a través de la señal acústica

Maxdom: Máximo de frecuencia dominante medida a través de la señal acústica

Modindx: Índice de modulación. Calculado como la diferencia absoluta acumulada entre mediciones adyacentes de frecuencias fundamentales divididas por el rango de frecuencias.

Tomando en cuenta las variables independientes, la variable respuesta (clasificadora) será:

Género: clasificación de las propiedades acústicas de la voz y el habla (femenino y masculino)

4.3 Análisis Random Forest

4.3.1 Análisis mediante el uso del programa R

- **Construcción del modelo (*data train*)**

  Se presenta el resumen del modelo entrenado con el 70 % de los datos, con el cual se entrenó un modelo *Random Forest* con 1000 árboles. En cada árbol se tomaron cuatro variables para cada división.

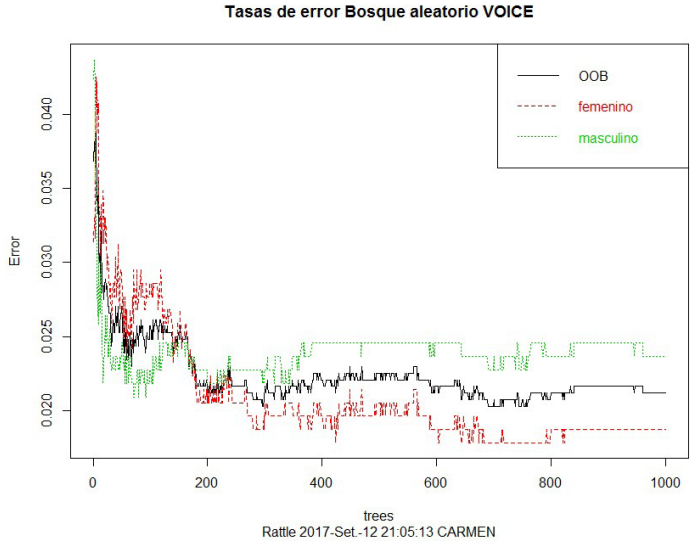
Figura 3. Resumen modelo *random forest*

```
Call:
 randomForest(formula = genero ~ ., 
              data = crs$dataset[crs$sample, c(crs$input, crs$target)], 
              ntree = 1000, mtry = 4, importance = TRUE, replace = FALSE, na.action = randomForest::na.roughfix)

               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 2.12%
Confusion matrix:
          femenino masculino class.error
femenino      1098        21  0.01876676
masculino       26      1072  0.02367942
```

Elaboración propia – Reporte R

La estimación de la tasa de error OOB es calculada a partir de las observaciones fuera de la bolsa. La estimación del error sugiere que cuando el modelo sea aplicado a nuevas observaciones, las respuestas tendrán un error de 2,12 %; también puede decirse que el modelo es 97,88 % exacto; entonces, el modelo es razonablemente bueno.

Figura 4. Tasas de error *random forest*



Elaboración propia – Reporte R

Para cada clase de la variable clasificadora también se halla el error de clasificación. Se observa que para las muestras de voz clasificados como femenino presentan un error de clasificación de tan solo el 1,9 %, esto es, nuestro modelo *Random Forest* clasifica muy bien a las voces femeninas; mientras que para el caso de las voces clasificadas como masculinas se presenta un error de clasificación mayor (2,4 %) respecto a las voces clasificadas como voces pertenecientes a una mujer.

- **Matriz de confusión**

Seguido de la estimación del error OOB para nuestra data VOICE, es necesario interpretar la matriz de confusión. En ella se registran los valores predichos por el modelo *Random Forest* (fila) versus los observados (columna).

Para las voces femeninas solo 21 registros fueron clasificados como voces masculinas; es decir, fueron mal clasificados; mientras que de las voces masculinas 26 fueron mal clasificadas por el modelo.

Figura 5. Matriz de confusión

```
Confusion matrix:
          femenino masculino class.error
femenino      1098        21  0.01876676
masculino       26      1072  0.02367942
```

Elaboración propia – Reporte R

- **Importancia de variables**

  Como se vio en teoría, existen dos medidas para hallar la importancia de variables en *Random Forest*. En la figura 6 se tiene ambas medidas: *Mean Decrease Accuracy* (MDA) y *Mean Decrease Gini* (MDG).

Figura 6. Variable de importancia según medidas de importancia

Importancia de variable

	femenino	masculino	MeanDecreaseAccuracy	MeanDecreaseGini
meanfun	62.96	102.58	107.48	295.60
Q25	30.92	51.10	53.52	142.60
sd_frecuencia	22.58	23.96	32.13	79.91
Q75	21.63	22.94	30.34	11.47
plenitud_espectral	21.45	16.75	26.11	28.05
centroid_de_frecuencia	17.10	18.01	25.14	16.39
entropia_espectral	21.67	12.40	24.64	33.74
asimetria	19.05	18.49	24.03	10.16
freq_media	15.58	18.42	23.11	16.38
dfrange	15.93	15.31	21.97	6.34
modindx	18.59	13.52	21.93	4.99
maxdom	17.50	14.36	21.71	7.01
frecuencia_modo	15.26	20.61	21.33	16.53
meandom	17.52	14.50	21.25	5.81
median	11.48	17.54	20.05	9.74
minfun	15.51	16.18	20.04	6.82
mindom	10.89	14.55	16.18	5.21
maxfun	13.82	10.22	16.11	3.97

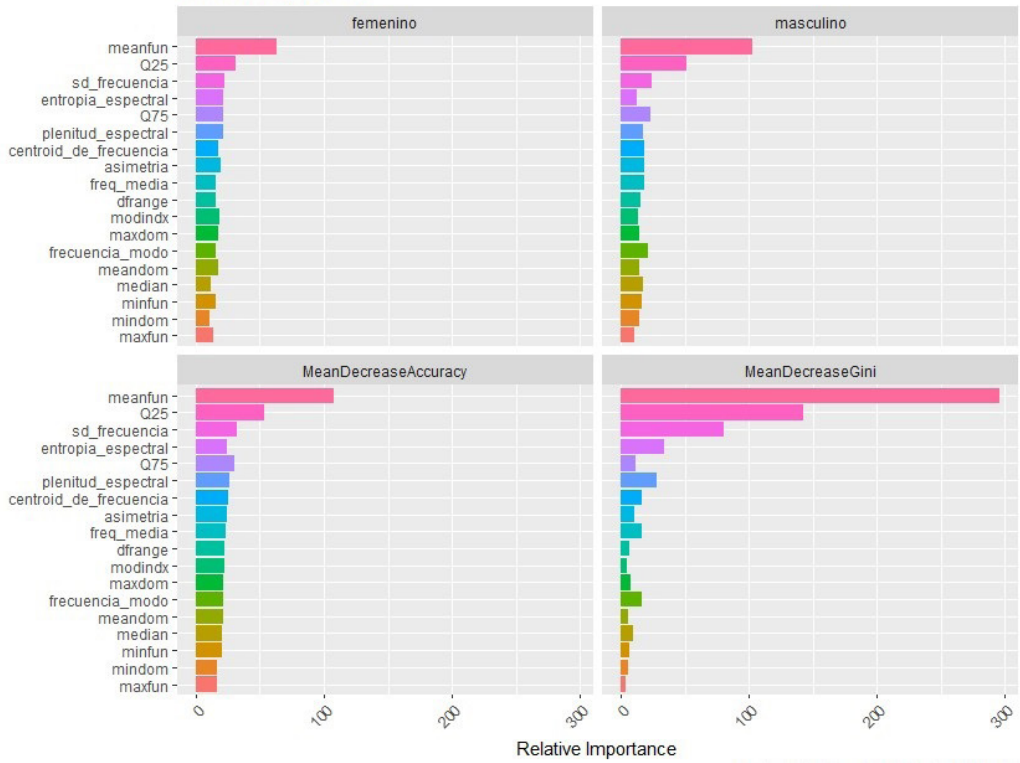
Elaboración propia – Reporte R

Para una mejor visión de estas dos medidas, en la figura 7 se tiene la medida de importancia *Mean Decrease Accuracy*; claramente se observa que las cinco primeras variables más importantes son: el promedio de la frecuencia fundamental medida a través de la señal acústica, seguido del primer cuartil (en kHz), la desviación estándar de las frecuencias, tercer cuartil (en kHz) y la plenitud espectral.

En la figura 7, en el lado derecho, se tiene a las variables importantes calculadas con la medida *Mean Decreace Gini*, de donde se aprecia que las primeras tres variables y la quinta variable coinciden con la medida de importancia mediante

Mean Decrease Accuracy. Mientras que la cuarta variable importante con el método de MDG es la entropía espectral.

Figura 7. Variable de importancia según medidas de importancia



Elaboración propia – Reporte R

- **Evaluación del modelo** (*data test*)

  Para hacer la validación del modelo, se procede a generar la matriz de confusión con la *data test* o data de validación. En ella se tiene que el 1,1 % de las voces femeninas fue mal clasificado por el modelo como voces masculinas. Sin embargo, el 49,7 % fuen bien clasificado por el modelo; para el caso de las voces masculinas el 47,6 % fue bien clasificado.

Figura 8. Matriz de confusión con la data test

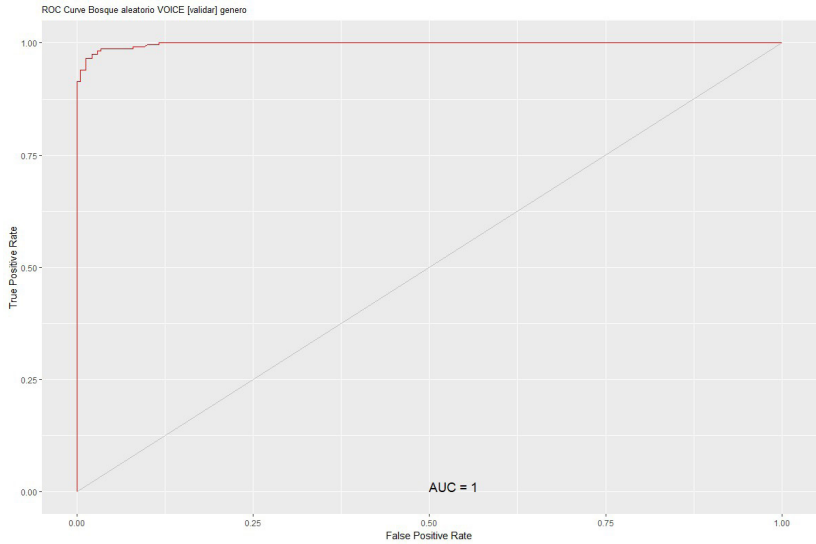
```
Matriz de error para el modelo Bosque aleatorio en VOICE [validar] (cuentas):

           Predicted
Actual      femenino masculino Error
  femenino       236         5   2.1
  masculino        8       226   3.4

Error matrix for the Bosque aleatorio model on VOICE [validar] (proportions):

           Predicted
Actual      femenino masculino Error
  femenino      49.7       1.1   2.1
  masculino      1.7      47.6   3.4

Overall error: 2.7%, Averaged class error: 2.75%
```

Elaboración propia – Reporte R

También se tiene el error promedio de la clasificación, el cual es de 2,75 %. Esto significa que, en promedio, el modelo *Random Forest* comete un error del 2,75 % al clasificar. El área bajo la curva o AUC (*Area Under the Curve*) es aquella que estima la capacidad de clasificar las voces femeninas y masculinas para el modelo *Random Forest*.

Para el conjunto de datos se ha obtenido un área bajo la curva de 1, del cual se podría decir que el modelo *Random Forest* es perfecto ya que clasifica al 100 % las voces femeninas como femeninas y al 100 % también para el caso de las voces masculinas.

Figura 9. Gráfica de la curva ROC



Elaboración propia – Reporte R

Ed. n.°10 // Enero-diciembre 2017 // ISSN 1993-4912

4.3.2 Análisis mediante el uso del programa Python

Trabajando con el programa Python se tiene los siguientes resultados:

- **Construcción del modelo**

  Para la construcción del modelo de *Random Forest* se tomó en cuenta lo siguiente: el número de estimadores son 1000 (árboles de clasificación) y se seleccionarán cuatro variables aleatoriamente para la construcción de cada árbol de decisión.

Figura 10. Construcción del modelo random forest

```
In [448]: ### MODELO RANDOM FOREST ####
          modelo = RandomForestClassifier(
           random_state      = 1,   # semilla inicial de aleatoriedad del algoritmo
           n_estimators      = 1000, # cantidad de arboles a crear
           min_samples_split = 4,   # cantidad minima de observaciones para dividir un nodo
           oob_score=True
          )
          modelo.fit(x, y )

Out[448]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_split=1e-07, min_samples_leaf=1,
                      min_samples_split=4, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=1, oob_score=True, random_state=1,
                      verbose=0, warm_start=False)
```

Elaboración propia – Reporte R

Para determinar la importancia de variables, es decir, para obtener las variables más importantes para la clasificación de las voces, se usó el método de la importancia de variables de Gini, como se puede observar en la figura 11.

Figura 11. Variables importantes

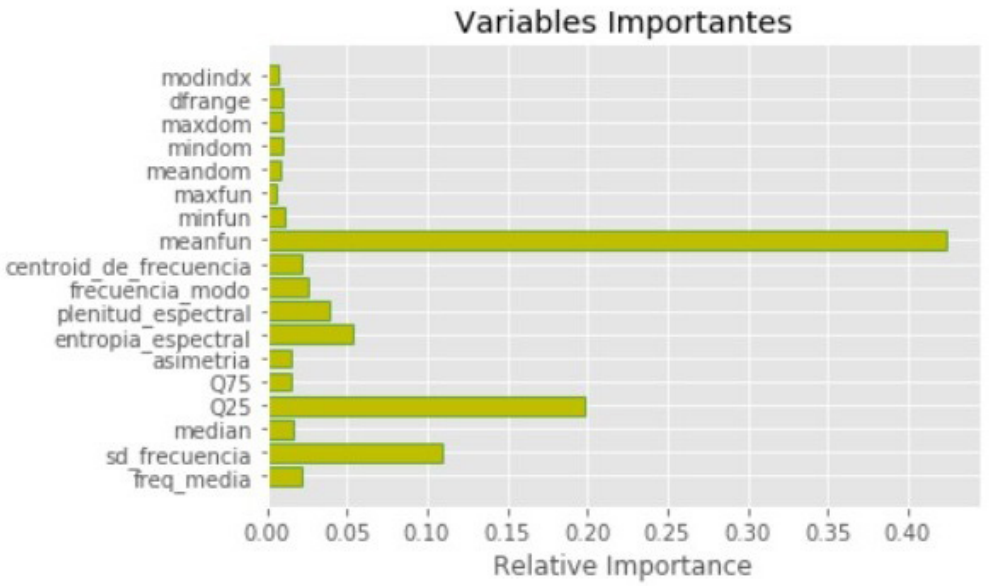
```
In [449]: var_imp=pd.DataFrame({
              'feature':headers,
              'v_importance':modelo.feature_importances_.tolist()
          })
          print (var_imp.sort_values(by='v_importance',ascending=False))
```

	feature	v_importance
10	meanfun	0.423298
3	Q25	0.198856
1	sd_frecuencia	0.109548
6	entropia_espectral	0.053267
7	plenitud_espectral	0.039427
8	frecuencia_modo	0.025921
0	freq_media	0.022051
9	centroid_de_frecuencia	0.021465
2	median	0.015759
4	Q75	0.015260
5	asimetria	0.014470
11	minfun	0.010845
14	mindom	0.009952
16	dfrange	0.009365
15	maxdom	0.009098
13	meandom	0.008376
17	modindx	0.007147
12	maxfun	0.005895

Elaboración propia – Reporte R

Las cinco primeras variables en importancia son el promedio de la frecuencia fundamental medida a través de la señal acústica, con el mayor valor de importancia de variables (0,42), seguida del primer cuartil (0,20), la desviación de la frecuencia (0,11), la entropía espectral (0,05) y la plenitud espectral (0,04).

Figura 12. Gráfica de la importancia de variables



Elaboración propia – Reporte R

Con el modelo construido con base en la data de entrenamiento se obtiene una matriz de confusión y a la vez se calcula la precisión del modelo, teniéndose para este caso un valor de 99,5 % de precisión.

Figura 13. Precisión del modelo

```
In [452]: ## Matriz de confusion para la data train
          confusion_matrix(y_0, prediccion_train)

Out[452]: array([[1252,    4],
                 [   5, 1273]])

In [453]: prediccion_train=modelo.predict(x_0)
          ## matriz de confusion prediccion global
          sum(prediccion_train==y_0)/(y_0.count())

Out[453]: 0.99486977111286501
```

Elaboración propia – Reporte R

- **Validación del modelo**

  Para proceder con la validación del modelo, se toma el conjunto de datos "Test". Con ello se obtiene la matriz de confusión, encontrándose valores pequeños en la diagonal de los mal clasificados; esto es, se tiene un error de 2,5 % al hacer las clasificaciones con el modelo de *Random Forest* propuesto, el cual es un error considerable para un modelo de clasificación.

Figura 14. Validación del modelo

```
In [454]: ##### VALIDANDO EL MODELO CON EL CONJUNTO DE DATOS TEST

In [460]: ### validacion del modelo (data_test)
          y_1=test['genero']
          x_1=test[['freq_media','sd_frecuencia','median','Q25','Q75','asimetria','entropia_espectral','plenitud_espectral',

In [461]: prediccion_test=modelo.predict(x_1)
          ## matriz de confusion prediccion global
          sum(prediccion_test==y_1)/(y_1.count())

Out[461]: 0.97476340694006314

In [462]: ## Matriz de confusion para la data test
          confusion_matrix(y_1, prediccion_test)

Out[462]: array([[326,   6],
                 [ 10, 292]])
```

Elaboración propia – Reporte R

Con el programa también se pudo estimar la importancia de variables para el modelo *Random Forest*, teniendo como resultado similar al obtenido en R. Las tres variables más importantes fueron:

i. El promedio de la frecuencia fundamental medida a través de la señal acústica

ii. El primer cuartil (en kHz), y

iii. La desviación estándar de las frecuencias

Dado que el objetivo es presentar el modelo *Random Forest*, para lo cual se usan los programas R y Python como herramientas para la obtención de resultados, se han obviado los procedimientos intermedios del Python, tales como los de la *data* de entrenamiento, dando preferencia a la presentación de sus resultados, los que son similares a los obtenidos por el programa R.

5. Conclusiones

i. Al someter los datos al modelo *Random Forest* se obtiene una buena tasa aproximada de error del 2,12 %. Esto significa que si se ingresaran nuevos casos asociados a las características de una voz, el modelo tendría un error máximo de 2,12 %, aproximadamente.

ii. Al analizar la importancia de variables, se observó que los resultados obtenidos mediante la medida de importancia MDA y MDG no difieren mucho, ya que en ambos casos se determinó a la variable "promedio de la frecuencia fundamental medida a través de la señal acústica" como la más importante, tanto en el programa R como Python, seguida de primer cuartil (en kHz), la desviación estándar de las frecuencias, tercer cuartil (en kHz) y la plenitud espectral, entre las principales.

iii. El modelo *Random Forest* dio buenos resultados en cuanto a buena clasificación y, sobre todo, para determinar la importancia de las variables, lo cual sirve mucho para hacer una posterior depuración de variables y un análisis más preciso.

Referencias

Ali, J., Khan, R., Ahmad, N., y Maqsood, I. (2012). Random forests and decision trees. *IJCSI International Journal of Computer Science Issues*, 9(5), 272-278. Recuperado de http://ijcsi.org/papers/IJCSI-9-5-3-272-278.pdf

Alpaydin, E. (2010). *Introduction to machine learning* (2.ª ed.). Massachusetts, Estados Unidos: MIT Press.

Breiman, L., Friedman, J., Stone, C., y Olshen, R. (1984). *Classification and regression trees*. California, Estados Unidos: Wadsworth, Inc.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5-32. DOI: 10.1023/A:1010933404324

Freund, Y., y Schapire, R. (1996). Experiments with a New Boosting Algorithm. En *Thirteenth International Conference on Machine Learning*, 148-156. Recuperado de https://webcourse.cs.technion.ac.il/236756/Spring2009/ho/WCFiles/FruendSchapireAdaboostExperiments.pdf

Hastie, T., Friedman, J., y Tibshirani, R. (2001). *The Elements of Statistical Learning*. Nueva York, Estados Unidos: Springer New York. DOI: 10.1007/978-0-387-21606-5

James, G., Witten, D., Hastie, T., y Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Nueva York, Estados Unidos: Springer New York / Heidelberg Dordrecht London. DOI: 10.1007/978-1-4614-7138-7

ANEXO

Programa R

El programa estadístico R es un lenguaje de programación que permite llevar a cabo procedimientos estadísticos desde los más elementales hasta los más complejos; gracias a esto puede ser utilizado en diversos campos como la minería de datos. En el momento de redactar el código del programa, se emplearon funciones implementadas en los distintos *packages* que ofrece R. Resulta importante mencionar la utilidad de las librerías más destacadas.

Se procede a indicar que dentro del código se halló una muestra de entrenamiento (70 % de los datos de manera aleatoria) y una de validación (30 % de los datos).

```
# muestra de entrenamiento y validación del modelo
ind <- sample(2,nrow(VOICE),replace=TRUE,prob=c(0.7,0.3))
train <- VOICE[ind==1,]
test <- VOICE[ind==2,]
```

Luego de obtener la muestra de entrenamiento se utiliza la librería *RandomForest()*: esta librería estima un modelo de *Random Forest*, dado que existe una *data* en la cual se tiene una variable respuesta categórica y variables independientes bien definidas e identificadas.

```
# libreria RANDOM FOREST
library(randomForest)
```

Después de ello, se procedió a modelar los datos y hallar el error OOB tanto para el modelo general como para cada clase. Finalmente se pidió la gráfica de la importancia de las variables.

```
# importancia de variables
VOICE_rf <- randomForest(genero~.,data=train,ntree=1000,mtry=4, proximity=TRUE)
table(predict(VOICE_rf),train$genero)
print(VOICE_rf)
importance(VOICE_rf)
varImpPlot(VOICE_rf, col='violetred4',n.var = 10,main = 'Variables Importantes')
```

```
# el modelo con la data de entrenamiento
VOICE_Pred<-predict(VOICE_rf,newdata=test)
```

```
# Gráfica de la curva ROC
library(ROCR)
# Predicción
predict.rpart <- predict(VOICE_rf,test,type = "prob")[,2] #prob. clase=yes
predict.rocr <- prediction (predict.rpart,test$genero)
perf.rocr <- performance(predict.rocr,"tpr","fpr")
# Gráfico
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
plot(perf.rocr,type='o',col='darkmagenta', main = paste('Area Bajo la Curva =',round(auc,2)))
abline(a=0, b= 1,col='navy', lwd=2)
```

Programa Python

Python es un lenguaje de programación creado originalmente para uso estadístico. Aun cuando en nuestro medio Python no es un programa común en estadística, una de las razones por las que se eligió este programa ha sido la diversidad de librerías para trabajar el *machine learning*. En este caso, la librería usada para este fin ha sido la librería *sklearn.ensemble*, que contiene el algoritmo del clasificador *Random Forest* y está diseñada para trabajar junto a dos librerías más: *Numpy* y *Pandas*. *Numpy* es la librería necesaria para trabajar con vectores, matrices, *arrays*, y cuenta con funciones de alto nivel matemático:

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
```

Mientras que *Pandas* es una librería de Python destinada al análisis de datos que proporciona unas estructuras de datos flexibles y que permite trabajar con ellos de forma muy eficiente:

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')
```

Ventajas de trabajar con Python:

- **Simplificado y rápido**. Simplifica la programación, ya que su lenguaje de programación se asemeja al lenguaje natural.
- **Comunidad**. La comunidad de Python actualiza el lenguaje, y estas actualizaciones se realizan de manera democrática.
- **Librerías incluidas**. Las principales librerías que se requieren se encuentran implementadas dentro del código.