INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA
ESCUELA DE POSTGRADO

Predicción de pacientes con enfermedades reumáticas en México

AUTOR: Roberts Karen Natalí (Leg. Nº 104275)
TUTORES: Pelaez Ingris y Ramele Rodrigo

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE ESPECIALISTA EN CIENCIAS DE DATOS

BUENOS AIRES
SEGUNDO CUATRIMESTRE, 2019

Índice

1. Introducción

La prevalencia de la patología reumatológica en los países desarrollados tiene un perfil conocido basado en estudios de grandes cohortes. El impacto de estas enfermedades en la calidad de vida se traduce en una alta prevalencia de discapacidad asociada y costos económicos. A pesar de esto, en los países en vías de desarrollo las enfermedades reumáticas no son consideradas una prioridad en salud pública. Ante esta falencia de datos la Organización Mundial de la Salud (OMS) ha diseñado una estrategia basada en un cuestionario sencillo que identifica a individuos con síntomas reumáticos a través de una entrevista con encuestadores entrenados. Incluye preguntas relacionadas con los síntomas (dolor y rigidez), discapacidad, tratamiento y adaptación al problema.

A partir de los datos recolectados por el cuestionario el propósito del presente estudio es caracterizar a los individuos con enfermedades reumáticas y poder predecir de manera automática si un individuo está en presencia de enfermedades músculo-esqueléticas a partir del llenado del formulario para la detección temprana y remitirle al especialista para el tratamiento, de esta manera se puede prevenir discapacidad secundaria a estas enfermedades.

2. Estado del arte

Las enfermedades reumáticas se encuentran entre las dolencias que, con mayor frecuencia, afectan a los seres humanos; son un conjunto de más de 200 enfermedades con perfil demográfico, genético y clínico diverso que afectan al aparato locomotor, es decir, a las articulaciones, músculos, tendones y ligamentos y también incluyen las enfermedades metabólicas del hueso (osteoporosis, osteomalacia, Paget, etc.) [1]. Se caracterizan por dolor, inflamación crónica y persistente, alteración de la capacidad funcional y deterioro de la calidad de vida; estos síntomas son muy frecuentes en la población [2]. Junto con las infecciones respiratorias y enfermedades cardiovasculares, constituyen las causas más frecuentes de consulta en Atención Primaria [3]. Uno de los principales órganos donde asientan las enfermedades reumáticas es la articulación. En la Figura 1.1 se muestra gráficamente las partes de la articulación, la misma está formada por la unión de dos huesos y permite la movilidad del esqueleto. Las estructuras que permiten esta unión son la cápsula articular y los ligamentos. En la parte interior de la cápsula, las articulaciones presentan la membrana sinovial que segrega un fluido, el líquido sinovial, que actúa como lubricante facilitando, por su viscosidad, el roce de los extremos óseos que se articulan. Otro componente muy importante de las articulaciones es el cartílago articular, que recubre las superficies óseas articulares y actúa a modo de almohadilla en los movimientos de desplazamiento entre las mismas [3].

Las enfermedades del aparato locomotor son muy frecuentes y se pueden manifestar de distintas maneras. En general, los síntomas que con más frecuencia notan los pacientes son dolor, deformidades articulares y dificultad para moverse. La causa de estas enfermedades es multifactorial, incluyendo factores genéticos, ambientales y endógenos fundamentalmente. Las principales medidas que ayudan a prevenir la aparición de enfermedades reumáticas son: una buena dieta, evitar la obesidad, evitar el tabaco y hacer ejercicio de forma regular.

Las enfermedades reumáticas se pueden clasificar en los siguientes grupos generales:

- **Patología degenerativa**. Se refiere a la artrosis, que puede afectar a cualquier articulación (mano, rodilla, cadera, etc.) o a la columna espinal. La artrosis es un proceso que conlleva el

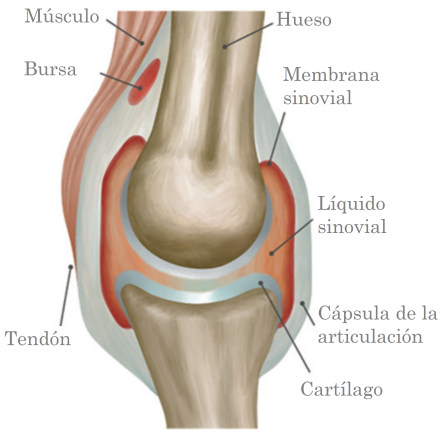


Figura 1.1: Partes de la articulación

desgaste y degeneración articular, sin necesidad de que exista una inflamación; el componente articular que inicialmente se lesiona es el cartílago. No existe tratamiento curativo pero existen medidas para frenar su progresión y medicamentos que mejoran los síntomas. En casos graves se puede operar.

- **Patología no inflamatoria o extraarticular**. Este grupo incluye a los procesos que afectan a las partes blandas alrededor de las articulaciones, que provocan principalmente las tendinitis, bursitis o el atrapamiento de nervios periféricos como el síndrome del túnel carpiano.

- **Patología inflamatoria**. En este caso se incluyen las enfermedades que producen inflamación articular o artritis, como la gota o la artritis reumatoide. Sin embargo, algunas de ellas, denominadas enfermedades del colágeno o autoinmunes, tienen características propias y pueden provocar afectación de órganos internos, como la propia artritis reumatoide, el lupus eritematoso, el síndorme de Sjögren o las vasculitis, entre otras. La artritis es un proceso en el que se inflaman las articulaciones. Una parte importante de las mismas se produce como consecuencia de mecanismos autoinmunes. En ellos, probablemente como consecuencia de la acción de un agente infeccioso sobre un organismo predispuesto genéticamente, se pone en marcha un proceso inflamatorio crónico con potencial erosivo y destructivo articular. El prototipo de esta situación sería la artritis reumatoide, para estos casos existen tratamientos farmacológicos de fondo que son eficaces para frenar su progresión, antiinflamatorios, corticoides y fármacos antireumáticos modificadores de la enfermedad (FARME) pueden ser sintéticos (metotrexato, leflunomida, sulfasalazina, entre otros) o biológicos (etanercept, infliximab, adalimumab y otros). Para los casos de artritis infecciosas se utilizan antibióticos.

- **Patología metabólica**. La osteoporosis y la enfermedad de Paget son las enfermedades principales de este apartado. La osteoporosis es la enfermedad ósea más frecuente; se produce por la pérdida de masa ósea y la alteración de la disposición de las trabéculas óseas. Como consecuencia de ello, es decir, de la alteración de la cantidad y de la calidad del hueso, nuestro esqueleto se vuelve más frágil y aparecen las fracturas espontáneas o ante mínimos golpes o traumatismos (por ejemplo, caerse al suelo desde la posición de estar de pie).

- **Miscelánea**. En este grupo podemos incluir el resto de enfermedades reumáticas, como la fibromialgia, enfermedades hereditarias del colágeno, infecciones, los tumores, etc.

El dolor musculoesquelético es un problema universal, pero el dolor es difícil de medir, es por esto que la Organización Mundial de la Salud y la Liga Internacional de Asociaciones de Reumatología (ILAR) desarrollaron el Programa de salud pública Orientado a la Comunidad para el Control de Enfermedades Reumáticas (COPCORD, por sus siglas en inglés) como una estrategia epidemiológica diseñada para la identificación, prevención y control de las enfermedades reumáticas en países en vías de desarrollo. El programa COPCORD consta de tres etapas, la primera consiste en la administración de un cuestionario simple casa por casa por parte de trabajadores de atención primaria de la salud a toda una comunidad, en la Fase 2 un profesional de salud evalúa los respondedores positivos de la Fase 1 con un cuestionario más detallado y por último, en la Fase 3, los sujetos con alguna posible enfermedad reumática son examinados por un médico especializado en reumatología. En esta etapa, se hace un diagnóstico en el campo clínico y se sugiere el asesoramiento sobre la prevención y la posible intervención de atención médica [4, 5].

La estrategia COPCORD permite identificar la frecuencia de estas enfermedades, su distribución y potenciales factores de intervención como estrategias de salud pública. Se ha implementado en varios países de Latinoamérica: Argentina, México, Guatemala, Cuba, Perú, Venezuela, Brasil y Ecuador. También se han realizado estudios en poblaciones indígenas de Venezuela (Warao, Kariña y Chaima), México (Mixteco, Maya-Yucateco y Rarámuri) y Argentina (Qom). La prevalencia, de enfermedades reumática,s en estos países ha sido reportada entre el 17 y 50 %. La osteoartritis (OA) y la artritis reumatoide (AR) han sido las enfermedades más frecuentes [3].

Las iniciativas mundiales de salud para las poblaciones socialmente desfavorecidas se centran principalmente en las enfermedades infecciosas y prestan poca atención al hecho de que las enfermedades no transmisibles están aumentando tanto en los países de ingresos bajos/medios como en los países desarrollados. Este impacto es mayor en las poblaciones indígenas, que son invisibles para programas de salud y se enfrentan a un acceso limitado a la atención de salud [6]. Los estudios epi-

demiológicos son la base para la determinación del impacto económico de las enfermedades a través de la estimación del número de individuos afectados. Los costos de las enfermedades reumáticas, en países desarrollados pueden representar entre el 1 y 3 % del producto interno bruto, lo que los constituye en un problema de salud pública de alto impacto. En Latinoamérica existe poca información relacionada con la prevalencia, los costos y el impacto económico de estas enfermedades [3].

Un estudio realizado por la Sociedad Reumatológica Argentina (SAR) reveló que la mayoría de los pacientes que padecen artritis reumatoidea tarda 12 meses en acudir a un médico reumatólogo para tratar su patología [7]. Los retrasos en el diagnóstico resultan tanto del nivel de conciencia de los pacientes (que ignoran los síntomas tempranos) como del funcionamiento inadecuado del sistema de atención médica [8]. La alta prevalencia de las enfermedades reumáticas en la población general, con gran tendencia a la cronicidad y un gran riesgo de conducir a limitación funcional que impacta en lo personal, laboral y familiar, las convierten en un problema de salud de primera magnitud. Afortunadamente, hay muchas situaciones en las que esta discapacidad es en gran medida prevenible; la carga de la enfermedad puede ser disminuida por un estilo de vida más saludable junto al diagnóstico e intervención precoz [3].

Esta investigación busca, a través de modelos matemáticos, medir la probabilidad de que un individuo presente alguna enfermedad reumática a partir de los datos recolectados en el instrumento COPCORD. De esta manera, se busca contribuir a la identificación de pacientes con presencia de enfermedades reumáticas en un estadio temprano de la enfermedad. Los métodos que se utilizan para tratar al paciente cuando el diagnóstico es precoz es de menor trascendencia, en cambio, cuando la enfermedad tiene un avance importante en el curso de su evolución, lograr la remisión es más costoso y el paciente queda con secuelas.

Son muchas las aplicaciones de las técnicas de *Machine Learning* que se encuentran en la literatura médica. Varias investigaciones demuestran la importancia que han tomado estas técnicas para el diagnóstico, tratamientos y pronóstico clínico, sobre todo cuando se tienen grandes volúmenes de datos. Gracias al desarrollo de nuevas tecnologías, áreas de conocimiento y a la cantidad de datos clínicos, es de esperar que en un futuro los médicos puedan ajustar un tratamiento concreto para cada necesidad y patología.

3. Problema

Las enfermedades reumáticas son causa frecuente de discapacidad, deterioran la calidad de vida y causan alto gasto en salud; la mayoría de los pacientes que padecen alguna enfermedad reumática tardan muchos meses en acudir a un médico reumatólogo para tratar su patología. El diagnóstico precoz de estas enfermedades es de importancia clave para lograr un tratamiento eficaz y un mejor pronóstico.

4. Justificación del estudio

Por un lado, debido a la heterogeneidad de estas enfermedades y la diversidad de profesionales implicados en su atención, resulta de relevancia implementar modelos matemáticos adecuados que contribuyan a la detención automática de las enfermedades reumáticas; de esta manera se puede obtener un diagnóstico precoz y lograr un tratamiento eficaz en los pacientes identificados. Por otro lado, existe una gran falencia en la información sobre estas enfermedades en países en desarrollo, es por este motivo que este estudio pretende mostrar datos precisos sobre las prevalencias de dolor musculoesquelético, de enfermedades reumáticas, así como los factores asociados a estas.

Como población en estudio se consideró individuos nacidos en algunos estados de México dado que se dispone de los datos recolectados, en este país, a través del método COPCORD.

5. Hipótesis

El modelo diseñado y probado detecta a las personas en riesgo de tener una enfermedad reumática y una discapacidad secundaria y de esta manera, ayuda a reducir el tiempo desde el comienzo de los síntomas hasta el diagnóstico de la enfermedad reumática, implementación del mejor tratamiento y prevención de discapacidad produciendo una mejora de la calidad de vida del paciente y una disminución en los gastos de la salud pública destinado a estas patologías.

6. Objetivos

6.1 Objetivo general

Diseñar un modelo predictivo que ayude a detectar, de forma temprana, si un paciente manifiesta una enfermedad reumática.

6.2 Objetivos específicos

- Encontrar patrones para construir una regla de clasificación que permita caracterizar a las personas que presentan alguna enfermedad reumática.
- Detectar las variables que resultan más importantes en la predicción.

7. Materiales y métodos

7.1 Dataset

Los datos se recabaron en distinto estados de México, a través de encuestas casa por casa con el cuestionario COPCORD descripto en el capítulo 2.

En función de las variables clínicas recolectadas es de interés determinar, con la mayor precisión posible, si una personas padece o no alguna enfermedad reumática. La solución para este tipo de problemas se buscan a través de modelos matemáticos que permitan, a partir de un conjunto de r variables explicativas, predecir una variable de respuesta Y. Es decir, se busca resumir la relación entre $\mathbf{X} = (x^1, ..., x^r)$ e Y a través de una función f de forma tal que, dado X, sea posible predecir por medio de f(X) qué valor tomará Y.

7.2 Selección de variables y *feature engineering*

Hay una gran cantidad de algoritmos para la selección de variables, en este trabajo se utiliza el algoritmo *Boruta*. La idea es generar en cada iteración una serie de variables aleatorias también llamadas variables "sombra" a partir de los predictores originales, copiando cada uno de ellos y permutando entre sí los elementos de cada nueva columna. Se justa un modelo siguiendo el algoritmo *Random Forest*, el cual se detalla en el apartado 7.3.3 y se calcula la importancia de cada variable. Si se detecta que una variable original está por debajo de la variable sombra indica que su aporte al modelo es dudoso y por tanto se elimina. A continuación se detalla el procedimiento del algoritmo [9],

1. Replica todas las variables independientes.
2. Mezcla los valores de las variables replicadas para eliminar la correlación con las originales.
3. Combina las variables originales con las variables sombra.

4. Ejecuta el algoritmo *Random Forest* en el conjunto de datos combinado.
5. Calcula una puntuación Z como el promedio de la precisión dividido el desvío estándar de la pérdida de precisión.
6. Encuentra la puntuación máxima de Z entre los atributos sombra (MZSA).
7. Si Z obtenido es menor que MZSA entonces la variable se clasifica como "no importante" en caso contrario se la clasifica como "importante".

El proceso continua hasta que todas las variables son aceptadas, rechazadas o se alcanza un número de iteraciones límite.

A continuación se destacan algunas características relevantes que tiene el algoritmo Boruta [10]:

- Tiene en cuenta las relaciones multivariada.
- Funciona bien tanto para la clasificación como para regresión.
- Es una mejora del *Random Forest*, que es un método muy popular para la selección de variables.
- Comienza considerando que todas las variables son relevantes. La mayoría de los otros algoritmos de selección de variables siguen un método donde se basan en un pequeño subconjunto de variables que produce un error mínimo en un clasificador elegido.
- Tiene en cuenta posibles interacciones entre variables.

A partir del análisis exploratorio y previo a la aplicación del modelo, se busca crear nuevas variables a partir de los datos existentes para mejorar el rendimiento en la predicción. En caso de que una variable continua presente muchos valores perdidos se la categoriza y se consideran los valores perdidos como una categoría.

7.3 Modelos

La minería de datos puede definirse como el proceso de extraer conocimiento útil y comprensible, previamente desconocido. En la literatura se cuenta con varias definiciones de *"minería de datos"*, algunas de ellas son [11]:

- *Witten y Frank (2005)*, es el proceso de descubrir patrones en los datos que se presentan en grandes cantidades. Los patrones descubiertos deben ser significativos de manera que se permitan ventajas, por lo general, de tipo económicas.

- *Sumathi y Sivanandam (2006)*, es el proceso eficiente, no trivial, de extraer información valiosa (patrones y tendencias) de una gran colección de datos.

- *Hand et al. (2001)*, es el análisis de conjuntos de datos observados, a menudo extensos, para encontrar relaciones insospechadas y resumir los datos en forma comprensible y útil para el usuario de la información.

- *Piatetski-Shapiro (1991)*,es un conjunto de técnicas y herramientas aplicadas al proceso no trivial de extraer y presentar conocimiento implícito, previamente desconocido, potencialmente útil y humanamente comprensible, a partir de grandes conjuntos de datos, con objeto de predecir de forma automatizada tendencias y comportamientos, y describir de forma automatizada modelos previamente desconocidos.

Estas definiciones tienen en común la extracción de información potencialmente útil para explorar, a través de modelos matemáticos, patrones y tendencias que posiblemente existan entre los datos y no se visualizan con los análisis estadísticos tradicionales. Se resume que el objetivo de la minería de datos es el proceso de convertir datos en conocimiento útil a partir de modelos matemáticos.

En la Figura 1.2 se presenta el proceso de descubrimiento de conocimiento propuesto por Fayyad en 1996, el cual involucra los siguientes pasos [12]:

1. Determinar las fuentes de información útiles.

2. Diseñar el esquema de almacén de datos (del inglés *"Data Warehouse"*) a partir de las diferentes fuentes de datos para unificar de manera operativa toda la información recogida.

3. Implementación del *Data Warehouse* que permita la navegación y visualización previa de los datos, para discernir qué aspectos puede interesar que sean estudiados.

4. Selección, limpieza y transformación de los datos que se van a analizar.

5. Seleccionar y aplicar el método de minería de datos apropiado, por ejemplo, clasificación, agrupamiento o clustering, regresión, etc.

6. Evaluación, interpretación y representación de los resultados extraídos, para identificar patrones de interés.

7. Difusión y uso del nuevo conocimiento. El conocimiento se obtiene para realizar acciones, ya sea incorporándolo dentro de un sistema o simplemente para reportarlo a usuarios interesados.

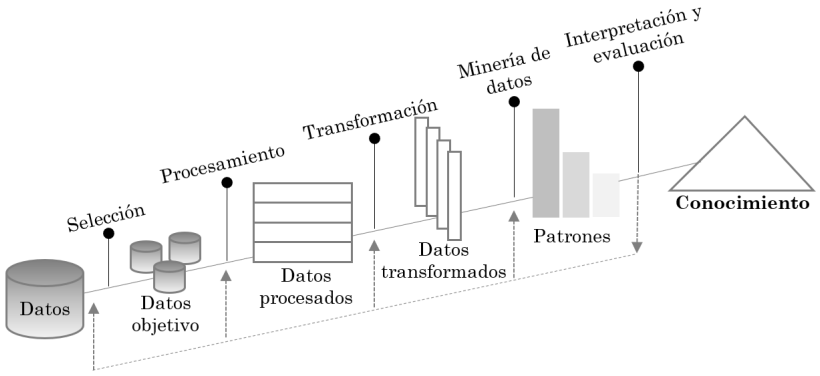


Figura 1.2: Proceso de descubrimiento del conocimiento

Tal como se muestra en la Figura 1.3 las técnicas de ciencias de datos usualmente se dividen en cuatro tipos, los dos más utilizados son el aprendizaje "supervisado" y el "no supervisado". En el primero cada una de las observaciones están etiquetadas con el evento de interés; este tipo de aprendizaje se subdivide en clasificación y regresión. En clasificación las salidas del sistema son finitas y discretas y son interpretadas como la clase a la que pertenece, ejemplo "ausente" o "presente", mientras que en la regresión, las salidas son continuas. En el otro tipo de aprendizaje, el "no supervisado", no hay una variable objetivo y lo que se requiere es buscar patrones, se puede buscar agrupamiento por medio de las técnicas de cluster o reducción de dimensiones como puede ser Componentes Principales entre otras. Los otros dos tipos de técnicas son: el aprendizaje "semi-supervisado", que se utiliza cuando no todos los datos están asignados a una etiqueta, el reto de estas técnicas es poder combinar tanto los datos no etiquetados con los etiquetados para construir un buen modelo y por último el aprendizaje "por refuerzo" que tiene como objetivo clasificar diferentes situaciones para tomar decisiones; el principal campo de aplicación es la robótica y la Inteligencia Artificial, donde el agente se posiciona dentro de un entorno y en cada paso se puede recibir una recompensa dada por una acción positiva. Un ejemplo común son simuladores y videojuegos.

En esta investigación, dado que los datos están ya categorizados, se aplica el tipo de aprendizaje supervisado.

Figura 1.3: Clasificación de *Machine Learning*

Un algoritmo de minería de datos es un conjunto de cálculos y reglas heurísticas que permite crear un modelo, el algoritmo analiza los datos proporcionados en busca de tipos de patrones o tendencias. A continuación se muestran algunas de las técnicas que engloba la ciencia de datos para modelos de clasificación.

7.3.1. Regresión logística

La regresión logística es un caso particular del modelo lineal generalizado donde la función de enlace que se utiliza es la función "logit", de allí es que recibe su nombre. El modelo asume que la respuesta Y_i para cada individuo i tiene distribución Bernoulli, o en forma equivalente, Binomial de parámetros 1 y p_i ($Y_i \sim B(1, p_i)$), con función de probabilidad

$$P(Y = y_i) = p_i^{y_i}.(1 - p_i)^{y_i},\ y_i = 0, 1,\ i = 1, ..., n$$

donde los parámetros $p_i, ..., p_n$ son estimados a partir de los datos. Para un vector de covariables $\mathbf{X_i}(i = 1, ..., n)$ de dimensión r, la regresión logística se basa en modelar p_i de la siguiente forma

$$p_i = P(Y_i|\mathbf{X} = \mathbf{X_i}) = \frac{e^{\beta_0+\beta_1 x_{1i}+...+\beta_r x_{ri}}}{1 + e^{\beta_0+\beta_1 x_{1i}+...+\beta_r x_{ri}}}$$

donde $\beta = (\beta_0, \beta_1, ..., \beta_r)'$ es un vector de (r + 1) parámetros a ser estimados. Entonces si p es la probabilidad de que ocurra el evento en cuestión y $g : [0, 1] \to R\ /\ g(p) = ln\left(\frac{p}{1-p}\right)$ la función logística es la función inversa de g:

$$g^{-1} : R \to [0, 1]\ /\ g^{-1}(z) = \frac{1}{1 - e^{-z}} = \frac{e^z}{1 + e^z} = p$$

Así, si $p = (Y = 1/X_1, X_2, ..., X_1)$ el modelo de regresión logística está dado por

$$logit(p) = ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1\mathbf{X_1} + ... + \beta_r\mathbf{X_r}$$

El modelo de regresión logística tiene la particularidad de poder cuantificar la relación entre cada una de las covariables y la variable dependiente a través de la Razón de Odds.

Una vez obtenido el modelo, como el objetivo es predecir, se predefine un punto de corte para clasificar a un individuo con presencia del evento de interés a partir de la probabilidad estimada del individuo i, $\hat{p_i} = P(Y_i|\mathbf{X}_i)$, donde $\mathbf{X}_i$ es el vector de covariables observadas para el individuo i. En el presente trabajo, se define 0.5 como el valor umbral.

7.3.2. Técnicas basadas en árboles de clasificación

Existen varios algoritmos para la construcción de árboles, pero en la mayoría se sigue una regla general: se particiona el espacio de las covariables utilizando una regla en forma recursiva y de esta manera se genera un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde el nodo raíz hasta alguna de sus hojas. Es decir, los datos se van dividiendo repetidamente en conjuntos más pequeños hasta que el algoritmo determina que los datos colocados dentro de los subconjuntos son homogéneos. En la Figura 1.4 se muestra un ejemplo del proceso de particionamiento [13].

En principio, hay muchos árboles de decisión que pueden construirse a partir de un conjunto dado de atributos. Si bien algunos de los árboles son más precisos que otros, encontrar el árbol óptimo no es factible computacionalmente debido al tamaño exponencial del espacio de búsqueda. Sin embargo, se han desarrollado algoritmos eficientes para inducir un árbol de decisiones razonablemente preciso, aunque no óptimo, en un tiempo razonable.

Para la construcción de los algoritmos se deben considerar al menos los siguientes tres puntos [13]:

1. Selección de la covariable utilizada en cada paso

   En cada paso de la recursión el algoritmo debe decidir cuál es la mejor variable para realizar

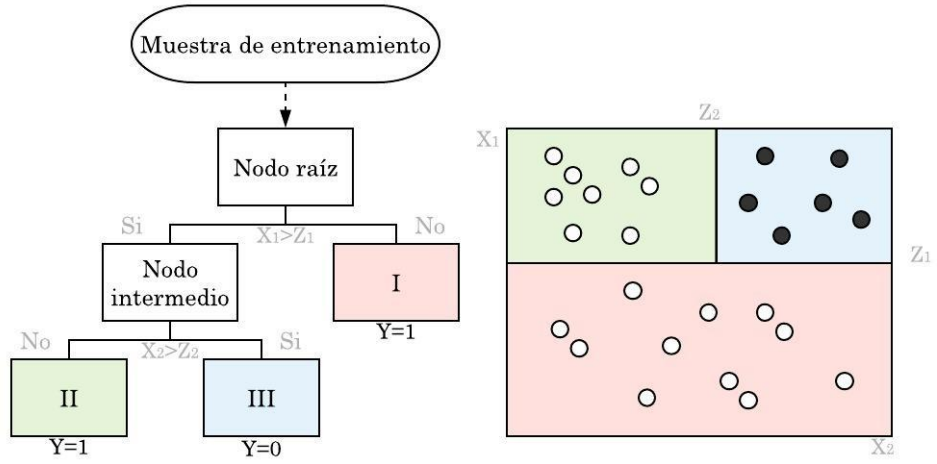


Figura 1.4: Esquema del proceso de particionamiento realizado por el árbol de clasificación

la división de los registros en subconjuntos más pequeños. Para esto es necesario considerar todas las divisiones posibles para cada una de las variables, luego evaluar cada una y decidir cuál es la mejor en algún sentido.

El número de divisiones posibles en un nodo para un predictor continuo u ordinal es el número de valores observados distintos de esa variable menos uno. Para una variable nominal con M categorías distintas, se tienen $2^{M-1}-1$ divisiones distintas posibles. Este valor surge de tomar los $2^M - 2$ subconjuntos posibles del conjunto de niveles distintos de la variable (ignorando las divisiones donde uno de los nodos hijos es vacío) y luego dividiendo esta cantidad por dos, debido a que la mitad de estas divisiones son redundantes.

2. Determinación del punto de división

   Sumando el número de divisiones posibles de los r predictores se obtienen las distintas formas de dividir un nodo padre en los nodos hijos. Para elegir la mejor división es necesario definir una medida de bondad de la división, en general se busca una que resulte en nodos nodos terminales lo más homogéneos posible con respecto a la variable que se desea predecir.

   Una medida cuantitativa que permite medir el grado de homogeneidad del nodo es el concepto de impureza, entre las más conocidas se encuentra [14]:

   Función de entropía, mide el grado de homogeneidad en un conjunto de datos y se define de la siguiente manera:

$$H(\tau) = -\sum_{c=1}^{C} p_{c,\tau}\ log_2\ p_{c,\tau}$$

donde c es la clase, C es el número de clases y $p_{c,\tau}$ es la proporción de casos de la clase c y el nodo τ.

- Si todos los datos pertenecen a una categoría, la entropía es 0.
- Si los datos están mezclados en partes iguales, la entropía alcanza su máximo en 1.
- Mientras más uniforme es p, mayor es su entropía.

Ejemplo:

Clase 1	0
Clase 2	6

Entropía= 0.000

Clase 1	1
Clase 2	5

Entropía= 0.650

Clase 1	2
Clase 2	4

Entropía= 0.918

Clase 1	3
Clase 2	3

Entropía= 1.000

Ganancia de la información, mide la reducción de entropía en el nodo causada por particionar según el atributo A y se define como,

$$Gain(\tau, A) = H(\tau) - \sum_{v \,\epsilon\, valores(A)} \frac{|\tau_v|}{|\tau|} H(\tau_v)$$

donde $H(\tau)$ es la entropía de la clase, el segundo término de la ecuación calcula la entropía en las tablas a medida que se va instanciando la variable A y calcula el promedio ponderado;los ponderadores están relacionados con el número de filas que toman el valor v $|\tau_v|$ divido por el total de filas $|\tau|$.

Índice de Gini, mide el grado de "pureza" con respecto a las clases, mayor Gini implica menor pureza y se define como 1 - Probabilidad de sacar dos registros de la misma clase,

$$Gini(\tau_i) = 1 - \sum_{c=1}^{C} p_{c,\tau_i}^2$$

Ejemplo:

Clase 1	0
Clase 2	6

Gini= 0.000

Clase 1	1
Clase 2	5

Gini= 0.278

Clase 1	2
Clase 2	4

Gini= 0.444

Clase 1	3
Clase 2	3

Gini= 0.500

El índice de Gini tiende a seleccionar divisiones que aíslan una clase mayoritaria en un nodo y el resto en otros nodos y tiende a crear divisiones desbalanceadas. En cambio, la entropía favorece a las divisiones balanceadas. En general, el índice de Gini es la opción por defecto

en muchos programas estadísticos que permiten la construcción de árboles de clasificación y es la función que se utiliza en este trabajo.

Luego de determinar la función de impureza es posible obtener la bondad de la división s en el nodo τ como,

$$\Delta i(s,\tau) = i(\tau) - p_{Li}(\tau_L) - p_{Ri}(\tau_R)$$

donde $i(\tau)$ se refiere a algunas de las funciones de impureza en el nodo τ_i, τ_L y τ_R denotan al nodo izquierdo y al nodo derecho, respectivamente, que nacen del nodo padre τ y con p_L a la estimación de la probabilidad condicional $P\{Y = 1/\tau_L\}$ en el nodo hijo izquierdo. De forma equivalente se define p_R pero para el nodo hijo derecho.

$\Delta i(s,\tau)$ mide el grado de reducción en la impureza al dividir el nodo τ en los nodos τ_L y τ_R. La mejor división para una covariable X_j es aquella que tiene el mayor valor $\Delta i(s,\tau)$ sobre todas las divisiones $s \epsilon S_j$, el conjunto de distintas divisiones posibles de X_j.

Por último, se compara la mejor división realizada con X_j con las mejores divisiones que originan los demás predictores, y se define la mejor división s para ese nodo como la que tiene el mayor valor de $\Delta i(s,\tau)$ entre las r mejores divisiones de cada variable en ese nodo [13].

3. Elección del criterio de parada

   Si el árbol crece hasta que ninguno de los nodos puede ser dividido, se dice que el árbol es saturado; en caso contrario, la recursión puede terminar previamente cuando se cumpla algún criterio de parada prefijado. Decidir sobre este punto no es trivial. De hecho, los árboles grandes pueden sufrir un sobreajuste a las características de la muestra de aprendizaje y luego para datos nuevos, donde algunas características varían, la predicción no resulta eficiente. y árboles pequeños pueden obviar aspectos importantes del problema. Al final del proceso de construcción, los nodos terminales reciben la etiqueta de grupo o clase. Esta etiqueta se establece según algún criterio especificado previamente, como por ejemplo, por voto mayoritario. Puede haber más de un nodo terminal con la misma etiqueta [13].

   Una forma de evitar el sobreajuste son los procedimientos de poda, los cuales básicamente consisten en dejar crecer el árbol hasta saturarse y luego elegir un sub-árbol a partir de algún criterio que se determine. Otra forma, es restringir el crecimiento del árbol con un criterio de parada, por ejemplo declarando que un nodo es terminal si tiene menos observaciones que

un cierto tamaño establecido a priori, esto es τ es terminal si $n(\tau) \leq n_{min}$, donde $n(\tau)$ es el número de observaciones en el nodo τ y n_{min} es algún tamaño mínimo del nodo declarado previamente. Esto constituye un freno en el crecimiento del árbol y es más severo cuánto mayor el valor de n_{min} [13].

A partir de los expuesto recientemente se detallan algunos algoritmos de segmentación:

Árbol de clasificación CHAID

Es el acrónimo de *Chi-Squared Automatic Interaction Detector*, estos árboles fueron desarrollados por Kass en 1980; es una herramienta utilizada para descubrir relaciones entre una variable respuesta y varias variables explicativas, todas categóricas y es capaz de construir árboles con más de dos ramas en cada nodo (Figura 1.5) lo que constituye una ventaja frente al algoritmo CART que se detalla en el siguiente apartado.

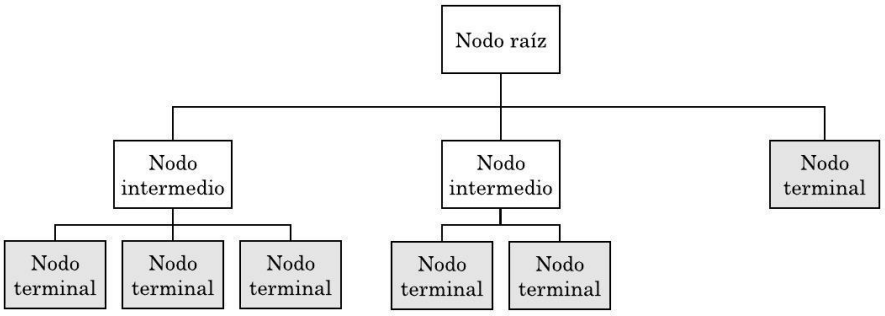


Figura 1.5: Esquema del algoritmo CHAID

Es apropiado para clasificación de variables categóricas y en el caso de que no lo sean, estas se discretizan de la siguiente manera: eligiendo puntos a_d del rango de variación del predictor y luego se trabaja con las D categorías que surgen $A_i = x : a_{d-1} < \ x \leq a_d$ con d = 1,..., D; siendo a_0 el mínimo del predictor y a_D igual al máximo del predictor. Estos algoritmo son recomendable para tamaños muestrales grandes.

El algoritmo comienza dividiendo a la población en dos o más grupos basados en las categorías de la variable explicativa que se selecciona como mejor predictor. Luego divide a cada uno de estos grupos en subgrupos más pequeños basándose siempre en la variable de mayor poder predictivo. El proceso de partición continúa hasta que no se encuentran variables que produzcan particiones significativas y luego se muestran los segmentos resultantes en un gráfico de árbol fácil de comprender [15].

Como criterio de parada considera todos los cortes posibles en todas las variables y selecciona

el corte que da el menor p-valor asociado. Si la variable criterio es categórica la medida es la χ^2 de Pearson y si es continua se utiliza el test de la F.

La búsqueda de la variable y el corte óptimo se lleva a cabo en dos fases: merge (fusión de categorías) y split (selección de la variable de corte) [16].

Las particiones que construye CHAID son:

- Mutuamente excluyentes: las ramas no se superponen o sea un individuo no puede pertenecer a dos ramas.
- Exhaustivos: un individuo ha de pertenecer siempre a una rama y no pueden existir individuos aislados.

El algoritmo CHAID realiza particiones sucesivas del conjunto de datos siguiendo las reglas sobre los predictores que mejor explican a la variable respuesta; para esto se utilizan los p-value asociados a las pruebas entre los predictores y la respuesta (menor p-value, mayor poder predictivo). Para cada predictor con más de dos categorías se evalúa unir o agrupar las categorías en las cuales la distribución de la respuesta sea similar. Luego de cada unión de tres o más categorías, se puede (opcionalmente) evaluar dividir o re-agrupar de alguna manera esa unión con el objeto de lograr mayor poder predictivo. El algoritmo también utiliza p-value asociados a las pruebas para realizar las uniones y las divisiones de las categorías, para lo cual es necesario definir dos p-value críticos: α-marge y α-split.

A continuación se resumen los pasos del algoritmo CHAID [15]:

1. Cuando el predictor es binario se continúa con el paso 7 porque no es necesario unir categorías. Si el predictor tiene tres o más categorías, se continúa con el paso 2.

2. Para cada agrupamiento posible de dos categorías se realiza la prueba de asociación entre el predictor (sólo se consideran las observaciones del par de categorías que se evalúan) y la respuesta. Se busca el par de categorías del predictor en las cuales la distribución de la respuesta sea más similar en función de los p-values (a mayor p-value mayor similaridad).

   - Para variables nominales, todos los pares son elegibles.

   - Para variables ordinales, solo se buscan agrupar categorías adyacentes.

3. El valor del p-value para ese par de categorías es ajustado por Bonferroni[1] para evitar falsos

[1]Técnica estadística que ajusta el nivel de significación en relación al número de pruebas estadísticas realizadas simultáneamente sobre un conjunto de datos. El nivel de significación para cada prueba se calcula dividiendo el error global de tipo I entre el número de pruebas a realizar. El ajuste de Bonferroni se considera conservador.

positivos por los múltiples test. Si el valor del p-value ajustado del par de categorías con el p-value más alto supera el límite α-merge, se combina el par de categorías en una sola. De lo contrario, se continúa con el paso 6.

4. Cuando la categoría compuesta recién formada contiene tres o más categorías originales y se elige evaluar la separación de categorías combinadas, se busca la mejor división binaria dentro de la categoría compuesta (aquella para la cual el p-value de la prueba de asociación es más pequeño). Si ese p-value es menor o igual al límite α-split, se forman dos nuevas categorías a partir de la categoría compuesta. Para el cálculo del p-value (a diferencia de cuando se evalúa una unión de categorías) se utilizan las observaciones de todas las categorías del predictor.

5. Se continúa combinando categorías para el mismo predictor desde el paso 1.

6. Toda categoría cuya frecuencia sea menor a un límite especificado, se une con la categoría más similar (lo que le da el mayor p-value en comparación con la categoría poco frecuente).

7. Cuando las categorías ya se han combinado para todos los predictores, se evalúa la asociación de cada predictor con la respuesta, basado en el p-value de la prueba de asociación.

8. Se elige el predictor con la asociación más fuerte (menor p-value) y se compara con el límite de división, α-split. Si el p-value$\leq\alpha$-split, ese predictor se selecciona como la variable de división o partición para el nodo actual. Cada una de las categorías compuestas de la variable de división define un nodo hijo de la división.

9. Luego, para cada uno de los nodos hijos se continúa el proceso de unión y división (pasos 1 al 8). El proceso continúa de forma recursiva hasta que se cumpla una o más reglas de parada y no sea necesario realizar más divisiones.

El criterio de parada depende del valor prefijado de α-split, del número de niveles de la estructura del árbol y del umbral mínimo para el tamaño de los nodos descendientes.

Algoritmo CART

Es el acrónimo de *Classification And Regression Trees*, fue desarrollado por matemáticos de la Universidad de Berkeley y Stanford (Breiman, Friedman, Olshen y Stone) a mediados de los 80. Tienen la ventaja de ser útiles para variable categóricas y continuas y la desventaja es que las reglas de decisión son binarias, es decir, determinan en cada momento sólo dos alternativas posibles (Figura 1.6), esto da lugar a estructuras de árbol de mayor profundidad.

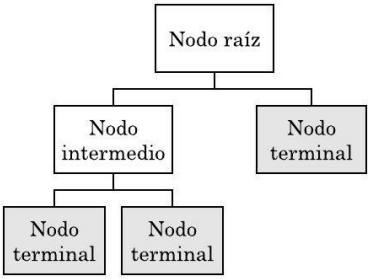


Figura 1.6: Esquema del árbol CART

Se basa en la idea de impureza. CART realiza la partición que conduce al mayor decrecimiento de la impureza. Así se consiguen descendientes homogéneos en la variable respuesta Y [16].

El algoritmo de los árboles CART realiza una partición recursiva del espacio de las variables explicativas a partir de un conjunto de reglas de decisión para lo cual es necesario definir previamente una medida de impureza (o variabilidad), un criterio de Bondad de la partición y una regla de parada.

Los pasos del algoritmo se detallan a continuación [15]:

1. Se comienza con un primer factor predictivo, seleccionando un valor para dividir los datos del nodo raíz en dos.

   - Para un predictor nominal la regla binaria es de la forma: $\{x : x = A\}$ y $\{x : x \neq A\}$ donde A es un subconjunto de valores posibles de la variable.

   - Cuando la variable explicativa es ordinal las particiones son de la forma: $\{x : x < A\}$ y $\{x : x \geq A\}$ donde A es un valor que pertenece al rango de variación de las variables.

   - Si el predictor es cuantitativo, se discretiza la variable eligiendo D puntos a_d del rango de variación y luego se comparan los grupos $\{x : x < a_d\}$ y $\{x : x \geq a_d\}$ con d=1,. . . , D.

2. Se calculan las impurezas de las particiones y se evalúa si la reducción de la impureza del nodo padre es mayor al criterio de la Bondad de Partición.

3. Entre las particiones que cumplen el criterio, se selecciona la que muestra una mayor reducción de la impureza y se la denomina la “mejor” división para el predictor.

4. Se repiten los pasos 1 a 3 para el resto de los predictores.

5. Se ordenan las “mejores” particiones de cada predictor de acuerdo con la reducción de la

impureza de la partición. Se selecciona el predictor (y su partición) que logra la mayor reducción.

6. Se asignan las observaciones a los dos nodos hijos.

7. Se evalúa si los nodos hijos son terminales utilizando la regla de parada.

8. Se repiten los pasos 1 a 7 para cada uno de los nodos hijos no terminales.

El objetivo de la partición recursiva es obtener nodos terminales homogéneos tanto como sea posible en el sentido de que contengan observaciones de solo uno de los grupos o categorías de la variable respuesta. Cuando en un nodo la variable respuesta no varía, el nodo se considera puro.

Todos los métodos que desarrollan árboles de decisión se basan en las técnicas anteriormente descriptas para el algoritmo CART y sólo difieren de éste en algunos detalles. A continuación se nombran y describen de forma muy general los otros métodos más extendidos y sus principales diferencias con lo anteriormente expuesto (estos métodos no van a ser aplicados en este trabajo):

Induction Decision Trees (ID3)

Es un algoritmo enfocado al uso de variables categóricas, fue desarrollado en los años 80 por Quinlan. Si existen variables continuas, se agrupan primero sus valores en intervalos para luego ser tratadas como variables categóricas (sin relación de orden). Otra de sus características distintivas es que sólo considera como variables posibles para la división del nodo aquellas que no han sido utilizadas anteriormente y por lo tanto, los árboles generados mediante este método tienen como máximo tanta profundidad como tenga la dimensión de los datos [17].

C4.5

Es el sucesor de ID3 y el otro método más utilizado para la generación de árboles de decisión junto con CART; este algoritmo fue propuesto por Quinlan en 1993.

Algunas de las diferencias respecto de CART son: permite nodos con más de dos hijos, el valor usado para el corte en un nodo no corresponde al punto medio entre dos valores de una variable sino que usa el valor de la menor y se basa en la entropía para calcular la impureza. El procedimiento de poda es distinto del usado por CART ya que utiliza heurísticas basadas en test de significancia estadística. La diferencia entre los resultados finales obtenidos con ambas podas es que mientras que la poda CART tiende a generar árboles más pequeños que el óptimo, la poda C4.5 tiende a podar menos de lo necesario [17].

A continuación se detallan algunas ventajas y desventajas de los árboles [18]:

- Fácil de entender e interpretar. Puede ser interpretados por no expertos en el tema, especialmente si son árboles pequeños.
- Requiere poca preparación de los datos. Otras técnicas a menudo requieren la normalización de datos, utilización de variables dummy y valores nulos deben ser eliminados.
- Permiten tratar a los datos perdidos como categorías independientes dentro de cada variable.
- Capaz de manejar datos numéricos y categorizados. Otras técnicas son generalmente especializadas en el análisis de conjuntos de datos que tienen sólo un tipo de variable.
- Utiliza un modelo de caja blanca. Si una situación dada es observable en un modelo entonces la condición se explica fácilmente por la lógica booleana.
- Es posible validar un modelo utilizando pruebas estadísticas. Eso hace que sea posible tener en cuenta la fiabilidad del modelo.
- Funciona bien con grandes conjuntos de datos. Grandes cantidades de datos pueden ser analizados utilizando recursos informáticos estándar en un plazo razonable.
- Son eficaces para buscar perfiles o patrones en las bases de datos pero son ineficaces para descubrir relaciones lineales en los datos.

Sin embargo, los árboles de decisión tienden a tener una gran variación cuando utilizan diferentes conjuntos de pruebas y entrenamiento de los mismos datos, ya que tienden a adaptarse en exceso a los datos de entrenamiento. Desafortunadamente, esto limita el uso de árboles de decisión en modelos predictivos. Sin embargo, al usar métodos ensamblados, se puede crear modelos que utilicen árboles de decisión subyacentes como base para producir resultados poderosos. A continuación se presenta el algoritmo de *Random Forest*.

7.3.3. *Random Forest*

Como su nombre lo indica son bosques aleatorios formados por un conjunto de árboles construidos de tal manera que trata de reducir la correlación entre ellos gracias a dos fuentes de aleatoriedad. Una vez construido todos los árboles se genera una predicción promediando las predicciones individuales.

Es conocido por su buen desempeño en problemas de clasificación. Además, es un modelo relativamente fácil de construir y no requiere mucho ajuste de hiperparámetros. Esto se debe a que

los hiperparámetros principales son la cantidad de árboles en el bosque y la cantidad de entidades a dividir en cada nodo de hoja.

El algoritmo de formación del Forest es el siguiente [19]:

1. Para cada uno de los árboles, dada la muestra inicial con N observaciones diferentes, se eligen de forma aleatoria n datos de la muestra con reemplazo. Esto se conoce como boostraping. El hecho de que cada árbol se forme con una muestra ligeramente distinta constituye la primer fuente de aleatoriedad.

2. En cada nodo de cada árbol, en lugar de buscar la característica más importante se busca la mejor entre un subconjunto aleatorio de las variables explicativas, es decir, se eligen de forma aleatorio m $<$ r variables candidatas para la partición (siendo r el número de variables explicativas). El número de variables m elegido será constante durante todo el proceso de formación del árbol. Esta reducción en el número de variables candidatas constituye la segunda fuente de aleatoriedad del proceso. Esto da como resultado una amplia diversidad que generalmente resulta en un mejor modelo.

3. Se deja crecer cada árbol sin podar hasta la máxima extensión posible.

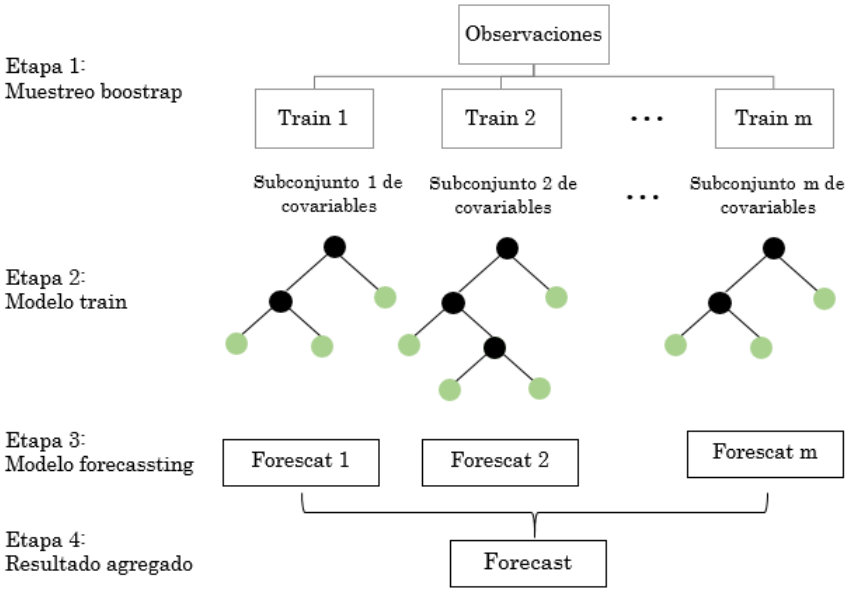


Figura 1.7: Funcionamiento del *Random Forest*

Por lo tanto, este algoritmo para dividir un nodo solo tiene en cuenta un subconjunto aleatorio de las características. Es decir, los *Random Forest* tienen dos parámetros fundamentales de diseño:

Ntree: número de árboles individuales que forman el Forest.

Mtry: número de variables elegidas en cada una de las particiones.

Variaciones en ambos parámetros conducen a resultados ligeramente diferentes. Al reducir el valor de *mtry*, se reduce la correlación entre los árboles debido a que en cada nodo se tienen menos posibilidades de variables entre las que elegir con el objetivo de reducir la impureza. Es más improbable que salgan las mismas variables en las sucesivas elecciones aleatorias. Sin embargo, reducir el valor de *mtry* también puede reducir la precisión de cada árbol individual dado que en cada nodo se tiene menos opciones entre las que elegir las variables que mayor reducción de impureza genera en el árbol. El número de árboles *ntree* también tiene efecto en la precisión de la predicción. De forma lógica, cuanto más árboles individuales diferentes se construyan con las distinta muestras de los datos iniciales mejor será el carácter de análisis del Forest y mejor serán sus predicciones, puesto que se está promediando con más datos. Sin embargo, existe un cierto valor de *ntree* en el cual se estabiliza el error de predicción, contribuyendo el incremento en el número de árboles de forma muy poco significativa a la reducción del error. Este valor es el número óptimo de árboles a construir, puesto que la construcción de más tiene un alto coste en tiempo que no se traduce en una mejora en la predicción [20].

Siendo r el número de variables explicativas, las recomendaciones de Breiman y Adele para el valor de *mtry* son:

Para clasificación: $\sqrt{r}$

Para regresión: $\frac{r}{3}$

El Out of the Bag Mean Squared Error (MSE-OOB) es una medición de error típica de los *Random Forests* y de otros algoritmos que emplean la técnica del boostrapping.

En la elección aleatoria con reemplazo de los n datos de las N observaciones que forman la muestra inicial se demostró que, en general, queda fuera de esta submuestra aproximadamente el 36 % del total de los datos, las observaciones no usadas para ajustar cada árbol se conocen como out-of-bag (OOB) [19]. El MSE-OOB estima el error de predicción teniendo en cuenta estas observaciones que se han quedado fuera, de la siguiente manera:

$$MSE - OOB = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{iOOB})^2$$

Siendo y_{iOOB} la predicción para la observación i obtenida promediando las predicciones individuales de los árboles para los que esa observación se ha quedado fuera de la bolsa e y_i el valor real de la

variable respuesta. Según lo comentado en el apartado anterior, el MSE-OOB tiene una dependencia importante con los parámetros *ntree* y *mtry*, la influencia del valor de *mtry* en el error depende del número de variables de entrada del modelo. Sin embargo, el MSE-OOB se reduce de forma asintótica con el número de árboles.

Otra ventaja de este algoritmo es que es muy fácil medir la importancia relativa de cada característica en la predicción; mide la importancia de las características al observar cuánto nodos de los árboles, que usan esa característica, reducen la impureza en todos los árboles. Calcula esta puntuación automáticamente para cada función después del entrenamiento y escala los resultados, de modo que la suma de toda la importancia sea igual a 1. Al observar la importancia de la característica, puede decidir qué características desea eliminar, ya que no contribuyen lo suficiente o nada al proceso de predicción. Esto es importante, ya que una regla general en el aprendizaje automático es que cuantas más características tenga, más probable es que su modelo sufra un exceso de adaptación y viceversa.

Una diferencia con los árboles de decisión es que estos pueden sufrir de sobreajuste; *Random Forest* evita el exceso de adaptación la mayor parte del tiempo, creando subconjuntos aleatorios de las características y construyendo árboles más pequeños utilizando estos subconjuntos.

La principal limitación de *Random Forest* es que una gran cantidad de árboles puede hacer que el algoritmo sea lento e inefectivo para las predicciones en tiempo real. En general, estos algoritmos son rápidos para entrenar, pero bastante lentos para crear predicciones una vez que están entrenados. Una predicción más precisa requiere más árboles, lo que resulta en un modelo más lento. Otra desventaja para destacar es la pérdida de interpretación.

7.3.4. Extreme Gradient Boosting (XGBOOST)

El algoritmo XGBoost (por extreme gradient boosting) es una implementación del método gradient boosting decision tree. Está basado en aprovechar los recursos de cómputo con los que se cuenta, permitiendo paralelizar y distribuir la creación de árboles de decisión optimizando el uso de memoria.

La principal diferencia de este método con el de *Random Forests* es que en este último los árboles que se ensamblan son independientes mientras que en XGBoost dependen de los errores de los modelos anteriores, es decir, uno usa bagging y el otro boosting para el ensamble. En la figura 1.8 se puede observar gráficamente el funcionamiento de cada uno, mientras que en bagging el entrenamiento se hace de manera paralela (cada modelo es independiente) en boosting la construcción

de modelos es secuencial.

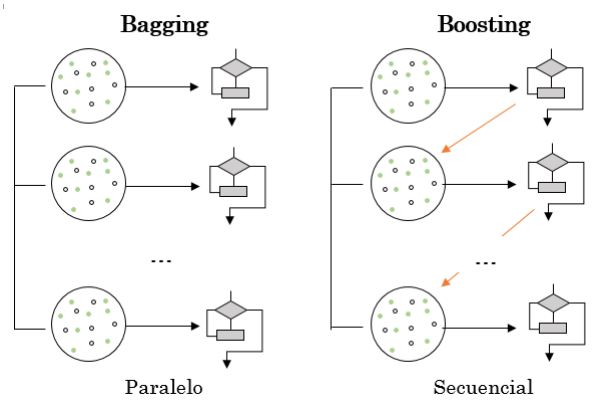


Figura 1.8: Comparación entre el algoritmo de *Random Forest* y *XGBoost*

Tanto este algoritmo como el de *Random Forests* devuelven la importancia de cada atributo en el resultado final. En el caso de XGBoost, la calcula sumando cuantas veces cada atributo parte un nodo en cada árbol del modelo. La diferencia con el de *Random Forests* está en los atributos que se encuentran correlacionados: suponiendo que se tienen dos atributos A y B perfectamente correlacionados, como en *Random Forests* los árboles se crean de manera independiente, un porcentaje de estos va a contener a A mientras que el porcentaje restante contendrá a B. De este modo, la información que contienen estos atributos (que es la misma por estar perfectamente correlacionados) será divida en estos dos. Esto en XGBoost no sucede puesto que cuando el algoritmo aprende la influencia de una variable en el resultado, no se vuelve a enfocar en ella. Por esta razón, la importancia aquí estará toda en A o en B pero no divida entre ambas como en el caso de *Random Forests* [21, 22].

7.3.5. Naïve Bayes

Naïve Bayes es una técnica de clasificación basada en el Teroma de Bayes; esta teoría supone un tamaño de muestra asintóticamente infinito e independencia estadística entre las variables explicativas. Con estas condiciones, se puede calcular las distribuciones de probabilidad de cada clase para establecer la relación entre los atributos y la clase. Entonces,

$$P(C/A) = \frac{P(C)P(A/C)}{P(A)}$$

donde C es la clase y A los atributos, estimar la $P(A_1, ..., A_n)$ es complejo, el teorema de Bayes supone que los atributos son independientes dada la clase; por lo tanto, el clasificador bayesiano establece que la probabilidad de la clase dados los valores de los atributos, se puede calcular de la

siguiente manera:

$$P(C = c/A_1, ..., A_n) = \frac{P(C = c) \prod_{i=1}^{n} P(A_i/C = c)}{P(A_1, ..., A_n)}$$

Observando el numerador, aplicando primero la definición Probabilidad Compuesta (pc)[2] y luego la regla de la cadena (rc)[3], se obtiene el siguiente resultado:

$$\begin{aligned} P(C)P(A_1, ..., A_n/C) &\overset{pc}{=} P(C, A_1, ..., A_n) \\ &\overset{rc}{=} P(C).P(A_1/C).P(A_2/C, A_1)...P(A_n/C, A_1, ..., A_{n-1}) \\ &= P(C) \prod_{1 \leq i \leq n} P(A_i/C) \end{aligned}$$

Este punto es donde el método obtiene el adjetivo de Naïve: asume que las variables explicativas son probabilísticamente independientes, con los cual utilizando la definición de probabilidad condicional [4] bajo este supuesto queda:

$$P(C/A_1, ..., A_n) = \frac{P(C) \prod_{1 \leq i \leq n} P(A_i/C)}{P(A_1, ..., A_n)}$$

De esta manera, partiendo del cálculo de una probabilidad a posteriori para determinar el grado de pertenencia a una clase, se obtiene una expresión equivalente que se basa exclusivamente en la evidencia y los datos obtenidos en etapa de entrenamiento, y es con esta expresión que Naïve Bayes realiza la clasificación de acuerdo a la clase para la que obtenga mayor probabilidad. Formalmente, siendo $A_1, ..., A_n$ las variables de un individuo i, la clasificación es realizada de la siguiente manera [23]:

$$p_i = \underset{c \epsilon C}{argmax} \frac{P(C) \prod_{1 \leq i \leq n} P(A_i/C)}{P(A_1, ..., A_n)}$$

Se utiliza argmax para obtener el máximo de la función en el argumento, en la Figura 1.9 se muestra un ejemplo,

[2] $P(A_1, ..., A_n) = P(A_1) * P(A_2, ..., A_n|A_1)$

[3] $P(A_1, ..., A_n) = P(A_1) * P(A_2|A_1) * P(A_3|A_1, A_2) * ... * P(A_n|A_1, ..., A_{n-1})$

[4] P(A/B)=P(A,B)/P(B), pero si A y B son independientes: P(A,B)=P(A)*P(B), por lo tanto P(A/B)=P(A)

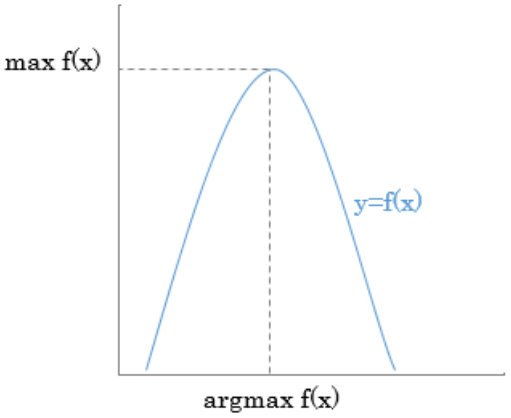


Figura 1.9: Ejemplo de función

7.3.6. Suport Vector Machine

Las Máquinas de Soporte Vectorial (del inglés Support Vector Machine) fueron desarrolladas en 1995 por Vladimir Vapnik. Están basadas en la teoría de aprendizaje estadístico que permiten resolver problemas de clasificación y regresión de manera eficiente. El éxito de las máquinas de soporte vectorial radica en tres ventajas fundamentales:

- Poseen una sólida fundamentación matemática.
- Se basan en el concepto de minimización del riesgo estructural, esto es, minimizar la probabilidad de una clasificación errónea sobre nuevos ejemplos, particularmente importante cuando se dispone de pocos datos de entrenamiento.
- Se disponen de potentes herramientas y algoritmos para hallar la solución de manera rápida y eficiente.

Las máquinas de soporte vectorial a diferencia de las redes neuronales, abstraen el problema desde un espacio de atributos a un espacio de patrones de características con mayor dimensión, a fin de que puedan ser separadas por un hiperplano. Así, mediante una función no lineal de mapeo apropiada, que aumente la dimensión de forma adecuada, es posible separar muestras que pertenezcan a dos categorías diferentes mediante un hiperplano.

- Clasificación linealmente separables

La idea central detrás de las SVMs es definir un margen entre dos clases por la separación máxima entre ellas. Los datos puede ser linealmente separables, y en ellos, minimizar la función de

costos resulta muy sencillo. La hipótesis de partida es que las clases son linealmente separables y por ello existen infinitos hiperplanos que separan las muestras de una clase, de la otra. Por ejemplo con dos clases se pueden considerar los siguientes hiperplanos:

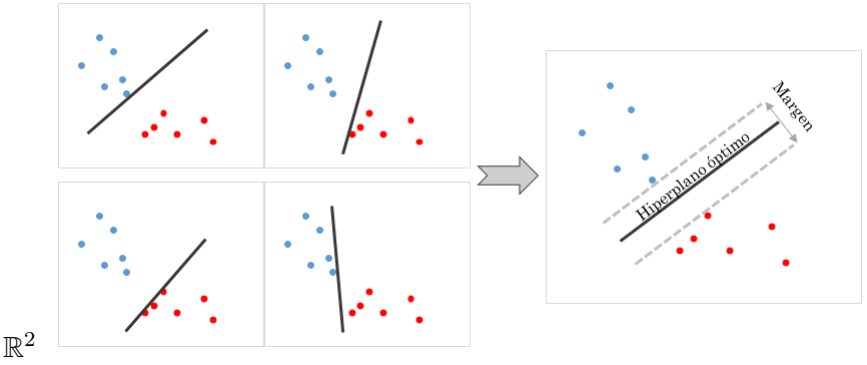


Figura 1.10: Posibles hiperplanos separadores

Los puntos del espacio que caen dentro de cada uno de estos hiperplanos son los que satisfacen la siguiente expresión.

$$h(x) = w^t x + b = 0$$

Donde w, Y, x ϵ $\mathbb{R}^d$, siendo d la dimensión del espacio de entrada.

Se busca el hiperplano que maximice el margen m entre las clases del espacio. Maximizar dicho margen es un problema de programación cuadrática, pudiendo ser resuelto introduciendo multiplicadores de Lagrange [24].

- Clasificación linealmente no separable

Cuando los datos no son linealmente separables, existe la posibilidad de transformar los datos a un espacio de mayor dimensión ν utilizando una función $x \rightarrow \phi(x) \epsilon\ \nu$, donde se encontrará un hiperplano que los pueda separar. La frontera de decisión resultante en el espacio de entrada ya no será lineal y vendrá dada por otro tipo de función que pueda ser polinómica de grado superior a 1, gaussiana, sigmoide, entre otras, dichas funciones se conocen como funciones núcleo o "Kernel".

Las muestras una vez proyectadas, pueden usarse como un nuevo conjunto de entrenamiento, de esta forma se buscará una frontera lineal en el espacio ν. Como se nota ya los datos en la formulación de la máquina de soporte vectorial sólo aparecen como producto entre las muestras $\phi(x_i).\phi(x_j)$, en este caso solo se necesita la función Kernel, tal que $K(x_i, x_j) = \phi(x_i).\phi(x_j)$ Teniendo esta función se puede aplicar el algoritmo de entrenamiento de la Máquina de soporte vectorial sin

conocer explícitamente cual es la transformación $\phi(x)$ o incluso el espacio ν [25]. La única condición necesaria es que el Kernel usado esté correctamente definido. Las funciones más utilizadas son:

- La Función Polinómica tienes la siguiente forma:

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

- La Función Gaussiana, tiene la siguiente forma:

$$f(x) = a.exp\left(\frac{-(x-b)^2}{2c^2}\right)$$

- La Función Sigmoide tiene la siguiente forma:

$$k(x, x') = tanh(s(x^T.x') + r) \quad s, r \epsilon R$$

La selección del mejor Kernel para una aplicación es todavía un tema de investigación. El procedimiento más común es el de seleccionar los parámetros de Kernel (el grado del polinomio d para las funciones polinomiales o el ancho del kernel para la función Gaussiana) calibrando estos parámetros en conjunto con el proceso de selección del modelo (parámetro C que controla la generalización del modelo) mediante una búsqueda de grilla [26].

7.3.7. Redes Neuronales

Las redes neuronales artificiales, inspiradas en el funcionamiento del cerebro humano, tienen por objetivo aprender de la experiencia acumulada en un conjunto de ejemplos, tratando de emular la actividad del cerebro para aprender. En 1943, Warren McCulloch y Walter Harry Pitts desarrollaron los primeros modelos de redes neuronales, en el año 1958 Rosenblatt desarrolló el perceptrón y luego en los años 80, la Redes Neuronales volvieron a resurgir con BackPropagation.

Tal como se muestra en la Figura 1.11, la neurona biológica consta de un cuerpo dendritas y un axón. El cerebro humano está compuesto por más de cien mil millones de neuronas interconectadas entre sí formando circuitos o redes que desarrollan funciones específicas. Una neurona biológica típica recoge señales procedentes de otras neuronas a través de las dendritas. La neurona emite impulsos de actividad eléctrica a lo largo de una fibra larga y delgada denominada axón, que se esconde en millares de ramificaciones axonales. Las extremidades de estas ramificaciones llegan hasta las dendritas de otras neuronas y establecen unas conexiones llamadas sinapsis, en las cuales se produce una transformación del impulso eléctrico en un mensaje neuroquímico, mediante la liberación de unas sustancias llamadas neurotransmisores. Los neurotransmisores liberados en las

sinapsis al final de las ramificaciones axonales pueden tener un efecto excitatorio o inhibitorio sobre la neurona receptora.

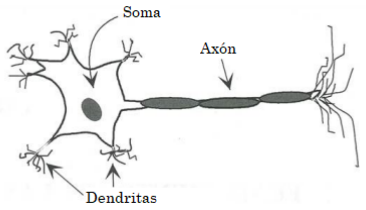


Figura 1.11: Estructura de una neurona biológica típica

En la Figura 1.12 se muestra el esquema de una neurona artificial; las entradas pueden ser binarias o continuas y se suelen representar con un vector de entrada: $X = (x_1, ..., x_n)$ (equivalen a las dendritas de donde reciben la estimulación), que se agregan de forma ponderada $W = (w_1, ..., w_n)$ (fuerza sináptica) que pasan la señal a otra neurona. La excitación S se calcula a partir de los valores de entrada y las ponderaciones. La forma habitual es la suma ponderada de las entradas:

$$S = \sum_{i=1}^{n} x_i w_i$$

Cuando la excitación alcanza un cierto umbral θ_j, permite la activación de la neurona que emite una señal (salida) que a su vez puede estimular a otras neuronas. De forma análoga se puede considerar que una neurona artificial es un procesador elemental que recibe una serie de entradas con ponderaciones diferentes que al agregarse son evaluadas en una función de activación $\sigma(x)$, generalmente no lineal, y esto produce una salida y_j. En términos matemáticos y en base a los elementos recientemente descriptos, una neurona artificial j se puede definir como [27]:

$$y_j = \sigma\left(\sum_{i=1}^{N} w_{ji} x_i \pm \theta_j\right)$$

Existen diferentes tipos de funciones de activación, las más usadas son:

- Función umbral

$$F(x) = 1 \quad si\ x \geq U$$
$$F(x) = 0 \quad si\ x < U$$

- Función umbral lineal (a>0)

$$\begin{array}{ll} -1 & si\ x \leq -1/a \\ ax & si\ -1/a < x < 1/a \\ 1 & si\ x \geq 1/a \end{array}$$

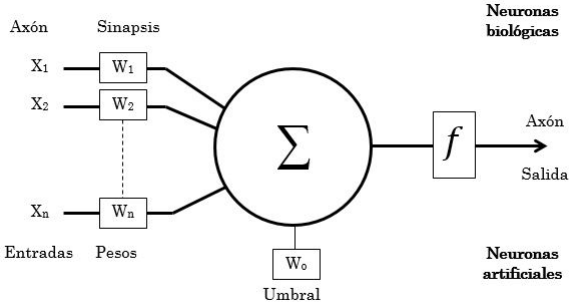


Figura 1.12: Esquema de una Neurona Artificial

- Función sigmoide

$$F(x) = \frac{1}{(1 + e^{-gx})}$$

- Función tangente hiperbólica

$$F(x) = \frac{(e^{gx} - e^{-gx})}{(e^{gx} + e^{-gx})}$$

  En estas últimas dos funciones, g varía con la pendiente de F.

Hay que tener en cuenta que los valores de los pesos son calculados durante el proceso de entrenamiento. Generalmente, estos valores inicialmente tienen valores aleatorios y se van modificando de acuerdo a una ley de aprendizaje.

Existen diversas arquitecturas de redes neuronales que se definen esencialmente a partir de su diseño estructural, su mecanismo de aprendizaje y la recuperación de la información o generalización.

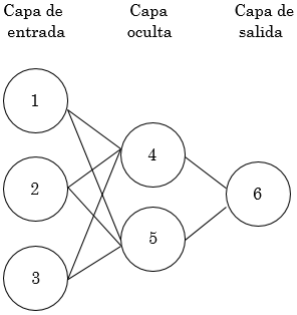
Algunos modelos de Redes Neuronales

- Perceptrón lineal, es el tipo de red más simple. Tiene una capa de entrada y una de salida, sin capas ocultas. Puede servir para resolver problemas lineales.

- Adaline y Madaline, son muy similares a un perceptrón lineal con la particularidad de que se agrega un factor de tendencia y que se añade una condición bipolar, tal que si la salida es positiva la salida del Adaline es +1 y si es negativa la salida es -1; esto genera una condición de salida binaria del Adaline.

- Perceptrón múltiple, con al menos una capa oculta y tiene la capacidad de resolver problemas lineales y no lineales. La red se organiza en capas de neuronas donde las conexiones son hacia delante, es decir, las salidas de las neuronas de la capa anterior corresponden con la entrada

de las neuronas de la capa siguiente, por este motivo también reciben el nombre de redes feedforward; tiene capa de entrada, capada de salida y una o más capa ocultas, en la Figura 1.13 se muestra un ejemplo de arquitectura de perceptrón múltiple.



Matriz de pesos sinápticos

	1	2	3	4	5	6
1	0			x	x	
2		0		x	x	
3			0	x	x	
4				0		x
5					0	x
6						0

Figura 1.13: Conexiones hacia adelante.

Entrenamiento de la red

La Matriz de Pesos Sinápticos se inicializa y luego se va actualizando según la salida correcta o errónea de cada ejemplo para entrenar la red. Este proceso de entrenamiento se denomina Back-propagation que consiste en ajustar los parámetros de la red, es decir, los pesos de las conexiones.

Como se trata de aprendizaje supervisado, el objetivo consiste en minimizar una función de error, E, que evalúa la diferencia entre la salida esperada y la obtenida por la red. Dicha función de error se suele definir de la siguiente forma:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n)$$

donde, N es el número de ejemplos de entrenamiento, e(n) es el error de la red para el ejemplo n y se define de la siguiente manera:

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_c}(s_i(n) - y_i(n))^2$$

s_i: es el valor obtenido de la salida

y_i: es el valor observado

Se define el algoritmo de la siguiente manera:

Se calcula el valor de la primer entrada.

Si valor obtenido es igual al valor esperado, se sigue con el siguiente elemento.

Si valor obtenido es diferente al valor esperado, se actualiza los pesos de la red neuronal.

Al calcular todos los elementos se vuelven al primer elemento. El cálculo de todos los elementos se llama Época o Iteración).

Finalización o parada según algún criterio definido como puede ser: aprendizaje de todos los casos, cota de error aceptada o cantidad de iteraciones.

La fórmula de aprendizaje para un peso w_t se define como:

$$w_{t+1} = w_t + \Delta_w$$

donde Δ_w es la derivada del error para obtener el gradiente descendiente y está asociado al valor esperado, al valor obtenido y a un coeficiente de aprendizaje.

Como se trata de un problema no lineal, es necesario utilizar una técnica de optimización no lineal sin restricciones para resolverlo, concretamente en el campo de las redes de neuronas artificiales se suele utilizar el método del descenso del gradiente. Dicha técnica consiste en variar los parámetros en función del sentido negativo del gradiente de la función E, es decir, la dirección por donde decrece más rápidamente dicha función. Por tanto, el funcionamiento del descenso del gradiente consiste minimizar el error para cada ejemplo entrenamiento, en lugar de minimizar el error global [28].

Un caso particular de las redes neuronales es lo que se conoce como *Deep Learning* que se caracteriza por tener múltiples capas de neuronas conectadas entre sí, lo cual brinda un mejor desempeño que las redes de una sola capa. Aunque en realidad no es el número de capas lo que define el aprendizaje profundo; el concepto subyacente en el aprendizaje profundo es el procesamiento de los datos de forma jerárquica, es decir, el uso de redes neuronales para obtener representaciones cada vez más significativas de los datos mediante el aprendizaje por capas [29].

Cada capa extrae características de un nivel cada vez más alto hasta llegar a su respuesta final. Al ir profundizando en la red, estas funciones más simples se van combinado para buscar relaciones más complejas como puedan ser partes concretas de la cara (ojos, boca, nariz). En un siguiente paso se identificaría la cara completa, y por último se identificaría a la persona a la que corresponde esa cara.

Por lo tanto, aunque existen innumerables formas de arquitectura de red, se pueden diferenciar por el número de capas y la cantidad de nodos dentro de cada capa de la red; actualmente no existe una manera de determinar estos valores de forma óptima.

7.4. Comparación de modelos

Frecuentemente los expertos en machine learning recomiendan en la construcción de los modelos, separar el conjunto de datos en subconjunto para entrenamiento y prueba para encontrar el modelo con menor error. Dentro de las técnicas encontradas para entrenamiento y validación, se propone en la literatura la utilización de *Cross Validation*, con el fin de garantizar la independencia de los resultados mediante la partición del conjunto inicial en datos de entrenamiento y datos de prueba (training set y test set). La validación cruzada se divide en tres tipos; el primero, validación cruzada *k Fold*, consiste en dividir el conjunto de entrenamiento en k partes. Una de las partes se utiliza como dato de prueba y las k-1 restantes como datos de entrenamiento, este proceso es repetido k iteraciones y al final se realiza la media aritmética de los resultados de cada iteración y se obtienen un solo resultado. El segundo tipo de validación es el *aleatorio* y consiste en dividir aleatoriamente el conjunto de datos de prueba. El tercer tipo de validación cruzada es *Leave one out*, consiste en separar una muestra del conjunto de datos para la validación y el resto se utiliza para el entrenamiento, este proceso se repite según el número de muestras que se tenga [30].

Dentro del proceso de elegir el mejor modelo se deben tener en cuenta la complejidad y la medida de error de los mismos, contemplando el sesgo y la varianza. Los errores que se determinan para evaluar el aprendizaje, son el "error de entrenamiento" y "error de validación" y se calculan sobre el conjunto destinado para tal fin. En la literatura se demuestra que cuando se tiene un error de entrenamiento bajo y un error de validación alto hubo un error por varianza, es decir se tiene un *overfitting*, y se entiende que el algoritmo sobre-ajustó los datos de entrenamiento y aprendió del ruido contenido en la muestra y por ello probablemente falle la generalización del modelo. Cuando se tienen un error de entrenamiento alto y un error de validación bajo se tiene un problema por sesgo, es decir *underfiting*. Lo indicado es buscar un nivel óptimo donde se decida la complejidad del modelo de acuerdo con los datos disponibles y no de acuerdo con la complejidad de la función que el investigador supone [30].

La efectividad de los modelos se probará utilizando las medidas resultantes de la Matriz de confusión y el área bajo la curva ROC. El propósito es determinar qué modelo da el mayor porcentaje de predicciones correctas para diagnosticar pacientes con una enfermedad reumática. Utilizando la matriz de confusión, se pueden obtener las siguientes tasas que proporcionan mucha información acerca del modelo generado:

Accuracy: proporción del número de predicciones correctas.

Valor predictivo positivo (VPP): casos positivos correctamente clasificados.

Esperado/Obtenido	**Positivo**	**Negativo**	
Positivo	Verdadero positivo (VP)	Falso positivo (FP)	*VPP=VP/(VP+FP)*
Negativo	Falso negativo (FN)	Verdadero negativo (VN)	VPN=FN/(FN+VN)
	Sensibilidad=VP/(VP+FN)	*Especificidad=FP/(FP+VN)*	*Accuracy=(VP+VN)/n*

Cuadro 1.1: Matriz de confusión

Valor predictivo negativo (VPN): casos negativos correctamente clasificados.

Sensibilidad: porcentaje de pacientes que detecta el test como afectados entre los que efectivamente presentan la alteración o enfermedad. Mide la capacidad del test de detectar los casos.

Especificidad: porcentaje de pacientes que el test identifica como sanos entre los que no presentan la alteración. Mide la capacidad del test para discriminar los casos que no presentan el evento de interés.

8. Resultados

En este apartado se presentan los datos relevados por el cuestionario COPCORD para México, un total de 25805 personas fueron clasificadas según la presencia o ausencia de alguna enfermedad reumática. A continuación se muestran las características de dichos sujetos, en la Figura 1.14 se observa que los individuos analizados nacieron en los estados Chihuahua, Distrito Federal, Nuevo Oaxaca, Sinaloa y Yucatán; se destaca que aproximadamente de las 4059 personas encuestadas en el Distrito Federal el 41.0 % fueron diagnosticadas con una enfermedad reumática siendo el estado con mayor porcentaje de casos encontrados.

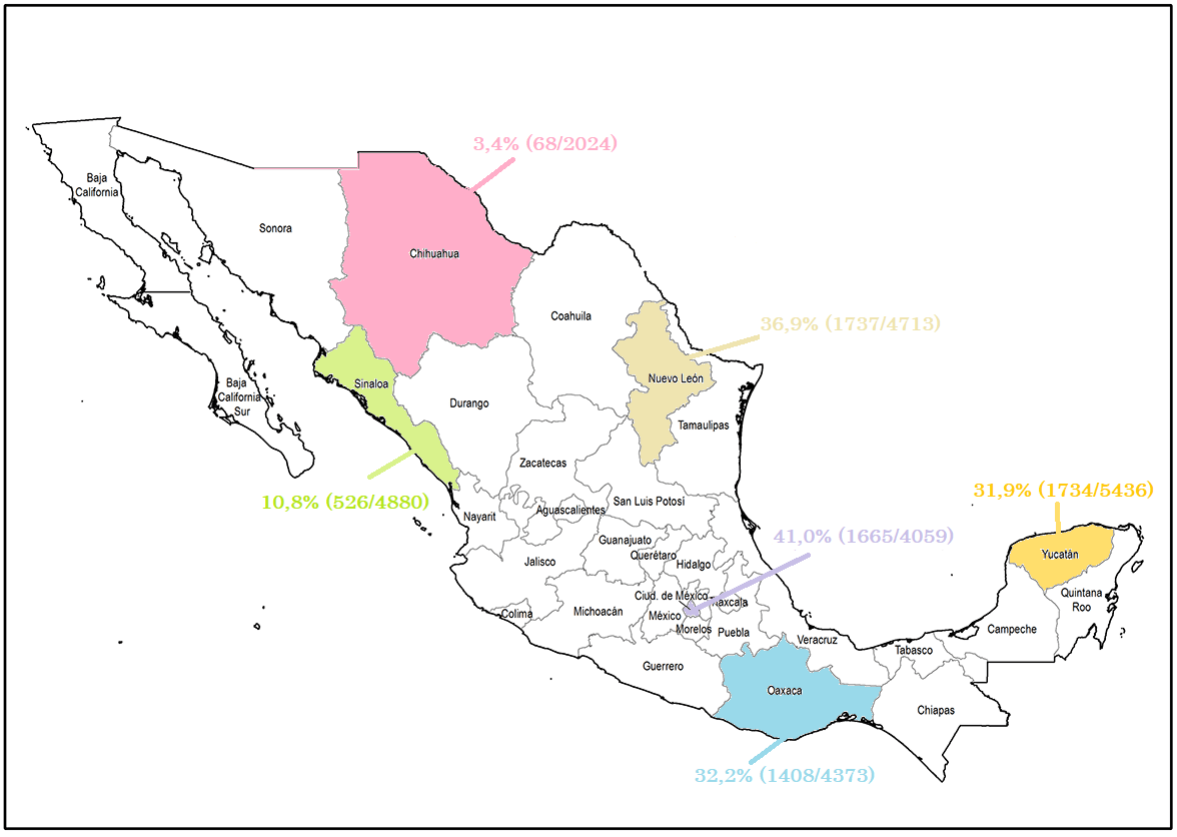


Figura 1.14: Prevalencia de las enfermedades reumáticas según estado

La prevalencia total de casos con enfermedades reumática fue 27.7 % con un Intervalo de confianza del 95 % ($IC_{95\,\%}$)(27,2 % − 28,3 %).

Las características de los individuos: edad, sexo, estado civil y si trabaja, se muestran en la Figura 1.15 según el diagnóstico final del médico; la edad promedio de los individuos con ER es mayor y es más frecuente en las mujeres. El porcentaje de pacientes que no trabajan y tienen ER es levemente mayor al porcentaje de pacientes que trabajan y tienen ER. En base a lo observado pareciera que las ER son más frecuentes en personas de mayor edad, en mujeres, en individuos unidos y que no trabajan.

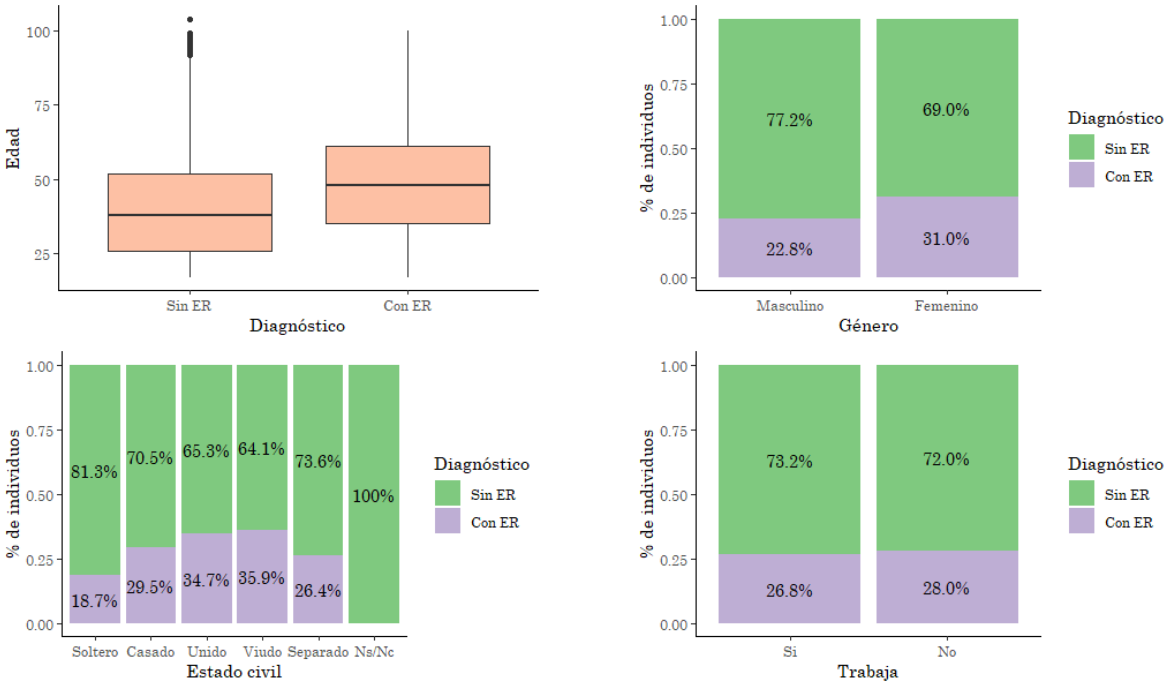


Figura 1.15: Características según diagnóstico

En la Tabla 1.2 se muestra la distribución de los individuos según la presencia de ER y la ausencia o presencia de cada comorbilidad. Ninguna de las comorbilidades pareciera asociarse directamente con la presencia de enfermedad reumática pero se observa un mayor diferencia en los porcentajes de presencia en ER entre los individuos según depresión y ansiedad.

En la Figura 1.16 se observa la distribución de los individuos según el número total de comorbilidades que presentan y el evento de interés. Se destaca que la mediana es mayor en los individuos con presencia de al menos una enfermedad musculo-esquelética. Se observan, también, algunos valores atípicos en pacientes con más de 5 comorbilidades que no tienen ninguna enfermedad reumática.

Se indagó sobre el dolor dentro de los últimos 7 días y el dolor histórico; se obtuvo que el 27.7 %

Comorbilidad	Ausencia de comorbildad y presencia de ER	Presencia de comorbildad y presencia de ER
Gastritis	27.2 %	41.9 %
Hipertensión Arterial	27.3 %	43.0 %
Diabetes	28.7 %	39.2 %
Obesidad	27.9 %	45.1 %
Tabaquismo	29.4 %	33.5 %
Depresión	27.6 %	54.1 %
Ansiedad	27.9 %	54.3 %
Alcoholismo	28.9 %	39.9 %
Problemas Cardiovasculares	29.1 %	46.8 %
Drogadicción	29.6 %	37.9 %

Cuadro 1.2: Distribución de los individuos según comorbilidades

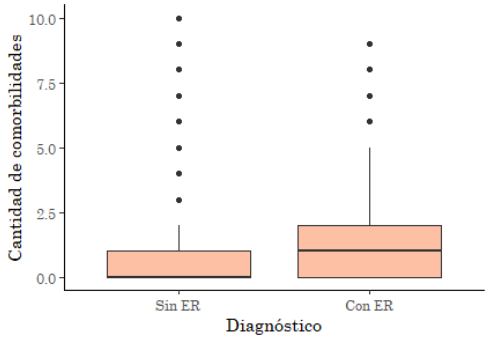


Figura 1.16: Cantidad de comorbilidades según la presencia de enfermedad reumática

de los 25805 presentaron dolor dentro de los últimos 7 días y el 9.4 % presentó dolor histórico. Dentro de los 7139 entrevistados con presencia de dolor en la última semana el 58.2 % fueron diagnosticados con presencia de ER y entre los 2417 individuos con dolor histórico, en al menos una articulación, el porcentaje de ER fue del 82.9 %, en ambos casos los porcentajes son superiores con respecto a la presencia de ER entre los individuos sin dolor; estos resultados se pueden visualizar en la Figura 1.17. En dicha figura se visualiza, además, la distribución de la cantidad de articulaciones con dolor, la intensidad y el tiempo de duración del dolor solo en los individuos que manifestaron dolor en alguna articulación; no se observan grandes diferencias en dichas variables entre los individuos con y sin el evento de interés; de forma exploratoria pareciera que estas variables no contribuyen a discriminar la presencia de alguna ER.

Sin embargo, muchos de los pacientes con presencia de dolor pueden deberse a la causa de un traumatismo previo. Es por eso que en la Figura 1.18 se analiza el dolor (reciente o histórico) según

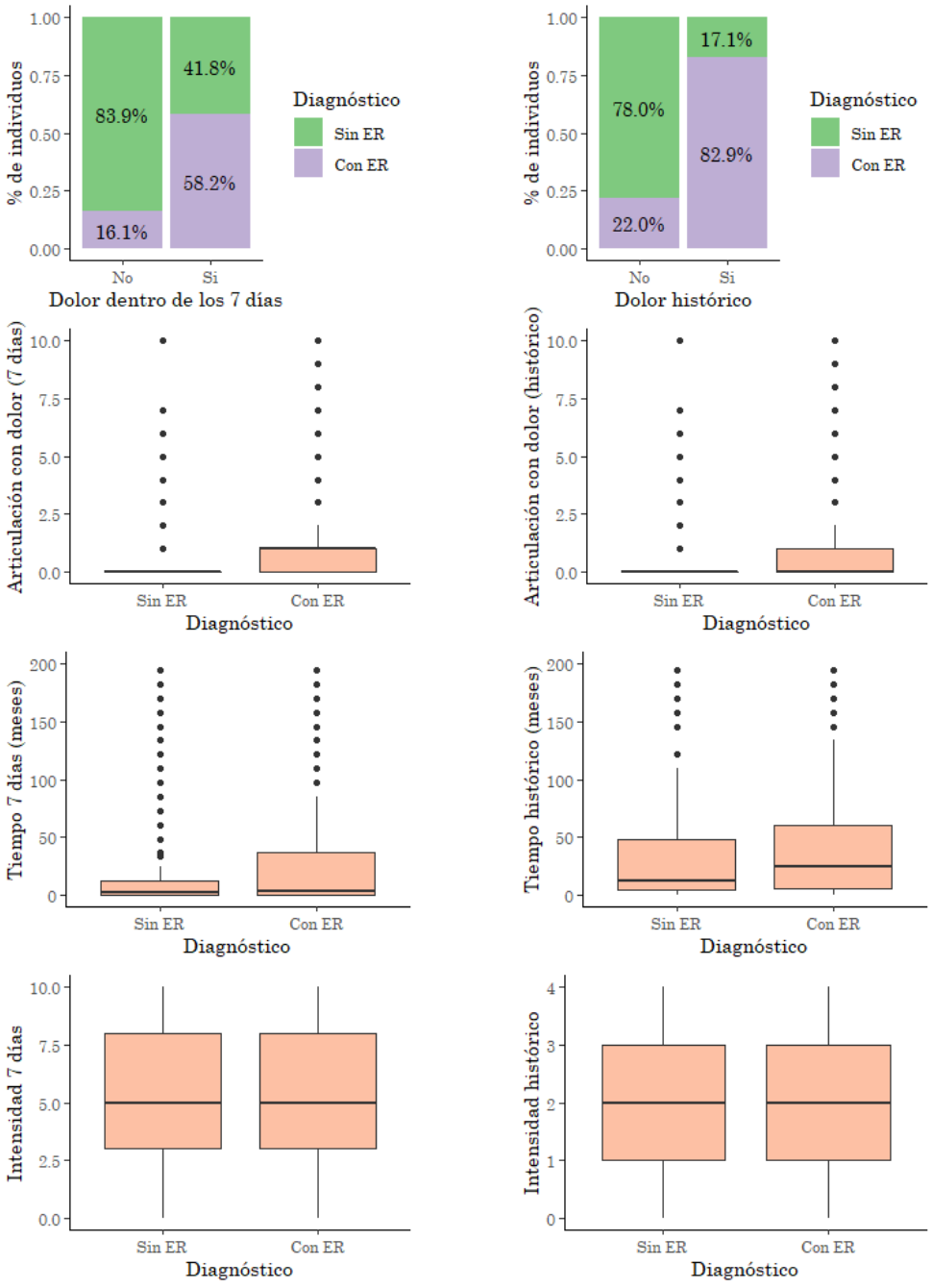


Figura 1.17: Caracterización del dolor histórico y a los 7 días

la presencia de traumatismo; en la misma se destaca que el porcentaje de individuos con ER y con dolor es similar entre los pacientes que tuvieron traumatismo y los que no tuvieron. Hay sólo 218 individuos que tuvieron traumatismo y que no tuvieron dolor.

Dentro del cuestionario se indagó la localización del dolor en los últimos 7 días, en la Figura 1.19 se observa que la rodilla fue la articulación más afectada con un 12.0 % de los 25805 individuos;

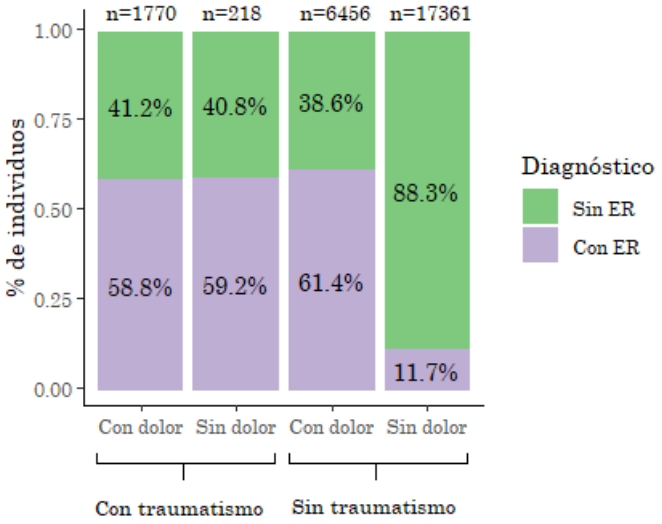


Figura 1.18: Distribución de los individuos según la presencia de dolor y traumatismo

el 25.8 % de los 7158 pacientes con ER habían manifestado dolor en la rodilla en la última semana, mientras que sólo un 6.6 % de los individuos sin el evento de interés presentaron un dolor en la rodilla en la última semana. La mano fue otra de las articulaciones con alta frecuencia de dolor, donde afecta al 7.7 % y el 16.9 % de los 7158 que fueron diagnosticados con ER manifestaron dolor en las manos. Las partes del cuerpo menos afectadas por el dolor fueron el tórax, la nuca y el antebrazo.

En la Figura 1.20 se observa la distribución de los individuos según el grado de dificultad para hacer algunas actividades y la presencia o ausencia de ER. En todos los casos se observa que los pacientes con ER tienen más limitaciones para realizar las actividades, especialmente hay un alto porcentaje de individuos con ER que no pueden realizar las actividades de arrodillarse (13,1 %) y de cortar con cuchillo (12,4 %).

Luego de este análisis exploratorio, se aplica el algoritmo Boruta para ver la importancia de cada una de ellas sobre el evento de interés. En la Figura 1.21 se muestran los resultados obtenidos siendo las variables: Adaptación malestar muculoesquelético, comunidad, presencia de dolor histórico y dolor en la última semana, estado, intensidad de dolor a los 7 días y la edad, son las que presentaron mayor importancia sobre la presencia de una ER; se observa, además, que hay variables que según este algoritmo no aporta información en la predicción y son la presencia de dolores en algunas de las partes del cuerpo como brazo, muslo, antebrazo, nuca y pierna.

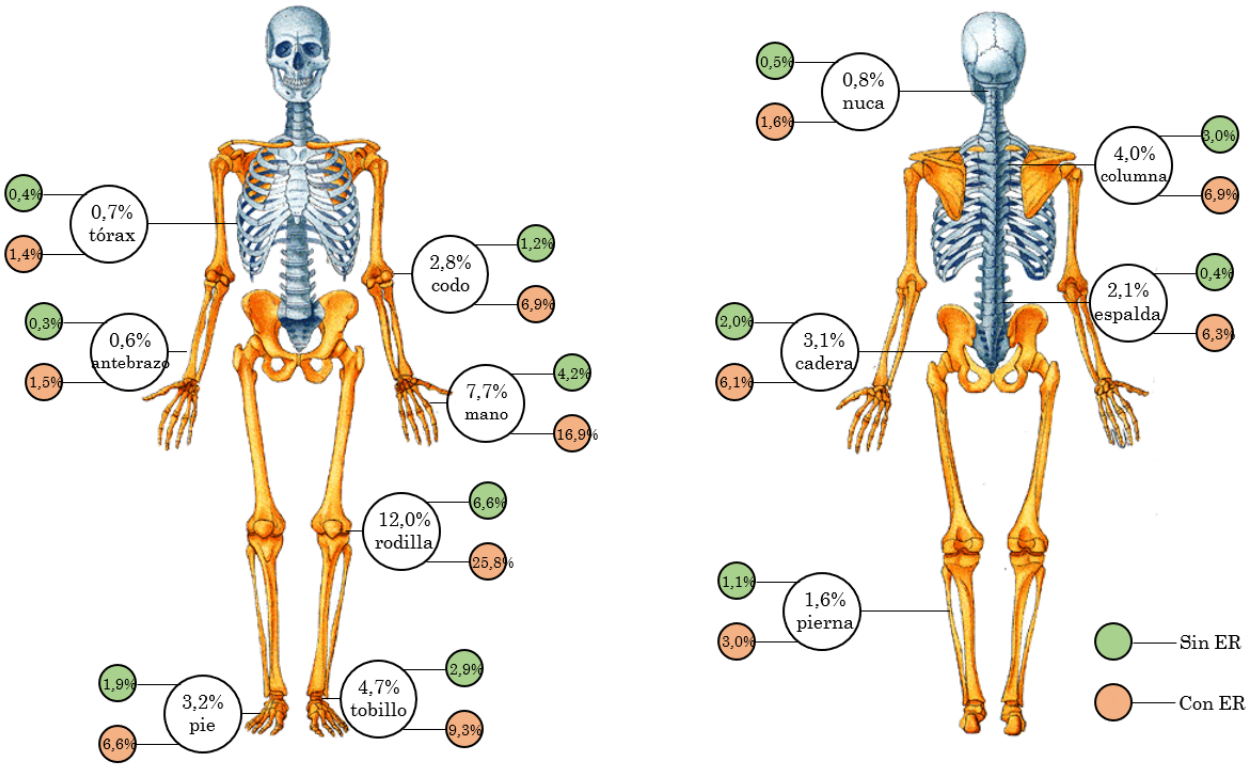


Figura 1.19: Porcentaje de individuos con dolor en las diferentes articulaciones

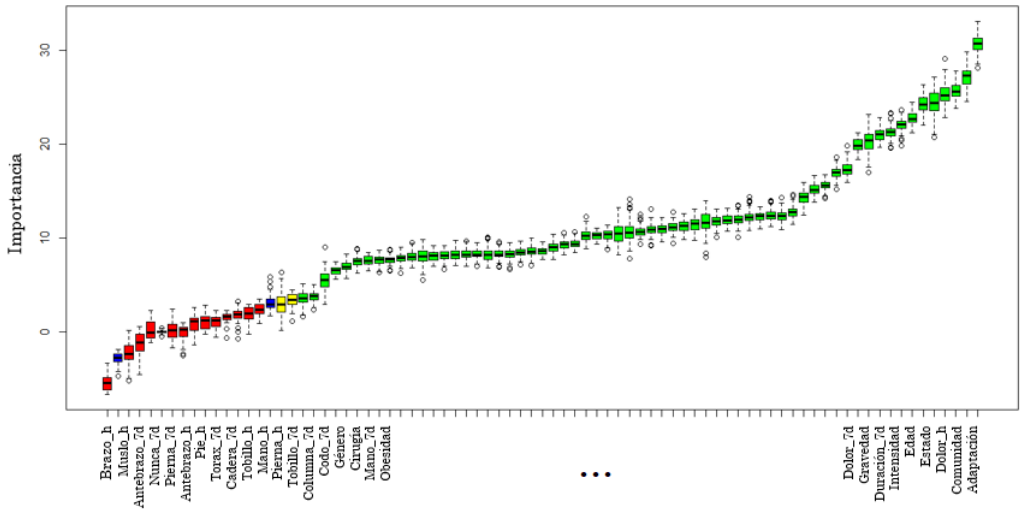
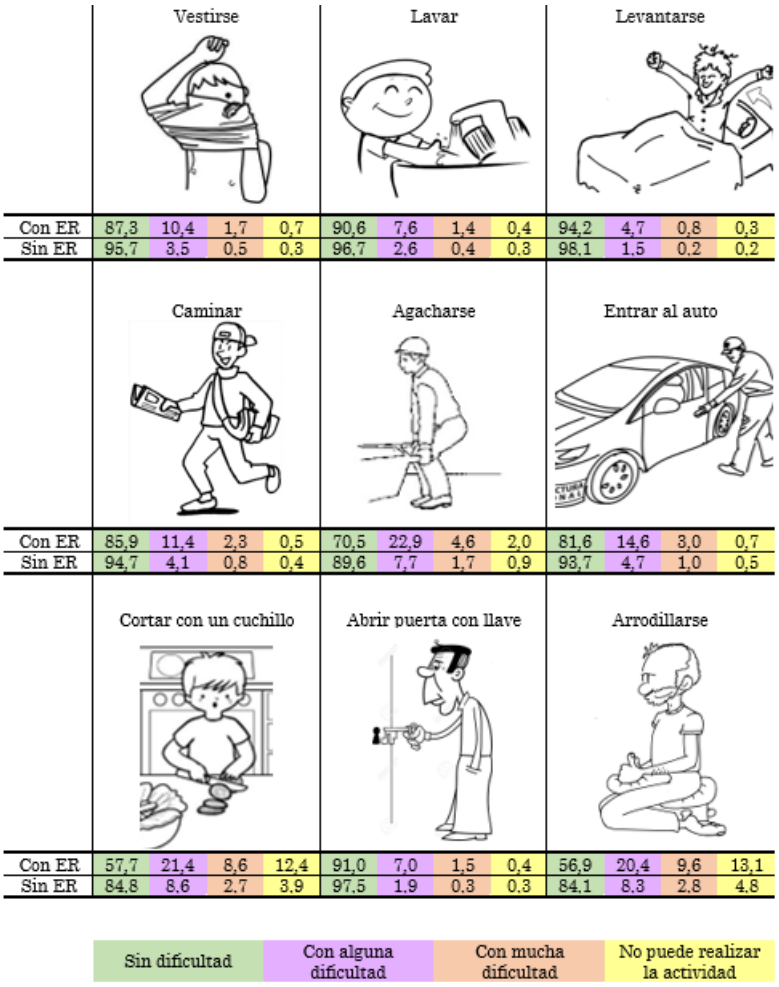


Figura 1.21: Importancia de las variables según el algoritmo Boruta

Para responder al objetivo se aplican los modelos desarrollados en el aparatado 7.3 y para poder hacer la comparación entre ellos se define el mismo conjunto de entrenamiento y de testeo para



Con ER	87,3	10,4	1,7	0,7	90,6	7,6	1,4	0,4	94,2	4,7	0,8	0,3
Sin ER	95,7	3,5	0.5	0.3	96,7	2.6	0.4	0.3	98,1	1.5	0.2	0.2
Con ER	85,9	11,4	2,3	0,5	70,5	22,9	4,6	2,0	81,6	14,6	3,0	0,7
Sin ER	94,7	4,1	0,8	0,4	89,6	7,7	1,7	0,9	93,7	4,7	1,0	0,5
Con ER	57,7	21,4	8,6	12,4	91,0	7,0	1,5	0,4	56,9	20,4	9,6	13,1
Sin ER	84,8	8.6	2.7	3.9	97,5	1.9	0.3	0.3	84,1	8.3	2.8	4.8

Figura 1.20: Distribución de los individuos según las actividades diarias

todos los modelos planteados, considerando el 80.0 % y 20.0 % respectivamente. En todos los casos se usa *Cross-Validation* con 10 repeticiones del entrenamiento.

Regresión logística A partir de este modelo se obtuvo que influyen significativamente las siguientes variables:

- Presencia de diabetes.
- Presencia de hiperlepidemia.
- Edad.
- Sexo.
- Dolor en los últimos 7 días.
- Algún traumatismo.

- Intensidad del dolor.
- Limitaciones.
- Consumo de medicamento.
- Si requirió fisioterapia.
- Si diagnósticos previos.
- Adaptación malestar muculoesquelético.
- Si puede caminar solo.
- Si puede inclinarse.
- Si puede abrir el auto.
- Si puede salir del auto solo.
- Si puede cortar.
- Cantidad de articulaciones con dolor.
- Dolor combinado con traumatismo.
- Gravedad de la molestia.

Este modelo tiene la ventaja de poder interpretar cada una de estas variables en términos de Razones de Odds a partir de los coeficientes obtenidos del modelo, por ejemplo, la chance de tener una ER es aproximadamente 3.8 veces mayor para los sujetos con dolor dentro de la última semana.

Árbol CART Para el algoritmo CART se utilizaron los algoritmos *rpart* [31], que utiliza el parámetros de complejidad el cual controla el tamaño del árbol, es decir, si el costo de agregar una nueva variable desde el nodo actual está por encima del valor fijado de cp el algoritmo no continua, y *rpart2*, que utiliza la profundidad del árbol. Se obtuvo mejores resultados con rpart por lo tanto se utiliza este último para la comparación con el resto de las técnicas.

Árbol CHAID En la Tabla 1.3 se muestran la importancia de las variables obtenidas con el algoritmo CHAID, se observa que la combinación entre dolor y traumatismo fue la que más aporta al momento de predecir seguida por la presencia de dolor en los últimos 7 días.

Variables	Importancia
Dolor y traumatismo	100.00 %
Dolor en los últimos 7 días	86.44 %
Dolor en al menos una articulación en los últimos 7 días	76.97 %
Adaptación malestar musculoesquelético	70.34 %
Dolor histórico	62.95 %
Medicamento	59.17 %
Dolor en al menos una articulación histórica	48.94 %
Gravedad de la molestia	46.91 %

Cuadro 1.3: Importancia de las variables según el algoritmo CHAID

Random Forest Se utilizó la librería *randomforest* [32], fue entrenado con diversos números de árboles (200,300,500,1000,1500 y 2000) y con diversos valores de mtry (2 a 10). Se obtuvo que el modelo con mejores resultados fue el que tuvo los siguientes parámetros: mtry=10, ntree=300

Como se mencionó anteriormente, el modelo de Random Forest tiene la ventaja de que permite conocer cuáles son las características más relevantes. La Figura 1.22 muestra las características ordenadas según su importancia en el eje y una estimación de su importancia en el eje x según el valor del Gini. En este caso se observa que la variable que más aporta en la predicción es la adaptación musculo-esquelética.

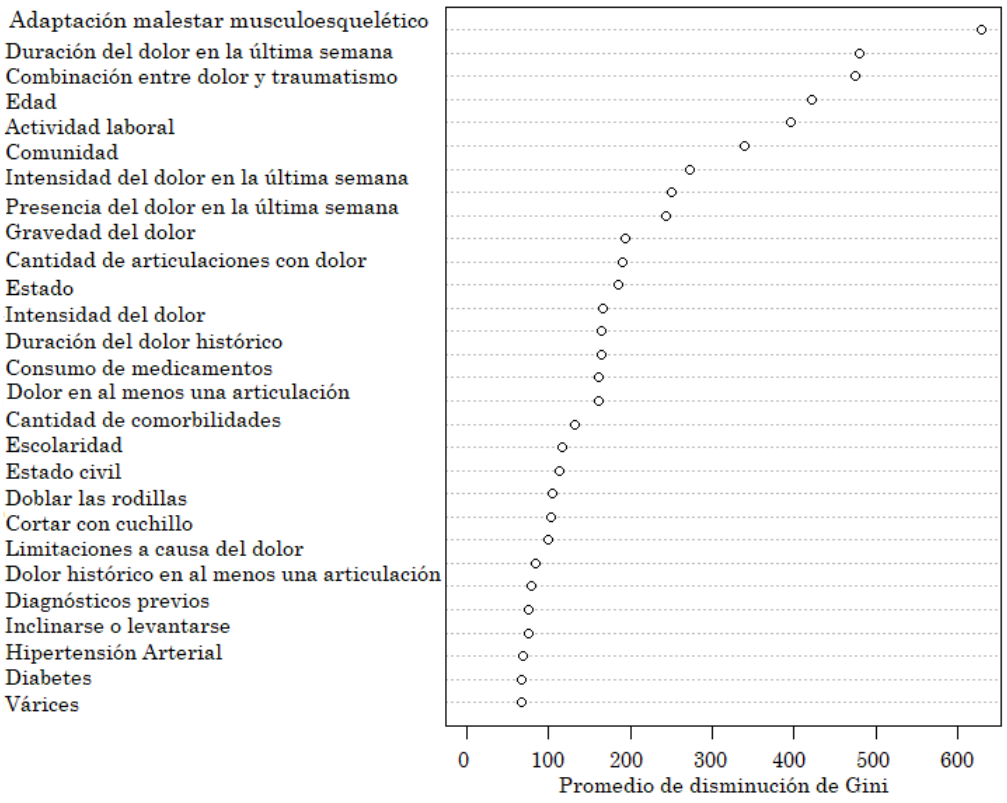


Figura 1.22: Importancia de las variables según el algoritmo Random Forest

XGBOOST Se aplicó este algoritmo a través de la función *xgboost* con los siguientes parámetros:

- eta, se refiere a la tasa de aprendizaje. Varía entre 0 y 1, menor valor significa que el modelo es más robusto a *overfitting*. En este caso se estableció eta=0.032.
- min_child_weight, suma mínima de los pesos necesaria para un nodo hijo. En este caso se estableció min_child_weight=7.756.
- max_depth, máxima profundidad de un árbol, en esta caso se estableció en 13.
- gamma, reducción de pérdida mínima para realizar una partición adicional en un nodo de hoja del árbol. Cuanto más grande gamma, más conservador será el algoritmo. En este caso se estableció 3.267974
- nround, cantidad de pasada de los datos. En este caso se estableció 186 veces.
- colsample_bytree, proporción de la submuestra de columnas cuando se construye cada árbol. En este caso se estableció 0.925.

En la Tabla 1.4 se presenta las variables con mayor importancia en la predicción, siendo la combinación entre dolor y traumatismo la que mejor clasifica seguida por la presencia de diabetes y el estado de nacimiento.

Variables	**Gain**
Dolor y traumatismo	0.3994
Diabetes	0.0719
Estado	0.0654
Dolor histórico	0.0653
Adaptación malestar muculoesquelético	0.0477
Duración del dolor en los últimos 7 días	0.0429
Comunidad	0.0328

Cuadro 1.4: Importancia de las variables según el algoritmo XGBOOST

Naïve Bayes

El modelo crea la probabilidad condicional para cada característica por separado. Al ser un algoritmo paramétrico que no tiene variación aleatoria, la única forma de mejorar este algoritmo es incorporando más datos o más variables. Para su implementación se utilizó la librería *mlr*.

SVM

Se utilizó la función *svm* del paquete *e1071* [33]. El modelo de SVM fue entrenado con diferentes kernels: *sigmoidal*, *polinomial* de grado 1,2 y 3, *spline* y se obtuvo un mejor poder predictivo con el kernel sigmoidal.

Redes Neuronales

Para la implementación de la red neuronal se usó el paquete *h2o* [34], el cual es una librería para análisis predictivo y *Machine Learning* que incorpora funciones para crear redes neuronales artificiales.

Como función de activación se utilizó la tangente con 3 capas ocultas. Se definió que se pasarán los datos de entrenamiento 100 veces a fin de aplicar el algoritmo de aprendizaje.

En la Figura 1.23 se muestran los resultados del *accuracy*, la sensibilidad y especificidad obte-

nidos para todos los modelos en los 10 entrenamientos realizados a través del *Cross-validation*. Se observa que el algoritmo CART tiene valores más alto en la especificidad pero en la sensibilidad es el que presenta menor valor; todos los modelos presentaron una mediana de *accuracy* mayor al 80 %. El algoritmo de *Random Forest* es el que presenta la mediana del *accuracy* y de la sensibilidad más alta, es decir, tiene una buena capacidad para detectar individuos con la enfermedad entre los sujetos que la presentan.

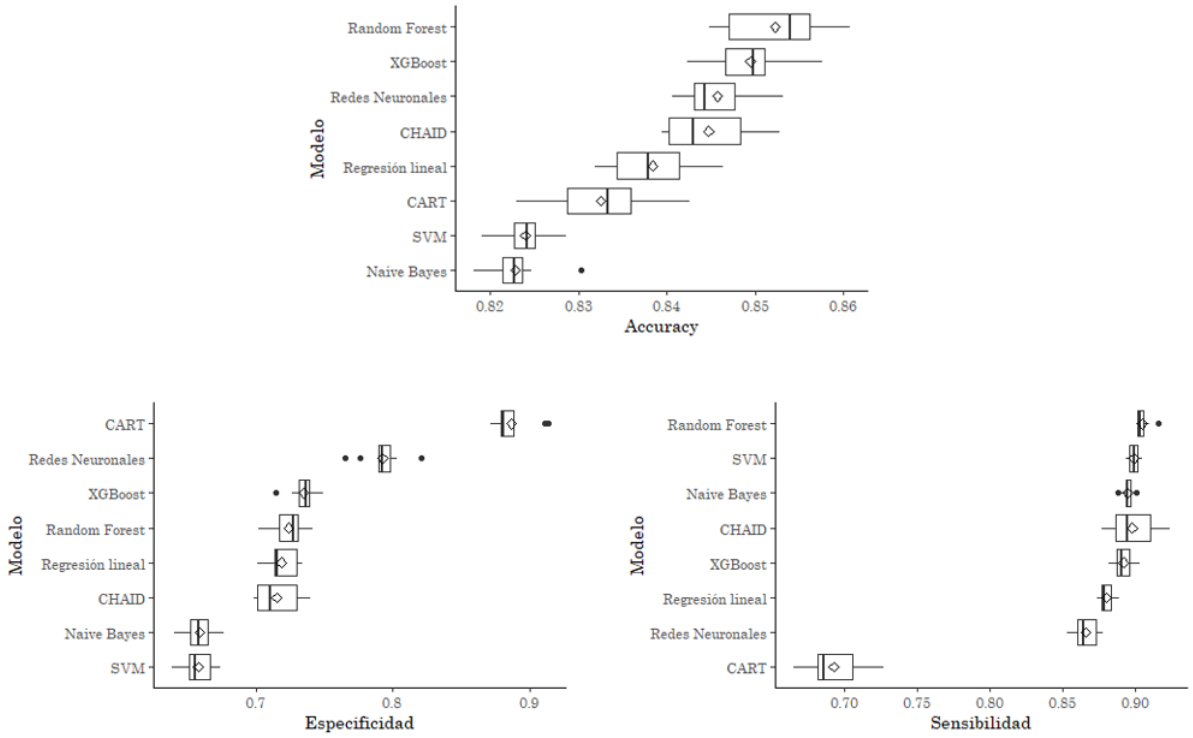


Figura 1.23: Comparación en los modelos a través del *Cross-validation*

Luego se utilizaron los restantes 20 % de los datos para evaluar la precisión de cada uno de estos modelos en nuevos datos; estos resultados se visualiza en la Tabla 1.5. Se observa que todos los modelos tuvieron medidas resúmenes similares presentando *accuracy* entre 0.82 y 0.85 y valor del área bajo la curva entre 0.84 y 0.91. El mayor *accuracy* lo obtuvo el modelo *Random Forest* pero observando el área bajo la curva el modelo de Redes Neuronales sería el seleccionado. Es por esto que es relevante analizar diferentes medidas de la capacidad predictiva. En la Figura 1.24 se muestran las curvas ROC de todos los modelos, no se observa grandes diferencias entre ellas, sin embargo se destaca que las curvas de los modelos CART y Naive Bayes están levemente por debajo de las restantes y esto se confirma observando los valores de AUC (0.8441 y 0.8595 respectivamente).

Modelo	Accuracy (IC 95 %)	Sensibilidad	Especificidad	AUC
Regresión logística	0.8295 (0.8189, 0.8396)	0.7033	0.8744	0.8991
CART	0.8378 (0.8274, 0.8478)	0.7089	0.8868	0.8441
CHAID	0.8432 (0.833, 0.853)	0.7788	0.8638	0.9133
Random Forest	0.8475 (0.8374, 0.8572)	0.7130	0.9032	0.9094
XGBOOST	0.8471 (0.837, 0.8568)	0.7283	0.8916	0.9147
Naive Bayes	0.8178 (0.807, 0.8283)	0.6559	0.8890	0.8595
SVM	0.8202 (0.8094, 0.8305)	0.6553	0.8955	0.8703
Redes neuronales	0.8386 (0.8282, 0.8485)	0.7925	0.8563	0.9134

Cuadro 1.5: Comparación de resultados de los modelos

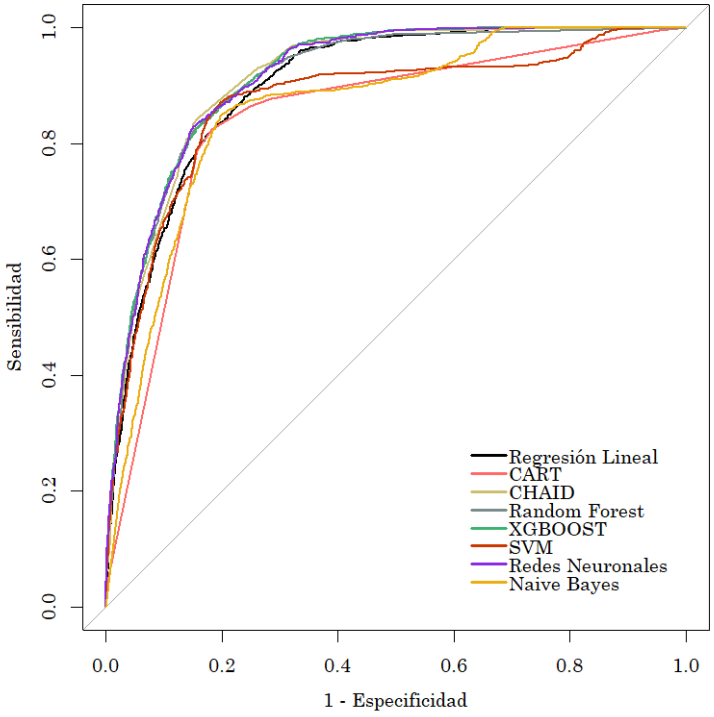


Figura 1.24: Curvas ROC para los modelos planteados

Analizando las ventajas y desventajas de cada modelo, la velocidad en la implementación y los resultados visualizados en la Tabla 1.5 se decide implementar el modelo *Random Forest* para la predicción de nuevos casos de pacientes con enfermedad reumática.

9. Consideraciones finales

La prevalencia de casos hallados con ER a través del cuestionario COPCORD fue aproximadamente del 30 %. En el análisis exploratorio se observó un alto porcentaje de pacientes con ER entre los pacientes que presentaban dolor y en especial entre los que tenían dolor de rodilla. El porcentaje de individuos con dolor y que presentaban ER fue muy superior al porcentaje de pacientes sin dolor y que fueron identificados como casos, en base a esto pareciera que la presencia del dolor que no sea por causa de alguna lesión es un síntoma que aporta mucha información al momento de clasificar a los individuos. Se destaca, también, un mayor porcentaje de individuos que pueden realizar tareas comunes como vestirse, agacharse, subir al auto, cortar en trozos o arrodillarse pero con un cierto grado de dificultad con respecto a los encuestados sin ninguna enfermedad reumática.

Para responder al objetivo, se lograron aplicar ocho técnicas de minería de datos para clasificar a los pacientes con enfermedades reumáticas; para todos los modelos se utilizó el mismo conjunto de entrenamiento y de testeo para poder comparar los resultados entre ellos. Todos los modelos presentaron valores de efectividad similares y, en general, por encima del 70 %.

A partir de la comparación entre los distintos modelos se obtuvo que *Random Forest* fue el más adecuado, dentro de los modelos evaluados, para predecir nuevos pacientes con enfermedades reumática. Este algoritmo es muy útil y fácil de usar, ya que los hiperparámetros predeterminados a menudo producen un buen resultado de predicción, además, los hiperparámetros que utiliza son fáciles de entender y se pueden conocer las variables que más aportan al momento de la predicción. Las características que presentaron una mayor importancia en la predicción según el modelo de *Random Forest* utilizando el Gini fueron: Adaptación malestar musculoesquelético, duración del dolor, combinaciones entre dolor y presencia de traumatismo, edad y actividad laboral.

Como siguiente paso, se busca incorporar los resultados de las encuestas realizadas en otros países de latinoamérica para aumentar la precisión de lo modelos y poder obtener resultados más generales y luego se planea implementar el modelo en una plataforma *online* y colocarlo en todos los efectores de salud para que la persona auto-complete el cuestionario con determinadas carac-

terísticas y se pueda obtener de manera automática la probabilidad de presentar una enfermedad reumática o no. El modelo encontrado tiene como fin de ayudar a los médicos a detectar más tempranamente la presencia de alguna enfermedad reumática para evitar consecuencias más graves.

Bibliografía

[1] Ministerio de Sanidad, Servicios Sociales e Igualdad (2013). "Estrategia en enfermedades reumáticas y musculoesqueléticas del Sistema Nacional de Salud". Madrid

[2] John Londoño, et al. (2018). "Prevalencia de la enfermedad reumática en Colombia, según estrategia COPCORD-Asociación Colombiana de Reumatología. Estudio de prevalencia de enfermedad reumática en población colombiana mayor de 18 años". Asociación Colombiana de Reumatología. Elsevier España, S.L.U. 25(4): 245-256.

[3] Sociedad Española de Reumatología [en línea]. "Enfermedades reumáticas: las preguntas de los pacientes". Disponible en: https://www.pfizer.es/Assets/docs/pdf/libros/Enfermedades_reumaticas_las_preguntas_de_los_pacientes.pdf [citado 10 de marzo de 2019].

[4] "Community Oriented Program for Control of Rheumatic Diseases (COPCORD)"[en línea]. Disponible en: http://copcord.org/tools.asp [citado 8 de marzo de 2019].

[5] Rosana Quintana, Adriana M. R. Silvestre, Mario Goñi, et al. (2016). "Prevalence of musculoskeletal disorders and rheumatic diseases in the indigenous Qom population of Rosario, Argentina". Clin Rheumatol 35 (Suppl 1):S5–S14

[6] Organización Mundial de la Salud (1992). "Enfermedades reumáticas". Serie de informes técnicos; 816).

[7] Sociedad Argentina de Reumatología [en línea]. "Enfermedades reumáticas, enfermedades que no esperan". Disponible en: http://www.reumatologia.org.ar/enfermedades_reumaticas.php [citado 15 de marzo de 2019].

[8] Filip Raciborski, Anna Kłak, Brygida Kwiatkowska, et al. (2017). "Diagnostic delays in rheumatic diseases with associated arthritis". Reumatología, 55(4): 169–176.

[9] Miron Bartosz Kursa [en linea]. "Wrapper Algorithm for All Relevant Feature Selection"[citado mayo de 2019]. https://cran.r-project.org/web/packages/Boruta/Boruta.pdf

[10] Manish Pathak [en linea]. "Feature Selection in R with the Boruta R Package". Disponible en: https://www.datacamp.com/community/tutorials/feature-selection-R-boruta [citado mayo de 2019].

[11] Guillermo Molero Castillo (2008). "Desarrollo de un modelo basado en técnicas de Minería de Datos para clasificar zonas climatológicamente similares en el estado de Michoacán". Universidad Nacional Autónoma de México.

[12] Timarán-Pereira, S. R., Hernández-Arteaga, I., Caicedo-Zambrano, S. J., Hidalgo-Troya, A. y AlvaradoPérez, J. C. (2016). "El proceso de descubrimiento de conocimiento en bases de datos. En Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional". (pp. 63-86). Bogotá: Ediciones Universidad Cooperativa de Colombia.

[13] Cecilia Papalardo (2013). "Predicción de displasia broncopulmonar en bebés prematuros y de muy bajo peso al nacer". Facultad de Ciencias Económicas y Estadística.

[14] Pang-Ning Tan, Michael Steinbach y Vipin Kumar (2006). "Introduction to Data Mining - Data Mining Classification: Basic Concepts, Decision Trees, and Model Evaluation".

[15] Escobar Gonalez (2018). "Clasificación de imágenes satelitales mediante algoritmo CHAID". Facultad de Ciencias Económicas y Estadística.

[16] Jorge Martín Arevalillo. "Data Mining con Árboles de Decisión". Dpto. Estadística, Investigación Operativa y Cálculo Numérico. Universidad Nacional Educación a Distancia, Madrid.

[17] Sara Hidalgo Ruiz-Capillas (2014). "Random Forests para detección de fraude en medios de pago". Universidad Autónoma de Madrid.

[18] Gareth James,Daniela Witten,Trevor Hastie,Robert Tibshirani (2013). "An Introduction to Statistical Learning with Applications in R".

[19] Navnina Bhatia [en linea] "What is Out of Bag (OOB) score in Random Forest?". Towards Data Science. Disponible en: https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710 [citado en julio 2019].

[20] Víctor Pita González-Campos (2017). "Modelado mediante Random Forest de la emisiones de autobuses en función de los ciclos cinemáticos". Universidad Politécnica de Madrid.

[21] Baylé, Federico. (2016) "Detección de villas y asentamientos informales en el Partido de La Matanza mediante teledetección y sistemas de información geográfica". Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires.

[22] Ziad Patrous (2018). "Evaluating XGBoost for User Classification by using Behavioral Features Extracted from Smartphone Sensors". School of Electrical Engineering and Computer Science, KTH.

[23] Maximiliano Neustadt (2011). "Modelos de sentimiento no dependientes de dominio en español". Universidad de Buenos Aires.

[24] González Abril. "Modelos de Clasificación basados en Máquinas de Vectores Soporte". Departamento de Economía Aplicada I. Universidad de Sevilla.

[25] Leonardo Jiménez Moscovitz Pervys Rengifo (2010). "Al interior de una máquina de soporte vectorial". Facultad de Ciencias Naturales y Exactas Universidad del Valle.

[26] Sebastián Maldonado, Richard Weber (2012). "Modelos de Selección de Atributos para Support Vector Machines". Revista Ingeniería de Sistemas Volumen XXVI.

[27] Rafael Correa Deves (2006) "Redes Neuronales Artificiales en Ingeniería Nuclear. Caracterización de espectros PIXE ". Universidad de Granada.

[28] Salvador Torra Porras (2003). "Herramientas de extracción de información: Redes Neuronales". Departament D'Econometria, Estadística I Economía Espanyola. Univesitat de Barcelona.

[29] Jürgen Schmidhuber (2014). "Deep learning in neural networks: An overview". Neural Networks 61 (2015) 85–117.

[30] Gisela Isabel García Gazabón. "Modelo de Machine Learning para la Clasificación de pacientes en términos del nivel asistencial requerido en una urgencia pediátrica con Área de Cuidados Mínimos". Maestría de Ingeniería Cartagena. Universidad Tecnológica de Bolívar.

[31] Brian Ripley, Terry Therneau, Beth Atkinson (2019). "Recursive Partitioning and Regression Trees". R package Version 4.1-15.

[32] Leo Breiman, Adele Cutler (2018). "Breiman and Cutler's Random Forests for Classification and Regression". R package Version 4.6-14.

[33] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch (2019). "Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien". R package Version 1.7-2.

[34] Erin LeDell, Navdeep Gill, Spencer Aiello, *et al.* "R Interface for 'H2O' ". R package Version 3.26.0.2.