Universitat Oberta de Catalunya

Análisis de los factores de riesgo de la enfermedad del Alzheimer y su detección temprana mediante machine learning

Hugo Fernández Cobas

Trabajo de Fin de Máster
Máster en Bioinformática y Bioestadística

Ivette Olivares Castiñeira
(Carles Ventura Royo)

24 de diciembre de 2021

FICHA DEL TRABAJO FINAL

Título:	Análisis de los factores de riesgo de la enfermedad del Alzheimer y su detección temprana mediante machine learning
Nombre autor/a:	Hugo Fernández Cobas
Nombre PDC:	Ivette Olivares Castiñeira
Nombre PRA:	Carles Ventura Royo
Fecha de entrega:	24 de diciembre de 2021
Titulación:	Máster en Bioinformática y Bioestadística
Área:	Trabajo de Fin de Máster
Idioma:	Castellano
Núm. de créditos:	15
Palabras clave:	Alzheimer, Demencia, Machine Learning

Resumen

Actualmente, el desarrollo de nuevas tecnologías como es el caso del machine learning (ML) ha provocado un aluvión de posibilidades de uso para la resolución de problemas. En nuestro trabajo, utilizaremos esta tecnología para hacer un estudio sobre la enfermedad del Alzheimer que abarca: un análisis para conocer los diferentes factores de riesgo, la aplicación de algoritmos de ML para prevenir la enfermedad a partir del estudio de estos factores y la detección de enfermos a partir de imágenes de resonancia magnética (MRI) mediante una red neuronal convolucional (CNN). Con ello, se busca una mejora en la detección temprana y prevención de la enfermedad lo que nos permitiría aplicar los tratamientos existentes para tratar de ralentizar su avance. Este tipo de métodos tal vez no son del todo eficientes por sí solos, ya que obtuvimos unos valores de precisión de aproximadamente el 60 %, pero al ser combinados con otro tipo de pruebas (neuropsicológicas, marcadores de fluídos, etc) dan buen resultado.

Abstract

Nowadays, the development of new technologies such as machine learning (ML) has caused a flood of possibilities of use for solving problems. In our work, we will use this technology to carry out a study on Alzheimer's disease that includes: an analysis to know the different risk factors, the application of ML algorithms to prevent the disease from the study of these factors and the detection of patients from magnetic resonance imaging (MRI) using a convolutional neural network (CNN). With this, an improvement in the early detection and prevention of the disease is sought, which would allow us to apply the existing treatments to try to slow down its progress. This type of method may not be efficient on its own, because we obtained accuracies of approximately 60 %, but when combined with other types of tests (neuropsychological, fluid markers, etc.) they give good results.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y justificación del trabajo

1.1.1. Descripción general

El aumento de la esperanza de vida y, por consiguiente, de la población anciana, ha provocado un incremento del número de enfermos por demencias como el Alzheimer. Actualmente, esta enfermedad es una de las causas de muerte más elevadas en la población humana, además de ser una de las enfermedades que empeora más la calidad de vida del paciente dado que afecta a la memoria, al pensamiento y al comportamiento. Todo ello, genera la necesidad de un estudio como el que llevaremos a cabo en este trabajo con el fin de conocer mejor la enfermedad para tratar de buscar una solución y mejorar la capacidad de detección temprana de la misma.

Inicialmente, realizaremos un análisis estadístico de cada uno de los posibles factores de riesgo de la enfermedad del Alzheimer para conocer mejor su comportamiento y como estos se distribuyen en la población recogida en nuestros datos. Seguidamente, utilizaremos estos factores para realizar un modelo de regresión que nos ayudará a conocer cuáles de estas variables tienen mayor influencia sobre la probabilidad de padecer la enfermedad. Conocer esta información podría ayudar a prevenir el desarrollo de esta patología cambiando ciertos hábitos de vida o comportamientos, además de ayudarnos a entender los fundamentos básicos que la producen.

Por otro lado, en este tipo de enfermedades en las que todavía no se conoce un tratamiento capaz de revertir sus efectos debemos primar la capacidad de predecir su aparición y mejorar la detección temprana de la misma. Esto ayudaría enormemente a tratar de impedir su avance antes de producirse daños graves e irreversibles en el cerebro. Por ello, en la segunda parte de este trabajo nos centraremos en desarrollar mecanismos de predicción de posibles enfermos mediante diferentes algoritmos de machine learning utilizando datos de los pacientes.

Por último, utilizaremos imágenes de resonancia magnética (MRI) del cerebro de personas enfermas para mejorar la detección temprana de la enfermedad. Actualmente, uno de los grandes problemas en el ámbito sanitario es el nivel de saturación de hospitales y clínicas, provocando

grandes listas de espera que podrían retrasar la detección temprana de una enfermedad. Por ello, la creación de mecanismos capaces de diagnosticar la enfermedad del Alzheimer de forma rápida y eficaz pueden ayudar enormemente a que la detección sea lo más temprana posible, pudiendo así utilizar tratamientos de choque experimentales o cuidados especializados para tratar de frenar el avance de la enfermedad.

1.1.2. Justificación

El principal motivo para escoger este tema de trabajo sobre la enfermedad del Alzheimer ha sido la importancia de este tipo de estudios en la mejora de la calidad de vida de muchas personas. La comunidad científica no deja de estar caracterizada por una relación simbiótica en la que nos nutrimos de los trabajos de otros y ellos de los nuestros en busca de la solución a un problema. Por ello, mi intención con este trabajo es aportar otro enfoque o punto de vista que pueda ayudar a entender la enfermedad desde una perspectiva diferente para tratar de conseguir el objetivo final de encontrar un tratamiento capaz de revertirla o ayudar a prevenirla.

La Organización Mundial de la Salud (OMS) prevé que en 2030 el número de afectados por la enfermedad del Alzheimer y otras demencias a nivel mundial será de 75 millones, una cifra desgarradora teniendo en cuenta que se trata de una de las enfermedades más incapacitantes. Estos datos deben generar un aumento de la investigación y suscitar un mayor interés científico por esta enfermedad que sigue sin presentar un tratamiento eficaz. Debemos aprovechar el desarrollo tecnológico y metodológico de estas décadas para tratar de encontrar una solución a este problema con el fin de frenar este aumento masivo de afectados.

Por otro lado, la elección se debe también a mi gran interés por las enfermedades relacionadas con el envejecimiento. Mi trabajo de fin de grado ya trataba sobre el estudio de ensamblajes moleculares que podrían ser utilizados para el tratamiento de este tipo de enfermedades. Como hemos visto, estas patologías en las que la edad juega un papel importante se encuentran en aumento dado que la esperanza de vida cada vez es mayor. Por ello, he querido continuar con esta línea de estudio pero esta vez enfocándome más en las causas y la prevención desde el punto de vista bioinformático.

1.2. Objetivos del trabajo

Los objetivos tanto generales como específicos que se pretenden alcanzar con este proyecto son los siguientes:

1. Conocer los factores con mayor influencia en la enfermedad del Alzheimer.
    1.1 Analizar estadísticamente los factores estudiados.
    1.2 Realizar un estudio de regresión con los diferentes factores.
    1.3 Obtener conclusiones de que factores influyen más en la enfermedad.

2. Predicción de posibles pacientes enfermos utilizando diferentes variables mediante algoritmos de machine learning.

   2.1 Predecir posibles enfermos mediante los factores de riesgo de la enfermedad.

   2.2 Elegir el mejor método de predicción.

3. Detección de la enfermedad utilizando imágenes del cerebro de los pacientes mediante una red neuronal convolucional (CNN).

1.3. Enfoque y método

Para realizar este tipo de estudios analíticos y de detección y predicción existe una gran variedad de programas, softwares y aplicaciones que tienen grandes utilidades en cuanto a algoritmos de machine learning y análisis estadístico se refiere. Pero para este proyecto utilizaremos el software R y R Studio debido a ser uno de los programas más punteros en este tipo de estudios, además de contener todas las funciones y complementos necesarios para lograr nuestros objetivos. Al tener un enorme abanico de posibilidades nos permite realizar todo con cierta continuidad y en un mismo archivo lo que agiliza enormemente el proceso y aporta mayor coherencia y claridad a los resultados. La parte de detección a partir de imágenes, en cambio, la realizaremos utilizando TensorFlow en un entorno de Python debido a que presenta mayor eficiencia en este tipo de labor.

En el caso del análisis de los factores de riesgo, llevaremos a cabo un primer estudio estadístico de cada uno de ellos y posteriormente realizaremos un modelo de regresión para conocer qué grado de relación existe entre estos factores y la enfermedad. De este modo, podremos esclarecer cuáles de los factores estudiados tienen una correlación real con la enfermedad y cuáles no.

Por otro lado, las predicciones de posibles pacientes enfermos se realizarán utilizando una serie de funciones de R que nos permiten aplicar los algoritmos de machine learning. Una vez aplicados estos, tendremos que determinar cuál de ellos presenta mayor eficacia.

Por último, para la detección a partir de imágenes utilizaremos una red neuronal convolucional mediante TensorFlow en un entorno de Python que nos permitirá clasificar las imágenes según el diagnóstico del enfermo.

1.4. Planificación del trabajo

1.4.1. Planificación con hitos y temporización

El proyecto se dividirá principalmente en 5 partes que contendrán diversas tareas relacionadas entre sí, que serán:

- Búsqueda de información de la enfermedad y de bases de datos que precisamos utilizar en el proyecto.
- Análisis de los diferentes posibles factores de riesgo
- Aplicación de los algoritmos de machine learning a nuestros datos
- Obtención de conclusiones y valoración de los resultados
- Realización de las diferentes actividades

1.4.1.1. Tareas

Dentro de las diferentes partes del proyecto anteriormente mencionadas habrá una serie de tareas que deberemos llevar a cabo para conseguir los objetivos propuestos a las cuales asignaremos una duración determinada:

A Búsqueda de información de la enfermedad: sintomatología, causas, posibles tratamientos, etc. (14 días)

B Búsqueda de bases de datos para obtener el dataset o datasets de posibles factores de riesgo que utilizaremos para nuestro estudio, así como, de imágenes del cerebro de pacientes. (14 días)

C Limpieza de datos y análisis estadístico de todas las variables de nuestro estudio sobre los factores de riesgo. (9 días)

D Análisis de regresión de los diferentes posibles factores de riesgo de la enfermedad y su relación con la misma. (10 días)

E Aplicación de diferentes algoritmos de machine learning para la predicción de posibles pacientes enfermos mediante los factores de riesgo de la enfermedad. (10 días)

F Aplicación de una red neuronal convolucional para la detección de posibles enfermos mediante imágenes de pacientes. (18 días)

G Obtención de conclusiones y resultados y análisis de los mismos. (9 días)

H Preparación de la memoria y la presentación. (31 días)

I Realización de la PEC 0. (6 días)

J Realización de la PEC 1. (8 días)

K Realización de la PEC 2. (25 días)

L Realización de la PEC 3. (23 días)

M Realización de la PEC 4. (11 días)

N Realización de la PEC 5a. (6 días)

Ñ Realización de la PEC 5b. (7 días)

1.4.1.2. Calendario

Con el fin de planificar correctamente el proyecto, hemos organizado las diferentes tareas temporalmente y las hemos distribuido en un diagrama de Grant que se muestra a continuación:

	Nombre	Fecha de inicio	Fecha de fin
A	Búsqueda de información sobre la enfermedad del Alzheimer	15/9/21	4/10/21
B	Búsqueda de bases de datos y datasets	15/9/21	4/10/21
C	Limpieza de datos y análisis estadístico	5/10/21	15/10/21
D	Análisis de regresión de los posibles factores de riesgo	18/10/21	29/10/21
E	Aplicación de algoritmos de ML a los factores de riesgo	1/11/21	12/11/21
F	Aplicación de algoritmos de ML con imágenes de pacientes	15/11/21	26/11/21
G	Obtención de conclusiones y resultados	29/11/21	9/12/21
H	Preparación de la memoria y la presentación	10/12/21	21/1/22
I	PEC 0	15/9/21	22/9/21
J	PEC 1	23/9/21	4/10/21
K	PEC 2	5/10/21	8/11/21
L	PEC 3	9/11/21	9/12/21
M	PEC 4	10/12/21	24/12/21
N	PEC 5a	27/12/21	3/1/22
Ñ	PEC 5b	13/1/22	21/1/22

Figura 1: Listado de tareas realizadas en el proyecto.

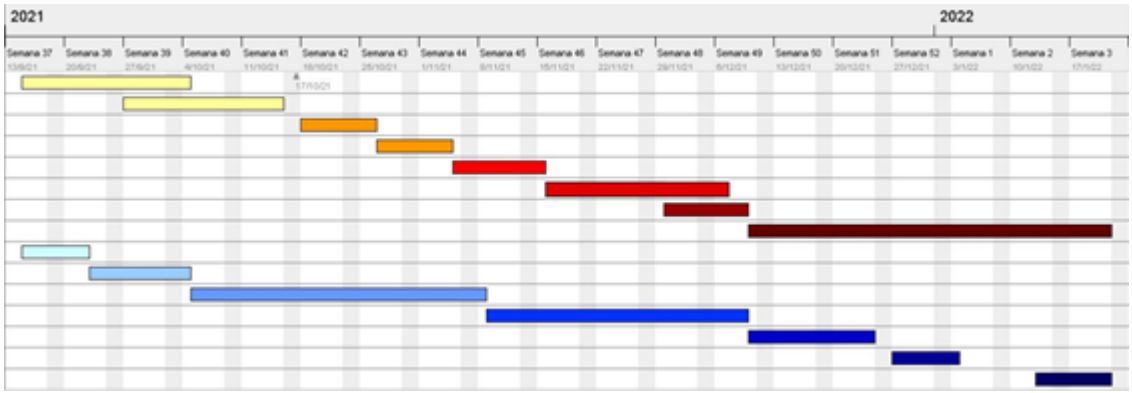


Figura 2: Diagrama de Grant con la planificación temporal del proyecto.

1.4.1.3. Hitos

En este proyecto, existen una serie de hitos que se corresponderían con las fechas límite de las diferentes actividades a entregar (PECs), dado que para cada una de ellas se precisa el cumplimiento de una serie de las tareas anteriormente establecidas. Por ello, estos serían:

- PEC 0 (22/09/21): En esta actividad debemos definir los contenidos de nuestro proyecto y realizar una propuesta de nuestro trabajo (título, palabras clave, temática, problemática a resolver, objetivos y bibliografía). Por ello, debemos de haber ya comenzado al menos el proceso de búsqueda de información y bases de datos.

- PEC 1 (04/10/21): Debemos realizar el plan de trabajo, es decir, como organizar las diferentes tareas a llevar a cabo para cumplir los objetivos del trabajo en el tiempo fijado. Debido a esto, debemos de terminar el proceso de búsqueda a la par o incluso antes para poder organizar el trabajo con conocimientos y con las bases de datos a utilizar ya decididas.

- PEC 2 (08/11/21): En esta actividad se nos pide la realización de un primer informe para el seguimiento del trabajo. Por ello, debemos de haber realizado en este punto ya la parte de la limpieza y análisis de los datos de factores de riesgo, el análisis de regresión y haber comenzado con la parte de predicción utilizando machine learning.

- PEC 3 (09/12/21): Al igual que en el caso anterior, se correspondería con un informe de seguimiento. En este caso, debemos ya de haber terminado tanto la parte de predicción de posibles enfermos a través de datos de pacientes con ML como la detección con ML a partir de imágenes de resonancia magnética (MRI). Por otro lado, en este punto ya debemos también de haber obtenido las conclusiones de los distintos resultados conseguidos durante el trabajo.

- PEC 4 (24/12/21): Esta entrega se corresponde con el cierre de la memoria, es decir, debemos de realizar la entrega del trabajo. Por ello, llegados a este punto debemos de haber finalizado completamente todas las tareas mencionadas en los anteriores hitos y haber redactado y revisado completamente la memoria.

- PEC 5a (03/01/22): Esta actividad abarca la preparación de la presentación y la entrega de la misma. Para ello, debemos utilizar como base toda la información obtenida en las anteriores tareas para de forma breve y concisa preparar la exposición oral de nuestro trabajo.

- PEC 5b (21/01/22): Finalmente, esta última actividad se correspondería con la defensa pública del trabajo y, por lo tanto, es clave tanto haber entregado la memoria como la presentación a realizar.

1.4.1.4. Análisis de riesgos

A priori, los riesgos del proyecto serían principalmente el tiempo y algún posible fallo informático tanto del software como de otras plataformas utilizadas. Aparte de estos dos puntos, otros factores que podrían detener el avance del proyecto y dificultar el seguimiento del plan de trabajo fijado pueden ser la descarga de grandes cantidades de datos necesarios, así como, la concesión del acceso a estos.

1.5. Breve resumen de las contribuciones

Los próximos capítulos del trabajo presentarán el contenido central del mismo donde trataremos varios puntos importantes. Inicialmente, haremos una introducción de la enfermedad del Alzheimer tratando su sintomatología, posibles causas, posibles factores de riesgo y posibles tratamientos.

Seguidamente, trataremos sobre el material y la metodología utilizada a lo largo del trabajo, es decir, los softwares, complementos y programas que hemos usado, así como, los datos y las fuentes de los mismos.

Por otro lado, llevaremos a cabo los diferentes análisis propuestos para este trabajo. Por un lado, realizaremos el análisis de regresión para conocer los posibles factores de riesgo de la enfermedad. Una vez conocidos estos factores, los utilizaremos para predecir los posibles enfermos mediante distintos algoritmos de machine learning conociendo cuál sería el mejor método para la predicción. Por último, llevaremos a cabo un análisis con una red neuronal convolucional para detectar posibles enfermos de Alzheimer mediante imágenes de resonancia magnética (MRI).

Capítulo 2

Enfermedad del Alzheimer

Actualmente, los avances tecnológicos y médicos están provocando un aumento de la esperanza de vida de los seres humanos, ya que cada vez las causas de muerte más comunes están siendo paliadas con tratamientos o con la prevención de las mismas. Esta debería de ser una muy buena noticia, pero existen ciertos inconvenientes derivados del aumento de la población envejecida, como son las enfermedades que afectan al correcto funcionamiento del cerebro y las demencias. Por ello, es tan importante el aumento de la esperanza de vida como el de la calidad de la misma, lo que implicaría una mejora en la detección temprana de estas patologías o nuevos posibles tratamientos para poder paliarlas.

Una de las enfermedades más conocidas de este tipo, y sobre la cual trata este trabajo, es el Alzheimer. Esta enfermedad fue descubierta en 1907 por Alois Alzheimer, el cual estudio el caso de una mujer que presentaba una deterioro de la memoria temprano (51 años) y la cual murió 4 años después. Esta patología se caracteriza por una pérdida de memoria progresiva y el deterioro de por lo menos otro dominio cognitivo, provocando una alteración en la función social o conductual. El dominio o dominios cognitivos, aparte de la memoria, que se ven afectados por la enfermedad suelen ser[1]:

- Pérdida de la capacidad de compresión de palabras (afasia).
- Pérdida de la capacidad de realizar tareas complejas que implican coordinación muscular (apraxia).
- Pérdida de la capacidad de reconocer y usar objetos familiares (agnosia).
- Pérdida de la capacidad de organizar, planificar y ejecutar actividades normales.

La incidencia de esta enfermedad es muy elevada, ya que representa el 70 % de los casos de demencia registrados y se estima que está presente en un 10 % de la población mundial de 65 años o más. Además, esta se duplica cada 5 o 10 años, es decir, en personas de edades de 65-69, 70-74, 75-79, 80-84 y 85 años o más, la incidencia se estima que es de 0.6 %, 1.0 %, 2.0 %, 3.3 % y 8.4 %[1].

La principal causa de esta enfermedad se cree que es el acúmulo extra e intracelular de una serie de proteínas que provocan el deterioro neuronal y sináptico a nivel regional. La proteína que se cree que se encuentra más relacionada con este proceso es la β-amiloide, la cual se acumula extracelularmente en forma de placas neuríticas, que alteran procesos neuronales, y oligómeros, que producen la alteración de la transducción de señales en cascada provocando cambios en las actividades neuronales y desencadenando la liberación de mediadores neurotóxicos por parte de la microglía (células inmunitarias). Este acúmulo puede estar generado por una sobreproducción de la misma, por el mal funcionamiento de sus enzimas de degradación o por alteraciones en los procesos de transporte a través de la barrera hematoencefálica. En estos procesos se encuentra implicada la proteína transportadora de lípidos apoE4, la cual en neuronas estresadas se escinde y produce productos neurotóxicos que también afectan a la neurona a nivel mitocondrial. A su vez, también podemos encontrar depósitos de β-amiloide en los vasos sanguíneos que generan alteraciones en la arteria (bajo suministro de nutrientes y metabolitos) y que pueden llegar a producir microinfartos corticales, microaneurismas y microhemorragias o macrohemorragias cerebrales. Estos agregados se cree que comienzan 20 años antes del desarrollo de los síntomas clínicos.

Por otro lado, también se produce la deposición intraneuronal de la proteína tau, que se encuentra vinculada a la unión de microtúbulos, en forma de ovillos neurofibrilares y de oligómeros patogénicos. Esto también ocurre con la proteína presináptica α-sinucleína, cuyos grandes agregados se denominan cuerpos de Lewy y también están presentes en la enfermedad de Parkinson. Estas estructuras de gran tamaño producen el desplazamiento de orgánulos importantes afectando a su correcto funcionamiento.[2][3]

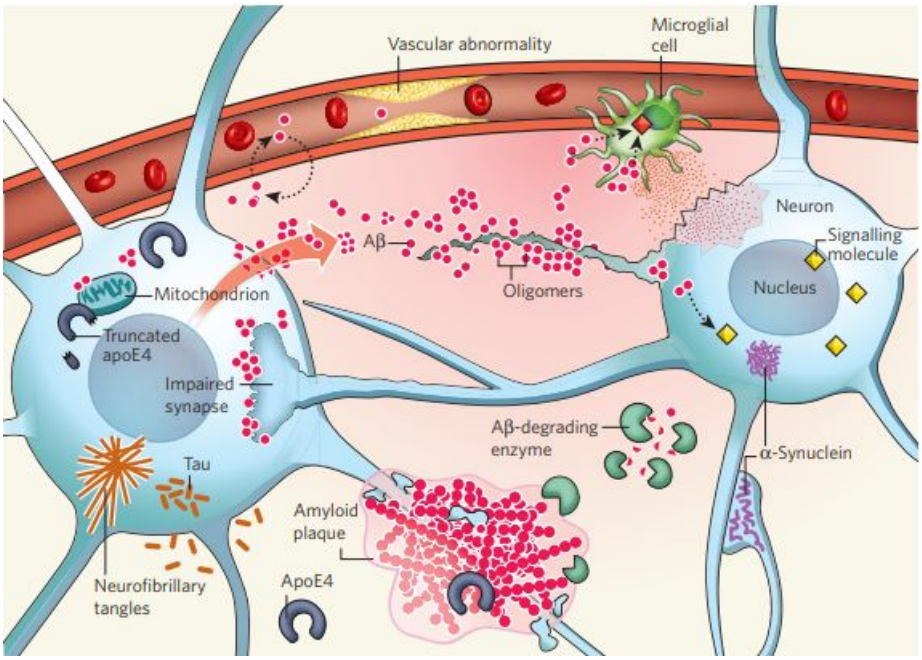


Figura 3: Acúmulos de diferentes proteínas en la enfermedad del Alzheimer[2]

Este punto se ha tenido en cuenta a la hora de buscar posibles tratamientos para la enfermedad y aunque sí que existen ciertos tratamientos paliativos que pueden frenar de algún modo los síntomas no existe ningún tratamiento definitivo para la EA. En la actualidad, los tratamientos aprobados más utilizados son[4]:

- Enzimas que provocan un aumento de neurotransmisores (acetilcolina, dopamina, serotonina, etc.), ya que finalmente todos estos procesos de destrucción neuronal progresiva generan una escasez de los mismos. Un ejemplo serían los inhibidores de acetilcolinesterasa (AChEI), como el donepezilo, galantamina y rivastigmina, que intenta reducir la degradación de acetilcolina manteniendo altos sus niveles en las sinapsis, retrasando así el deterioro cognitivo.
- Un antagonista del receptor de d-aspartato (NMDA), N-metilmemantina, que se une a este bloqueando el flujo de iones lo que provocaría una mejora de los niveles de glutamato que producen la disfunción neuronal.

A su vez, existen numerosos frentes de investigación para el desarrollo de nuevos posibles tratamientos más efectivos o que ayuden junto con estos a frenar en la medida de lo posible la enfermedad. La mayoría de estas líneas de investigación se encuentran orientadas hacia terapias modificadoras de la enfermedad (DMTs), que consisten en cambios que interfieren en los mecanismos fisiopatológicos de la EA. La gran mayoría de ellos se basan en frenar la sobreproducción de Aβ. Esta se produce por la sobreexpresión de su proteína precursora APP que mediante su escisión por la β-secretasa (BACE1) en la región extracelular y la γ -secretasa en la intracelular da lugar a esta proteína. Por otro lado, la α-secretasa produce la escisión de APP sin generar Aβ lo cual favorecería a evitar esta sobreproducción. Por ello, algunos de los DMTs anti-amiloides se centran principalmente en[4]:

- Reducción de la producción Aβ mediante inhibidores de la β-secretasa y γ-secretasa y potenciadores de la α-secretasa.
- Reducción de la formación de placas de Aβ mediante inhibidores de la agregación.
- Activación de la eliminación de Aβ mediante inmunoterapia activa y pasiva.

También existen DMTs basados en los mecanismos de la proteína Tau, similares a los anteriores, como son:

- Prevención de la formación de la proteína Tau mediante inhibidores quinasas implicadas en la hiperfosforilación de esta proteína.
- Inhibidores de la agregación de Tau.
- Estabilizadores de microtúbulos.
- Modificaciones postraducionales de Tau como la inhibición de la acetilación.

- Inmunoterapia activa y pasiva

Aunque estas son las dos líneas principales de estudio existen muchas otras basadas en otros factores de la enfermedad como el glutamato, agentes antiinflamatorios, factores de crecimiento, células madre y muchos otros.

Dado que como hemos dicho no existe un tratamiento definitivo y que los tratamientos paliativos existentes dependen en gran medida de ser suministrados en los primeros estadios de la enfermedad, podemos decir que la prevención y detención temprana de la misma juega un papel clave actualmente. A pesar de ello, no existe ningún tipo de prueba *ante mortem* que tenga una veracidad alta a la hora de diagnosticar la EA.

El diagnóstico es difícil debido a la alta similitud con otras patologías similares y con síntomas muy parecidos. A pesar de todo, el diagnóstico de la enfermedad suele realizarse con un conjunto de pruebas que abarcan:[5]

- Pruebas neuropsicológicas en las cuales se le realiza un cuestionario al paciente y a algún miembro de su familia de donde se obtendrá una puntuación que está regulada por una escala prefijada.
- Marcadores genéticos, es decir, una serie de genes que se ha visto hasta ahora que pueden estar implicados en el desarrollo de la enfermedad. Alguno de ellos son APP (proteína precursora de amiloide),PSEN1, PSEN2 y APOE 4 (alelo del gen de la apolipoproteína E).
- Marcadores de fluidos, de los cuales los más utilizados son los del líquido cefalorraquídeo (LCR). Dentro de estos los que mejores resultados han aportado han sido el Aß1-42 péptido (Aß42), la proteína tau total (T-tau) y la proteína tau fosforilada (P-tau).
- Pruebas electroencefalográficas (EEG) y magnetoencefalográficas (MEG) estudiando los ritmos de las señales cerebrales electromagnéticas a través del cuero cabelludo.
- Pruebas de neuroimagen, es decir, pruebas de imagen del cerebro del paciente mediante diferentes técnicas entre las cuales las que obtienen mejores resultados son la resonancia magnética (MRI) y la tomografía de emisión de positrones (PET).

En estas últimas, será en las que nos centraremos en este trabajo y las cuales utilizaremos para la detección de la enfermedad mediante algoritmos de machine learning.

Por último, como hemos visto la edad es el principal factor de riesgo de la EA, pero se ha descubierto que existen otros que también afectan en menor medida. A continuación, nos centraremos en conocer cuáles son estos factores que tienen gran influencia en la presencia de la enfermedad del Alzheimer.

Capítulo 3

Materiales y métodos

Para nuestro estudio, se han decidido utilizar una serie de bases de datos, además de diversos softwares y complementos que utilizaremos para realizar los diferentes análisis y predicciones que este comprende.

3.1. Datos utilizados

Los datos utilizados en la elaboración de este trabajo se obtuvieron del estudio de la iniciativa de neuroimagen de la enfermedad del Alzheimer (ADNI)[6]. ADNI fue creado en 2003 como una asociación público-privada, dirigida por el investigador principal Michael W. Weiner, MD. El objetivo principal de ADNI ha sido probar si la resonancia magnética (MRI), la tomografía por emisión de positrones (PET), otros marcadores biológicos y la evaluación clínica y neuropsicológica se pueden combinar para medir la progresión del deterioro cognitivo leve (DCL) y la enfermedad de Alzheimer temprana (EA).

Para el análisis de los factores de riesgo de la enfermedad del Alzheimer crearemos dos datasets a partir de varios conjuntos de esta base de datos:

- **Dataset 1**:
  Se encuentra compuesto por una versión reducida del conjunto de datos correspondiente al estudio comparativo ADNI-DIAN. Este presenta una serie de variables generales de los pacientes de las cuales utilizaremos únicamente las siguientes:
  - Variables respuesta:
    - 'CDRSUM': Suma de la clasificación clínica de la demencia (CDR)
    - 'cdrglob': Clasificación clínica de la demencia (CDR) (0,0.5,1,2,3)
    - 'GDS':Escala de deterioro global (0-15)
  - Variables explicativas:
    - 'visitage': edad del paciente en el momento del estudio
    - 'gender': género del paciente (1 = Hombre, 0 = Mujer)

- ‘EDUC‘: años de educación del participante
- ‘HISPANIC‘: paciente español o de etnia latina (1 = Yes, 0 = No, 9 = Desconocido)
- ‘RACE‘: raza del paciente (1 = Blanco, 2 = Negro o afroamericano, 3 = Indio americano o nativo de Alaska, 4 = Nativo hawaiano u otro isleño del Pacífico, 5 = Asiático, 50 = Otro)
- ‘PRIMLANG‘: primera lengua del paciente (1 = Inglés, 2 = Español, 3 = Mandarín, 4 = Cantonés, -5 = Ruso, 6 = Japonés, 8 = Otra, 9 = Desconocido)
- ‘MARISTAT‘: estado civil o marital del paciente (1 = Casado, 2 = Viudo, 3 = Divorciado, 4 = Separado, 5 = Nunca se ha casado, 6 = Viviendo como casado, 8 = Otro, 9 = Desconocido)
- ‘HANDED‘: mano dominante del paciente (1 = Zurdo, 2= Diestro, 3 = Ambidiestro, 9 = Desconocido)
- ‘MOMDEM‘: madre con demencia (1 = Yes, 0 = No, 9 = Desconocido)
- ‘DADDEM‘: padre con demencia (1 = Yes, 0 = No, 9 = Desconocido)

- **Dataset 2**:
  Este segundo conjunto de datos se encuentra formado por la combinación de varios que recogen información de los pacientes sobre sus signos vitales y su historial médico junto con su diagnóstico con la escala CDR. De todas las variables de los tres conjuntos de datos a partir de los cuales se forma este dataset se han escogido las siguientes:

  - Variable respuesta:
    - ‘CDGLOBAL‘: Clasificación clínica de la demencia (CDR) (0,0.5,1,2,3)
  - Variables explicativas:
    - ‘VSWEIGHT‘: peso del paciente (kg)
    - ‘VSHEIGHT‘: altura del paciente (cm)
    - ‘VSBPSYS‘: presión arterial sistólica del paciente
    - ‘VSBPDIA‘: presión arterial diastólica del paciente
    - ‘VSPULSE‘: pulso del paciente
    - ‘VSRESP‘: respiraciones por minuto del paciente
    - ‘VSTEMP‘: temperatura del paciente
    - ‘MHPSYCH‘: historial médico del paciente en psicología
    - ‘MH2NEURL‘: historial médico del paciente en neurología
    - ‘MH3HEAD‘: historial médico del paciente en cabeza,ojos,oídos y nariz
    - ‘MH4CARD‘: historial médico del paciente en cardiovascular
    - ‘MH5RESP‘: historial médico del paciente en respiratorio
    - ‘MH6HEPAT‘: historial médico del paciente en hepático
    - ‘MH7DERM‘: historial médico del paciente en dermatología

- ‘MH8MUSCL‘: historial médico del paciente en muscular
- ‘MH9ENDO‘: historial médico del paciente en endocrino
- ‘MH10GAST‘: historial médico del paciente en gástrico
- ‘MH11HEMA‘: historial médico del paciente en hematología
- ‘MH12RENA‘: historial médico del paciente en renal
- ‘MH13ALLE‘: historial médico del paciente en alergias o reacción a medicamentos
- ‘MH14ALCH‘: consumo de alcohol
- ‘MH15DRUG‘: consumo de drogas
- ‘MH16SMOK‘: consumo de tabaco

Todos los conjuntos de datos descargados de las bases de ADNI han sido obtenidos en formato csv junto a un archivo de texto que contiene un diccionario con datos explicativos de las variables que los componen.

En el caso de las imágenes para la detección mediante ML, también serán obtenidas a partir de la base de datos de ADNI, en concreto, de sus colecciones de imágenes. Para este estudio utilizaremos imágenes del cerebro de pacientes procedentes de pruebas de resonancia magnética (MRI) en plano axial, con peso TS2 y con un grosor de 3.0 mm. Estas MRI, en concreto, serán obtenidas utilizándola técnica de obtención de imágenes denominada double turbo spin echo (Double TSE). Para que todas las imágenes se correspondan con aproximadamente la misma sección del cerebro, dependiendo de la forma y tamaño del mismo, utilizaremos el corte número 48 que se encuentra cercano a la mitad de este.

Obtendremos tres grupos de imágenes según el diagnóstico cognitivo del paciente en cuestión que serán CN (cognitivamente normal), MCI (deterioro cognitivo leve) y AD (con enfermedad del Alzheimer). Cada grupo estará compuesto por 99 imágenes de las cuales un 30 % de cada uno serán seleccionadas para formar el grupo de prueba (test) y el resto compondrán el grupo de entrenamiento (train).

La descarga de las imágenes se hará en formato DCM, pero serán transformadas posteriormente a forma JPG para poder ser utilizadas correctamente por el sofware informático que llevará a cabo la clasificación de las mismas.

3.2. Software y complementos

Para el tratado de datos y su preparación, así como, para llevar a cabo cada uno de los análisis y predicciones que contiene este proyecto se ha utilizado el software R[7] junto con el entorno de desarrollo R Studio[8] y una serie de paquetes que se irán describiendo a continuación.
Para el análisis de factores de riesgo de la enfermedad, no necesitaremos ningún paquete a mayores debido a que las funciones que necesitamos en este caso ya vienen integradas de serie

en la biblioteca básica de R.

Por otro lado, en el caso de las predicciones, requeriremos una serie de funciones para aplicar los algoritmos de machine learning que vendrán integradas en paquetes que debemos instalar adicionalmente:

- *caret*[9]: este paquete contiene la función *confusionMatrix()* que utilizaremos con los resultados de todas las predicciones para conocer su eficiencia mediante una matriz de confusión, sabiendo así el porcentaje de datos clasificados correctamente.
- *gmodels*[10]: contiene la función *CrossTable()* que utilizaremos en algunos algoritmos para conocer como se han clasificado los datos predichos por el algoritmo.
- *class*[11]: presenta la función *knn()* que utilizaremos para la aplicación del algoritmo K-Nearest-Neighbor.
- *neuralnet*[12]: incluye la función *neuralnet()* para la aplicación del algoritmo de Red Neuronal Artificial.
- *e1071*[13]: comprende la función *naiveBayes()* que utilizaremos para aplicar el algoritmo Naive Bayes.
- *kernlab*[14]: contiene la función *ksvm()* para la aplicación del algoritmo Support Vector Machine (SVM).
- *c50*[15]: incluye la función *c5.0()* que usaremos para aplicar el algoritmo de Árbol de Decisión.
- *randomForest*[16]: presenta la función *randomForest()* para la aplicación del algoritmo Random Forest.

Por último, para la detección de posibles enfermos mediante imágenes de resonancia magnética de los pacientes utilizaremos TensorFlow[17] en un entorno de Python (versión 3.10.1)[18], ya que es la manera más eficiente y clara para llevar a cabo este tipo de predicciones. En concreto, utilizaremos una serie paquetes o librerías para crear una red neuronal convolucional (CNN) que sea capaz de clasificar correctamente imágenes procedentes de esta prueba diagnóstica. Las librerías adicionales utilizadas para el proyecto son las siguientes:

- *SciPy*[19]: esta librería contiene la función random que utilizaremos para fijar la semilla de la aleatoriedad para así utilizar siempre los mismos datos escogidos al azar.
- *NumPy*[20]: es una librería fundamental de Python que contiene gran cantidad de funciones útiles.
- *OpenCV (cv2)*[21]: es una librería de manipulación de imágenes de la que utilizaremos funciones para la lectura de estas.

- *PIL*[22]: esta librería contiene funciones para el tratado de las imágenes.
- *Scikit-learn (sklearn)*[23]: es una librería de aprendizaje automático con machine learning de la cual utilizaremos la función *train_test_split* que dividirá nuestro conjunto de datos en dos grupos, uno de entrenamiento (train) para el modelo y otro de prueba (test) para evaluarlo.
- *Keras*[24]: será la librería que más utilizaremos debido a que contiene todas las funciones que compondrán el modelo. Además, también utilizaremos *normalize* y*to_categorical* para normalizar los datos y trasformar variables en categóricas, respectivamente.
- *IPython*[25]: contiene la función *SVG* con la cual crearemos el esquema de nuestro modelo.

Además de las funciones contenidas en estas librerías adicionales, a lo largo del estudio también se utilizarán otras funciones que pertenecen a la librería de referencia de Python, que se encuentra ya instalada de serie[26].

Capítulo 4

Análisis de posibles factores de riesgo

En este apartado de nuestro estudio, trataremos de conocer que factores son los que influyen en mayor medida sobre la enfermedad del Alzheimer. En estos últimos años, este tipo de estudios que utilizan las nuevas tecnologías y recursos informáticos están siendo punteros en esta y otras enfermedades que no presentan una cura o tratamiento definitivo. En el caso de la EA, podemos observar a simple vista que la edad es la principal característica de los pacientes que la sufren, siendo muy extraños o inexistentes casos de este tipo de demencia en personas con una edad inferior a 50 años. Lo que queremos esclarecer en este proyecto es si este el único factor que de verdad influye sobre la presencia de esta patología o existen algún otro sobre el que debamos poner el foco de atención también.

Para ello, analizaremos una serie de variables en relación con la escala del grado de demencia clínica (CDR)[27] de los pacientes. Esta escala, es una forma de medir el grado de demencia de las personas y fue creada en 1982 por Hughes y modificada posteriormente por Morris en 1992[28]. Esta fue considerada un método adecuado de diagnístico neuropsicológico de la EA y contiene 5 niveles o grados de demencia (desde el nivel 0 o sin demencia al nivel 3 o demencia grave).

Esta clasificación se basa en dos entrevistas, una realizada al paciente en cuestión y otra realizada a un informante que normalmente es un familiar directo de este (conyugue o pareja). Las preguntas que componen estas entrevistas son de caracter psicológico en relación con tareas cognitivas que implican la memoria y la orientación. Por ello requerimos una persona cercana al posible enfermo, ya que si este de verdad sufre esta patología podría verse afectada la veracidad de sus respuestas durante la entrevista. Los resultados tras clasificar al paciente según su nivel de demencia pueden ser los siguientes:

Valoración Clínica de Demencia© (Spanish version of CDR)

Valoración Clínica de Demencia (CDR)	0	0.5	1	2	3

	Deterioro				
	Ninguno 0	Dudoso 0.5	Leve 1	Moderado 2	Grave 3
Memoria	Sin pérdida de memoria o leves olvidos inconstantes	Olvidos leves constantes; recolección parcial de eventos; olvidos "benignos"	Pérdida moderada de memoria; más marcada para eventos recientes; el defecto interfiere con las actividades diarias	Pérdida grave de memoria; sólo retiene materias con mucho aprendizaje; materias nuevas se pierden con rapidez	Pérdida grave de memoria; sólo retiene fragmentos
Orientación	Completamente orientado	Completamente orientado pero con leve dificultad para las relaciones temporales	Dificultad moderada con las relaciones temporales; orientado en el lugar del examen; puede tener algo de desorientación geográfica en otro lugar	Dificultad grave con las relaciones temporales; habitualmente desorientado en el tiempo; a menudo en el lugar	Orientado sólo en persona
Razonamiento y solución de problemas	Resuelve los problemas diarios y se encarga bien de los negocios y finanzas; razonamiento bueno con relación al comportamiento previo	Leve dificultad para resolver problemas, similitudes y diferencias	Dificultad moderada para hacer frente a problemas, similitudes y diferencias; razonamiento social habitual mantenido	Grandes dificultades para hacer frente a problemas, similitudes y diferencias; juicio social habitual limitado	Incapaz de razonar o resolver problemas
Actividades fuera de casa	Función independiente a su nivel habitual en el trabajo, compras, voluntariado y agrupaciones sociales	Leve dificultad en estas actividades	Incapaz de ser independiente en estas actividades, aunque aún puede participar en alguna; parece normal a primera vista	Incapaz de ser independiente fuera de casa Parece estar lo suficientemente bien como para realizar funciones fuera de casa	Parece demasiado enfermo/a como para realizar funciones fuera de su casa
Actividades domésticas y aficiones	Vida en casa, aficiones e intereses intelectuales bien conservados	Vida en casa, aficiones e intereses intelectuales algo limitados	Dificultad leve pero clara de su actividad doméstica; abandono de las tareas más difíciles; abandono de las aficiones e intereses más complicados	Sólo realiza tareas simples; intereses muy restringidos y mal mantenidos	Sin función significativa en casa
Cuidado personal	Completamente capaz de cuidarse por sí mismo/a		Necesita recordatorios	Requiere ayuda para vestirse, asearse y encargarse de sus efectos personales	Requiere mucha ayuda para su cuidado personal; incontinencia frecuente

Puntúe sólo cuando disminuya del nivel previo habitual debido a pérdida cognitiva, no a limitaciones debidas a otros factores.

Figura 4: Tabla de puntuaciones de la escala del grado de demencia clínica (CDR)[29]

En el caso del análisis del primer conjunto de datos utilizado, se realiza un modelo de regresión múltiple en el que la variable CDR (*CDRglob*) se corresponderá con la variable respuesta y el resto de variables, que contienen información general del paciente, serán las variables explicativas. Realizando un resumen estadístico de este modelo obtendremos que variables se consideran significativas para la predicción de la variable respuesta mediante el valor p de cada una de ellas.

En el caso de este dataset, también realizaremos dos modelos adicionales con la suma de las puntuaciones de CDR (*CDRSUM*) y con la escala de deterioro global (*GDS*) de Reisberg[30] (7 posibles categorías de deterioro cognitivo), con el fin de obtener una información más detallada de cuáles pueden ser los posibles factores de riesgo según estas variantes que también definen a la enfermedad.

Estadio	Fase clínica	Características FAST	Comentarios
GDS 1. Ausencia de déficit cognitivo	Normal MEC: 30-35	Ausencia de déficit funcionales objetivos o subjetivos.	No hay deterioro cognitivo subjetivo ni objetivo
GDS 2. Déficit cognitivo muy leve	Normal para su edad. Olvido MEC: 25-30	Déficit funcional subjetivo	Quejas de perdida de memoria en ubicación de objetos, nombres de personas, citas, etc. No se objetiva déficit en el examen clínico ni en su medio laboral o situaciones sociales. Hay pleno conocimiento y valoración de la sintomatología.
GDS 3. Déficit cognitivo leve	Deterioro límite MEC: 20-27	Déficit en tareas ocupacionales y sociales complejas y que generalmente lo observan familiares y amigos	Primeros defectos claros. Manifestación en una o más de estas áreas: • Haberse perdido en un lugar no familiar • Evidencia de rendimiento laboral pobre • Dificultad para recordar palabras y nombres • tras la lectura retiene escaso material • olvida la ubicación, pierde o coloca erróneamente objetos de valor • escasa capacidad para recordar a personas nuevas que ha conocido El déficit de concentración es evidente para el clínico en una entrevista exhaustiva. La negación como mecanismo de defensa ,o el desconocimiento de los defectos, empieza a manifestarse. Los síntomas se acompañan de ansiedad leve moderada
GDS 4. Déficit cognitivo moderado	Enfermedad de Alzheimer leve MEC: 16-23	Déficits observables en tareas complejas como el control de los aspectos económicos personales o planificación de comidas cuando hay invitados	Defectos manifiestos en: • olvido de hechos cotidianos o recientes • déficit en el recuerdo de su historia personal • dificultad de concentración evidente en operaciones de resta de 7 en 7. • incapacidad para planificar viajes, finanzas o actividades complejas Frecuentemente no hay defectos en: • orientación en tiempo y persona • reconocimiento de caras y personas familiares • capacidad de viajar a lugares conocidos Labilidad afectiva Mecanismo de negación domina el cuadro
GDS 5. Déficit cognitivo moderadamente grave	Enfermedad de Alzheimer moderada MEC: 10-19	Decremento de la habilidad en escoger la ropa adecuada en cada estación del año o según las ocasiones	Necesita asistencia en determinadas tareas, no en el aseo ni en la comida, pero sí para elegir su ropa Es incapaz de recordar aspectos importantes de su vida cotidiana (dirección, teléfono, nombres de familiares) Es frecuente cierta desorientación en tiempo o en lugar Dificultad para contar en orden inverso desde 40 de 4 en 4, o desde 20 de 2 en 2 Sabe su nombre y generalmente el de su esposa e hijos
GDS 6. Déficit cognitivo grave	Enfermedad de Alzheimer moderadamente grave MEC: 0-12	Decremento en la habilidad para vestirse, bañarse y lavarse; específicamente, pueden identificarse 5 subestadios siguientes: **a)** disminución de la habilidad de vestirse solo **b)** disminución de la habilidad para bañarse solo **c)** disminución de la habilidad para lavarse y arreglarse solo **d)** disminución de la continencia urinaria **e)**disminución de la continencia fecal	Olvida a veces el nombre de su esposa de quien depende para vivir Retiene algunos datos del pasado Desorientación temporo espacial Dificultad para contar de 10 en 10 en orden inverso o directo Puede necesitar asistencia para actividades de la vida diaria Puede presentar incontinencia Recuerda su nombre y diferencia los familiares de los desconocidos Ritmo diurno frecuentemente alterado Presenta cambios de la personalidad y la afectividad (delirio, síntomas obsesivos, ansiedad, agitación o agresividad y abulia cognoscitiva)
GDS 7. Déficit cognitivo muy grave	Enfermedad de Alzheimer grave MEC: 0	Pérdida del habla y la capacidad motora Se especifican 6 subestadios: **a)** capacidad de habla limitada aproximadamente a 6 palabras **b)** capacidad de habla limitada a una única palabra **c)** pérdida de la capacidad para caminar solo sin ayuda **d)** pérdida de la capacidad para sentarse y levantarse sin ayuda **e)** pérdida de la capacidad para sonreir **f)** pérdida de la capacidad para mantener la cabeza erguida	Perdida progresiva de todas las capacidades verbales Incontinencia urinaria Necesidad de asistencia a la higiene personal y alimentación Pérdida de funciones psicomotoras como la deambulación Con frecuencia se observan signos neurológicos

Figura 5: Tabla de estadios en la escala de deterioro global de Reisberg (GDS)[31]

Por otro lado, en el caso del segundo conjunto de datos analizado, se realizará también un modelo de regresión con las puntuaciones de la escala CDR (*CDGLOBAL*) como variable respuesta pero en este caso el resto de variables, que se corresponde con las variables explicativas del modelo, contienen información sobre los signos vitales e historial médico de los pacientes.

En ambos casos, realizaremos finalmente un análisis de varianza (ANOVA) de los modelos con el fin de confirmar la significancia de las variables obtenidas mediante el estadístico F.

Capítulo 5

Predicción con machine learning a partir de los factores de riesgo

En esta segunda parte del estudio, se llevará a cabo la división de nuestros conjuntos de datos en dos grupos, un grupo de entrenamiento y un grupo de prueba, que utilizaremos para aplicar una serie de algoritmos de machine learning realizando la predicción de posibles enfermos de EA. En ambos casos, utilizaremos únicamente las variables que han salido como significativas (factores de riesgo) en el apartado anterior que son las que mejor definen la variable respuesta (CDR).

Los algoritmos de machine learning que utilizaremos son los siguientes[32]:

- **K vecinos más cercanos (k-NN)**: recibe este nombre porque utiliza información sobre los k vecinos más cercanos de un ejemplo para clasificar ejemplos sin etiquetar. La letra k es un término variable que implica que se puede utilizar cualquier número de vecinos más cercanos. En nuestro caso, utilizaremos varios valores de k para tratar de obtener el mejor resultado posible en la predicción. Entonces para cada registro sin etiquetar en el conjunto de datos de prueba, k-NN identifica k registros en los datos de entrenamiento que son los "más cercanos" en similitud. A los datos de prueba sin etiqueta se le asigna entonces la clase de la mayoría de estos vecinos más cercanos

- **Naive Bayes**: es un algoritmo basado en los métodos bayesianos que utiliza los datos de entrenamiento para calcular la probabilidad observada de cada resultado en función de los valores de las características que luego usa para predecir la clase más probable para las nuevas características en datos no etiquetados.

- **Red Neuronal Artificial**: se trata de un tipo de algoritmo basado en las neuronas biológicas del cerebro animal que a partir de un conjunto de señales de entrada es capaz de generar señales de salida. Al igual que nuestro cerebro, este utiliza una red de neuronas o nodos artificiales que crean un procesador paralelo masivo capaz de resolver problemas de aprendizaje.

- **Support Vector Machine (SVM)**: es un tipo de algoritmo que se basa en la creación de un hiperplano en el espacio multidimensional, resultante de las distintas características,

para separar los datos de forma homogénea y con el mayor margen posible. Por lo tanto, este combina el modelo de regresión lineal con el aprendizaje mediante los vecinos más cercanos permitiendo modelar relaciones muy complejas.

- **Árbol de Decisión**: este algoritmo utiliza una estructura de árbol para modelar las relaciones entre las características y los resultados potenciales y que sigue una estructura de decisión ramificada (nodos de decisión) para la predicción final.
- **Random Forest**: consiste en la generación de un conjunto de árboles de decisión (bosque) con selección aleatoria de características cuyas predicciones se combinarán posteriormente.

Tras aplicar cada uno de ellos a los dos conjuntos de datos, estudiaremos cuál es el nivel de precisión o exactitud de estos mediante la matriz de confusión creada a partir de las etiquetas reales de diagnóstico de los pacientes pertenecientes al conjunto de prueba y las predicciones obtenidas por los distintos algoritmos.

De esta forma, podremos comparar su eficiencia y decidiremos cuál es el mejor método para prevenir la enfermedad mediante el estudio de los factores de riesgo de los pacientes. Esto podría ayudar a realizar una detección temprana que nos permitiría aplicar los diferentes tratamientos existentes con el fin de ralentizar la enfermedad y la aparición de sus síntomas.

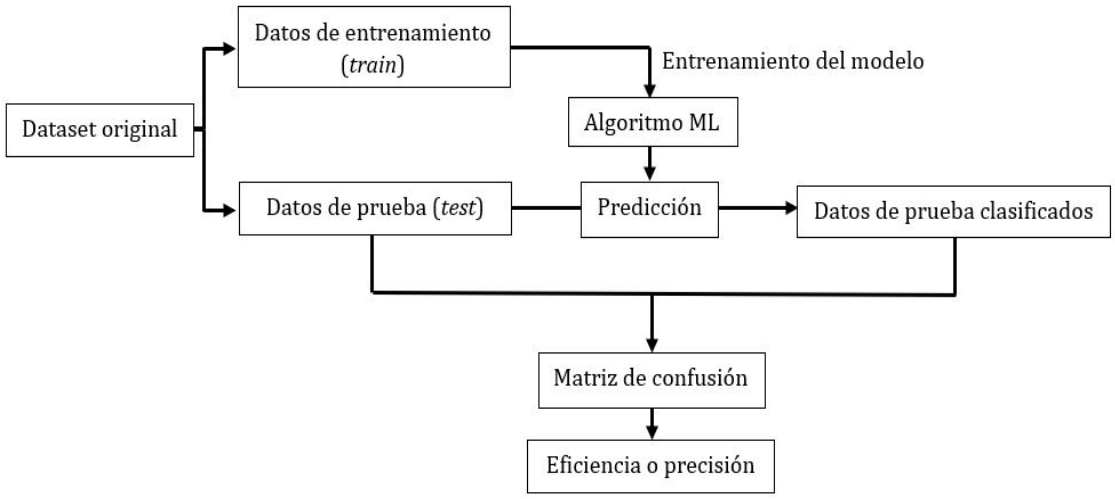


Figura 6: Esquema del proceso de predicción

Capítulo 6

Detección con machine learning a partir de imágenes

La enfermedad del Alzheimer es una enfermedad difícil de detectar dado que sus síntomas son muy similares a los de otras patologías y las pruebas diagnósticas que se utilizan hoy en día no son del todo eficientes. Por ello, para el diagnóstico de EA actualmente se requiere un conjunto de pruebas, que abarcan pruebas neuropsicológicas, de neuroimagen, marcadores de genéticos y de fluidos, así como, pruebas electro y magnetoencefalográficas. Tener que realizar varias pruebas para corroborar la presencia de la enfermedad hace de este un proceso tedioso y costoso además de duradero, siendo el tiempo muy importante para tratar de frenar la enfermedad.

De este modo, con el fin de mejorar la eficiencia del diagnóstico de esta enfermedad trataremos de crear un modelo algorítmico de aprendizaje profundo (*deep learning*) que sea capaz de clasificar imágenes obtenidas del principal tipo de prueba de neuroimagen utilizado hoy en día, es decir, la resonancia magnética (MRI).

Esta clasificación se basará principalmente en las diferencias existentes entre las imágenes de tres grupos que se corresponden con los posibles diagnósticos de la enfermedad: CN (estado cognitivo normal), MCI (deterioro cognitivo leve) y AD (enfermo de Alzheimer). Las principales diferencias existentes entre los cerebros de pacientes que padecen EA o demencia y los que no es la reducción de la materia gris (GM), lo que se asocia con la disfunción cognitiva y la sintomatología de esta enfermedad. Esta reducción de masa cerebral se cree que se encuentra relacionada con la atrofia cerebral provocada por esta patología[33].

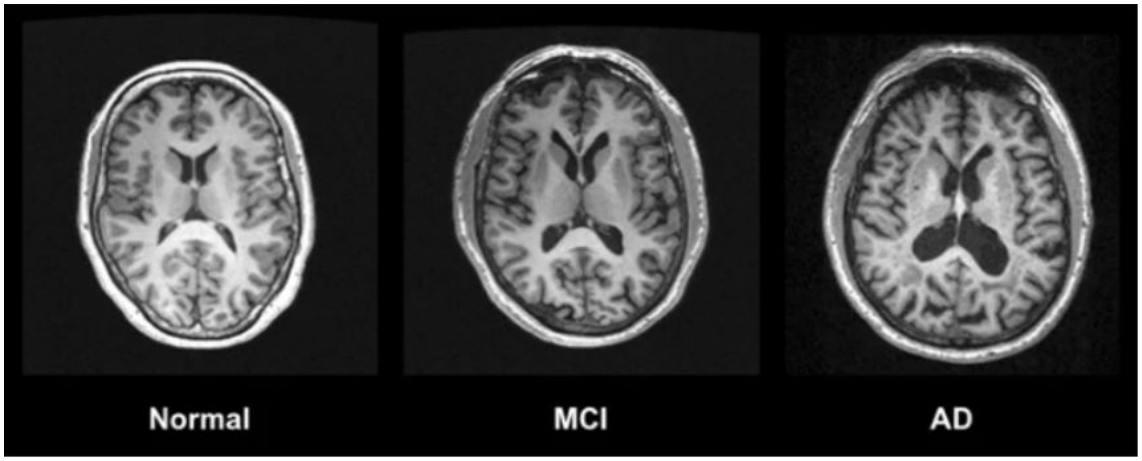


Figura 7: Diferencias entre imágenes de resonancia magnética (MRI) en pacientes con estados cognitivos distintos (CN, MCI y AD).[33]

Para ello, crearemos un modelo de red neuronal convolucional (CNN) que consiste en un tipo de red neuronal artificial con aprendizaje supervisado diseñada para procesar datos que vienen en forma de múltiples matrices como las imágenes. Este tipo de modelos tratan de simular el córtex visual del ojo humano y presentan una estructura de varias etapas en las que intervienen varios tipos de capas: capas convolucionales, capas no lineales y capas de agrupación o "pooling". Las capas convolucionales están organizadas en mapas de características, resultado de convoluciones entre parches locales (kernels) y vectores de ponderación llamados filtros. Su función es detectar patrones de características en la capa anterior. Por otro lado, las capas no lineales aumentan las propiedades no lineales de los mapas de características. Finalmente, la función de la capa de agrupación es fusionar características semánticamente similares en una.[34][35]

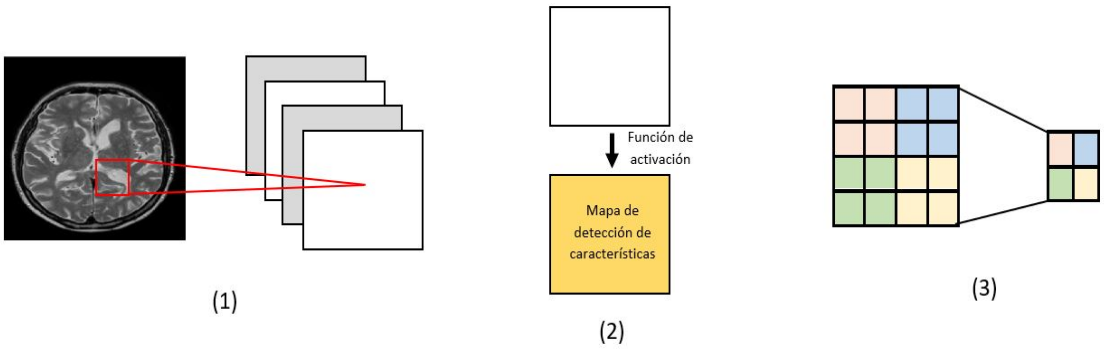


Figura 8: Procesos básicos de una red neuronal convolucional (CNN): (1) Convolución con la aplicación de los kernels en las diferentes regiones de la imagen (input); (2) Función de activación (normalmente ReLu) para obtener el mapa de detección de características; y (3) Subsampling o submuestreo mediante la función de agrupación (pooling) que fusiona características similares.

En nuestro caso, utilizaremos varias veces estos tres tipos de capas (proceso de convolución) consecutivamente, ya que cuanto más lo hagamos mayor será la capacidad de los mapas de características para reconocer formas complejas. En concreto, utilizaremos cuatro capas de convolución, ya que tampoco debemos utilizar demasiadas debido a que requiere un mayor procesamiento.

Seguidamente, se llevará a cabo el aplanamiento mediante la función *flatten* para transformar la salida tridimensional de las convoluciones en una capa de neuronas a la que se le aplicará la función *softmax* que dará lugar a la capa de salida final que contendrá tantas neuronas como clases de imágenes estemos clasificando, en nuestro caso, tres.

Finalmente, una vez creado nuestro modelo de clasificación lo que haremos será entrenarlo con el grupo de datos de entrenamiento (train) que mediante *backpropagation* irá mejorando los pesos de los diferentes conexiones entre capas hasta llegar al valor óptimo. Una vez entrenado nuestro modelo, lo validaremos con el conjunto de datos de prueba (test), es decir, comprobaremos la eficiencia clasificando imágenes de este obteniendo el dato de precisión del mismo.

Capítulo 7

Resultados y discusión

A continuación, analizaremos los resultados de los diferentes análisis realizados y trataremos de comprenderlos para posteriormente obtener las conclusiones de nuestro estudio.

7.1. Análisis de los posibles factores de riesgo

Como ya hemos dicho anteriormente, en el primer conjunto de datos se han utilizado 3 variables respuesta por separado cuyos resultados, a nivel de variables explicativas significativas son los siguientes:

Variable respuesta del modelo de regresión	**Variables explicativas significativas**
CDRSUM	*visitage, HISPANIC, MARISTAT, DADDEM*
GDS	*gender, HANDED*
cdrglob	*EDUC, MARISTAT, DADDEM*

Cuadro 1: Tabla resumen de las variables significativas obtenidas en el análisis del primer conjunto de datos

Como podemos observar, las variables significativas y que, por lo tanto, consideraremos factores de riesgo de la EA, son la edad, el pertenecer o no a la etnia latina o hispana, el estado civil o marital, antecedentes de esta enfermedad en el padre del paciente, el género, la mano dominante y los años de educación.

Por otro lado, en el segundo conjunto de datos, en el que se estudiaron variables relacionadas con los signos vitales y el historial médico del paciente, hemos obtenido que las variables significativas son las siguientes:

Variable respuesta del modelo de regresión	Variables explicativas significativas
cdrglob	*VSHEIGHT, VSBPDIA, MH3HEAD, MH5RESP, MH13ALLE, MH14ALCH*

Cuadro 2: Tabla resumen de las variables significativas obtenidas en el análisis del segundo conjunto de datos

En este caso, estas variables se corresponden con la altura del paciente, la presión diastólica y antecedentes en el historial médico de cabeza, nariz, ojos u oídos, respiratorios, alergias o reacciones a medicamentos y consumo de alcohol.

7.2. Predicción con machine learning

Tras haber realizado las predicciones de los datos de prueba (*test*) con los distintos algoritmos y contrastado estas con las etiquetas reales de los mismos mediante una matriz de confusión los resultados de precisión obtenidos para el primer conjunto de datos son los siguientes:

Algoritmo utilizado	Precisión (Accuracy)
k-NN	0.6545
Red Neuronal Artificial	0.6481
Naive Bayes	0.0299
SVM	0.6503
Árbol de decisión	0.6406
Random Forest	0.6492

Cuadro 3: Tabla de las precisiones obtenidas en cada algoritmo utilizado para la predicción de posibles pacientes enfermos de primer conjunto de datos (características generales de los pacientes) (en rojo el valor más alto).

En el caso del segundo conjunto de datos, los resultados de precisión obtenidos por los algoritmos serían:

Algoritmo utilizado	Precisión (Accuracy)
k-NN	0.5718
Red Neuronal Artificial	0.5559
Naive Bayes	0.2819
SVM	0.5691
Árbol de decisión	0.5559
Random Forest	0.5532

Cuadro 4: Tabla de las precisiones obtenidas en cada algoritmo utilizado para la predicción de posibles pacientes enfermos del segundo conjunto de datos (signos vitales e historial médico) (en rojo el valor más alto).

En ambos casos, vemos que las precisiones más altas se obtiene con los algoritmos de k vecinos más cercanos (k-NN) y de Support Vector Machine (SVM). Por otro lado, coinciden también en que las peores predicciones fueron obtenidas por el algoritmo Naive Bayes, las cuales distan bastante de las del resto de algoritmos que son muy similares.

Para el algoritmo k-NN en los dos datasets hemos probado diferentes valores de k en busca del que aporte un mejor resultado a la predicción:

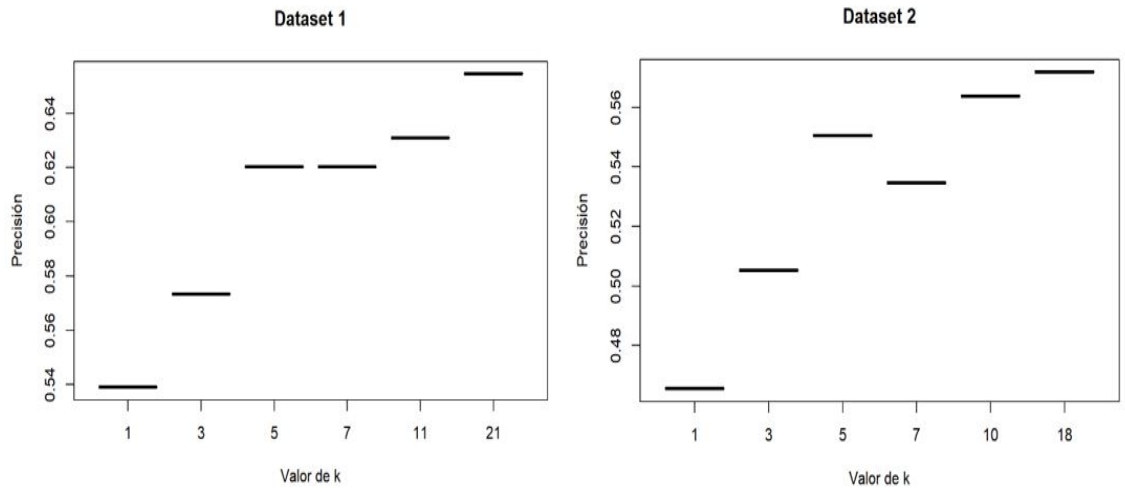


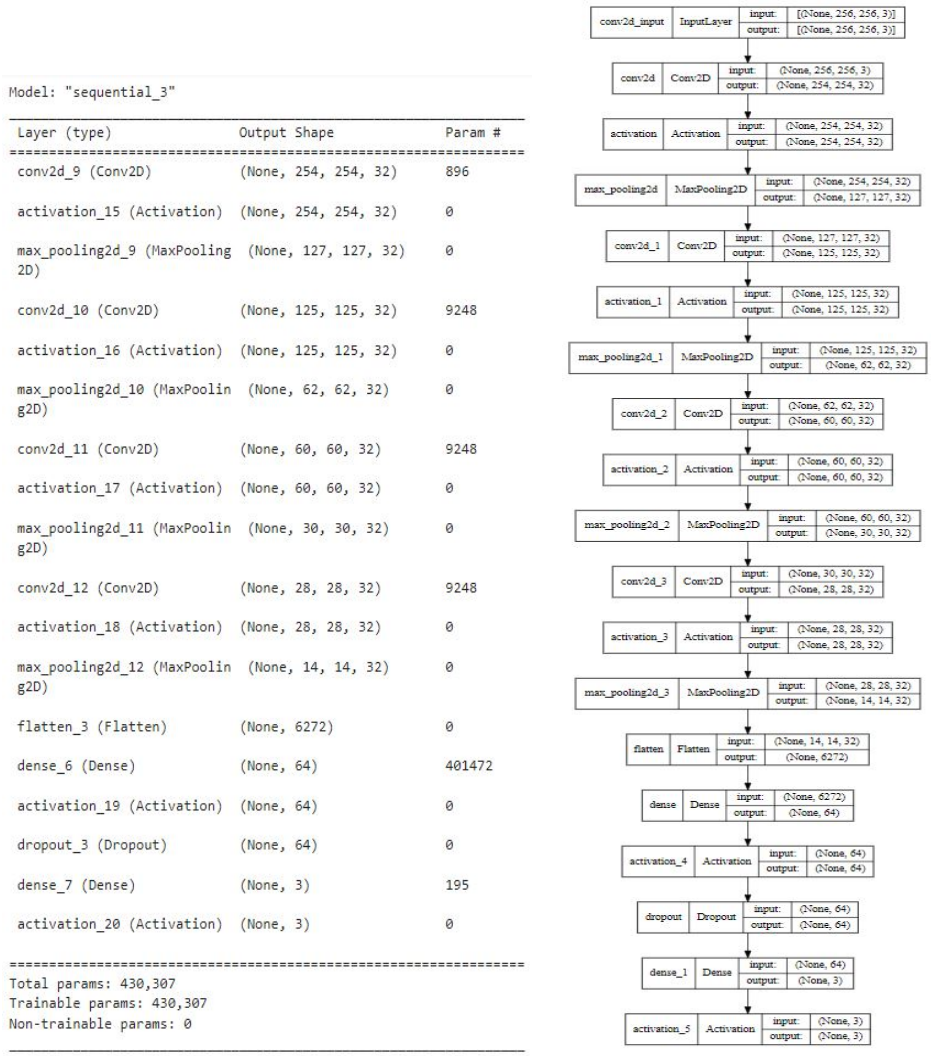
Figura 9: Comparación de la precisión obtenida según los diferentes valores de k en los dos conjuntos de datos (datase 1 y dataset 2).

En el estudio del primer dataset los niveles de precisión oscilan aproximadamente sobre 0.65, es decir, que un 65 % de los pacientes fueron clasificados correctamente. Este valor es bueno pero no lo suficiente para un estudio de este tipo, ya que la prevención utilizando este método no sería del todo fiable. A pesar de todo, si podríamos utilizar este mecanismo para complementarlo con otro tipo de pruebas y conseguir así un mejor resultado.

En cambio, en el segundo dataset apreciamos que los valores de precisión en general aún son más bajos, alrededor de 0.55, lo que implica que poco más de la mitad de los pacientes fueron clasificados correctamente. En este caso, sí que la utilidad de la prevención mediante este tipo de algoritmos se ve mermada y no aportaría información relevante y veraz a un estudio real.

7.3. Detección con machine learning a partir de imágenes

Los resultados del modelo de red neuronal convolucional (CNN), con sus distintos componentes mencionados anteriormente, presentan la siguiente estructura:



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 254, 254, 32)      896

 activation_15 (Activation)  (None, 254, 254, 32)      0

 max_pooling2d_9 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_10 (Conv2D)          (None, 125, 125, 32)      9248

 activation_16 (Activation)  (None, 125, 125, 32)      0

 max_pooling2d_10 (MaxPoolin  (None, 62, 62, 32)       0
 g2D)

 conv2d_11 (Conv2D)          (None, 60, 60, 32)        9248

 activation_17 (Activation)  (None, 60, 60, 32)        0

 max_pooling2d_11 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_12 (Conv2D)          (None, 28, 28, 32)        9248

 activation_18 (Activation)  (None, 28, 28, 32)        0

 max_pooling2d_12 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 flatten_3 (Flatten)         (None, 6272)              0

 dense_6 (Dense)             (None, 64)                401472

 activation_19 (Activation)  (None, 64)                0

 dropout_3 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 3)                 195

 activation_20 (Activation)  (None, 3)                 0

=================================================================
Total params: 430,307
Trainable params: 430,307
Non-trainable params: 0
_________________________________________________________________
```


Figura 10: Estructura de modelo de CNN.

Como podemos observar, a medida que avanzamos en la red y aplicamos mayor número de convoluciones y de funciones de agrupación (pooling) a nuestra imagen se reduce las dimensiones del output, aumentando a su vez el número de parámetros. Por otro lado, cuando tras realizar todas las convoluciones aplicamos la función flatten apreciamos que la forma de nuestro output pasa de ser tridimensional a ser plana, es decir, bidimensional. Por último, se formarán una serie capas de neuronas que finalmente darán lugar a tres únicamente.

En cuanto a la capacidad de detección, tras haber validado el modelo con los datos de prueba podemos concluir que ha sido medianamente buena, ya que la precisión obtenida por nuestro modelo es del 0.62, lo que implica que se han clasificado correctamente un 62 % de las imágenes. Si bien es cierto que no son niveles de predicción malos estos no serían suficientes para nuestro objetivo, es decir, este tipo de algoritmo de reconocimiento de imagen no sería idóneo para detectar la enfermedad mediante pruebas de neuroimagen, sino que seguirían requiriendo de

otro tipo de pruebas para corroborar el diagnóstico predicho.

A pesar de todo, este mecanismo nos sería de gran ayuda a la hora de reducir la saturación hospitalaria, ya que representa un buen apoyo a la hora de la lectura de imágenes médicas. Además, en nuestro caso puede ser que este nivel de precisión se deba también a que el conjunto de datos no es demasiado amplio. Por ello, este mecanismo se podría ir refinando para finalmente alcanzar niveles de precisión de alta fiabilidad que no nos hagan dudar sobre la capacidad de clasificación.

Capítulo 8

Conclusiones

A lo largo de este trabajo, hemos visto una serie de análisis y predicciones con la intención de mejorar o ayudar de algún modo a la prevención y detección de la enfermedad de Alzheimer. En gran medida, hemos conseguido buenos resultados, ya que hemos cumplido todos y cada uno de los objetivos preestablecidos al mismo, pero si bien es cierto que aún se podrían mejorar.

En el caso del análisis de los factores de riesgo, hemos obtenido una serie de variables que influyen directamente sobre la presencia de esta enfermedad. Muchas de ellas, son algunos factores de riesgo que también confirman otros estudios similares a este, lo que corrobora una gran eficiencia en la búsqueda de estos a partir de los datos de pacientes estudiados. Esto puede ayudar enormemente a las personas, ya que se podría tratar de prevenir la enfermedad evitando algunos destos factores como el consumo de alcohol o cuidándonos frente afecciones respiratorias o relacionadas con la cabeza.

Muchos otros factores definidos como significativos en este estudio son inmutables, ya que son inherentes a las personas como la edad, el género o la altura y no se podría variar, por lo tanto, su efecto sobre la enfermedad. En cambio, todos ellos fueron utilizados para predecir posibles enfermos de EA mediante algoritmos de machine learning.

En este segundo punto del trabajo, hemos obtenido resultados dispares según que conjunto de datos y que variables utilizamos. Las predicciones realizadas a partir del las variables generales (dataset 1) han tenido una precisión bastante alta con casi un 70 %, mientras que en el caso de las variables relacionadas con los signos vitales y el historial médico del paciente la precisión es bastante baja superando apenas el 55 %.

De igual forma, en los resultados obtenidos de la detección de enfermos de EA utilizando imágenes de resonancia magnética se ha conseguido una precisión de entorno a un 65 %, es decir, bastante buena pero no lo suficiente como para utilizar este método de forma aislada para la detección de la patolgía.

En resumen, los resultados han sido por lo general buenos, pero insufientes, ya que estos méto-

dos siguen presentando un alto índice de error poniendo en duda la fiabilidad de los mismos. Esto ha podido deberse en alguno de los casos a la utilización de conjuntos de datos demasiado pequeños que no son suficientes para el entrenamiento de los modelos viéndose así afectada la eficiencia de predicción de los mismos.

De todas formas, estos mecanismos nos siguen siendo útiles para complementar otro tipo de pruebas diagnósticas y para sistematizar cada vez más el proceso de diagnóstico de pruebas de imagen, evitando así tiempos prolongados a la espera de resultados por parte paciente, mejorando la detección temprana y, por lo tanto, permitiendo una acción más rápida sobre el problema, así como, la reducción de la saturación hospitalaria. Por otro lado, también creo que debemos seguir explorando nuevos algoritmos para conformar nuevos métodos con tal fiabilidad que no precisemos de pruebas complementarias para corroborar diagnósticos.

Por todo ello, de cara a un futuro cercano muy prometedor en cuanto a biología computacional se refiere, creo que debemos seguir ampliando nuestro conocimiento y creando nuevas aplicaciones o softwares que utilicen sistemáticamente una serie de algoritmos que puedan llegar a complementarse entre si y dar una predicción consensuada con una precisión muy alta.

Por último, en lo referente al seguimiento del plan de trabajo y de la metodología propuesta inicialmente, hemos cumplido correctamente con su cronología a pesar de haber existido una serie de desviaciones temporales provocadas por algunos inconvenientes, como la obtención de los datos o pequeñas complicaciones con el uso de los softwares, estas han sido subsanadas correctamente. Por otro lado, la metodología utilizada ha variado ligeramente a la originalmente pensada, ya que TensorFlow no ha sido utilizado dentro del entorno de R Studio sino mediante un entorno Python debido a su mejor funcionamiento con esta aplicación.

En mi opinión, la biología, al igual que la medicina y muchas otras disciplinas, debe nutrirse de todos los recursos posibles de sus competencias transversales con el fin de mejorar. Por ello, debemos seguir implementando este tipo de tecnologías a nuestros estudios para poder obtener los mejores resultados posibles.

Capítulo 9

Glosario

- ML: machine learning.
- MRI: imágenes de resonancia magnética.
- CNN: red neuronal convolucional.
- CDR: escala del grado de demencia clínica.
- GDS: escala de deterioro global.
- OMS: organización mundial de la salud.
- PEC: prueba de evaluación continua.
- EA: enfermedad del Alzheimer.
- AChEI: inhibidores de acetilcolinesterasa.
- NMDA: N-metilmemantina (antagonista del receptor del aspartato).
- DMT: terapias modificadoras de la enfermedad.
- Aβ: proteína β-amiloide.
- APP: proteína precursora de Aβ.
- BACE1: β-secretasa.
- PSEN: proteína presenilina.
- APOE4: alelo del gen de la apolipoproteína E.
- LCR: líquido cefalorraquídeo.
- Aβ42: péptido Aβ1-42.

- T-tau: proteína tau total.
- P-tau: proteína tau fosforilada.
- EEG: pruebas electroencefalográficas.
- MEG: pruebas magnetoencefalográficas.
- PET: tomografía de emisión de positrones.
- ADNI: iniciativa de neuroimagen de la enfermedad del Alzheimer.
- DCL: deterioro cognitivo leve.
- ANOVA: análisis de la varianza.
- k-NN: algoritmo de los k vecinos más cercanos.
- SVM: algoritmo support vector machine.
- CN: estado cognitivo normal.
- MCI: deterioro cognitivo leve.
- AD: enfermo de Alzheimer.
- GM: materia gris.
- ReLu: función de activación de las redes neuronales convolucionales.

Bibliografía

[1] Castellani, R. J., Rolston, R. K., Smith, M. A. (2010). Alzheimer disease. Disease-a-month: DM, 56(9), 484.

[2] Mucke, L. (2009). Alzheimer's disease. Nature, 461(7266), 895-897.

[3] Apostolova, L. G. (2016). Alzheimer disease. Continuum: Lifelong Learning in Neurology, 22(2 Dementia), 419.

[4] Yiannopoulou, K. G., Papageorgiou, S. G. (2020). Current and future treatments in Alzheimer disease: an update. Journal of central nervous system disease, 12, 1179573520907397.

[5] Rossini, P.M., Di Iorio, R., Vecchio, F., Anfossi, M., Babiloni, C., Bozzali, M., ... Dubois, B. (2020). Diagnóstico precoz de la enfermedad de Alzheimer: el papel de los biomarcadores, incluido el análisis avanzado de señales de EEG. Informe del panel de expertos patrocinado por la IFCN. Neurofisiología Clínica, 131(6), 1287-1310.

[6] Data used in the preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). For up-to-date information, see www.adni-info.org.

[7] R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[8] RStudio Team (2021). RStudio: Integrated Development Environment for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.

[9] Max Kuhn (2021). caret: Classification and Regression Training. R package version 6.0-90. https://CRAN.R-project.org/package=caret

[10] Gregory R. Warnes, Ben Bolker, Thomas Lumley, Randall C Johnson. Contributions from Randall C. Johnson are Copyright SAIC-Frederick, Inc. Funded by the Intramural Research Program, of the NIH, National Cancer Institute and Center for Cancer Research under

NCI Contract NO1-CO-12400. (2018). gmodels: Various R Programming Tools for Model Fitting. R package version 2.18.1. https://CRAN.R-project.org/package=gmodels

[11] Venables, W. N. Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

[12] Stefan Fritsch, Frauke Guenther and Marvin N. Wright (2019). neuralnet: Training of Neural Networks. R package version 1.44.2. https://CRAN.R-project.org/package=neuralnet

[13] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2021). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-9. https://CRAN.R-project.org/package=e1071

[14] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis (2004). kernlab - An S4 Package for Kernel Methods in R. Journal of Statistical Software 11(9), 1-20. URL http://www.jstatsoft.org/v11/i09/

[15] Max Kuhn and Ross Quinlan (2021). C50: C5.0 Decision Trees and Rule-Based Models. R package version 0.1.5. https://CRAN.R-project.org/package=C50

[16] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean,...others. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[18] Van Rossum, G., Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

[19] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

[20] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[21] Bradski, G. (2000). The OpenCV Library. Dr. Dobbx27;s Journal of Software Tools.

[22] Umesh, P. (2012). Image Processing in Python. CSI Communications, 23.

[23] Pedregosa, F., Varoquaux, Ga.[el], Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[24] Chollet, F., others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[25] Pérez, Fernando and Granger, Brian E.. 2007. IPython: a System for Interactive Scientific Computing.

[26] Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.

[27] Hughes, C. P., Berg, L., Danziger, W., Coben, L. A., Martin, R. L. (1982). A new clinical scale for the staging of dementia. The British journal of psychiatry, 140(6), 566-572.

[28] Morris, J. C. (1997). Clinical dementia rating: a reliable and valid diagnostic and staging measure for dementia of the Alzheimer type. International psychogeriatrics, 9(S1), 173-176.

[29] https://docer.com.ar/doc/xxevexe

[30] Reisberg, B., Ferris, S. H., de Leon, M. J., Crook, T. (1982). The Global Deterioration Scale for assessment of primary degenerative dementia. The American journal of psychiatry.

[31] https://studylib.es/doc/4631945/escala-de-deterioro-global–gds

[32] Lantz, B. (2015). Machine learning with R: expert techniques for predictive modeling. Packt publishing ltd.

[33] Chandra, A., Dervenoulas, G., Politis, M. (2019). Magnetic resonance imaging in Alzheimer's disease and mild cognitive impairment. Journal of neurology, 266(6), 1293-1302.

[34] Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O. (2016). Deep learning for computational biology. Molecular systems biology, 12(7), 878.

[35] Min, S., Lee, B., Yoon, S. (2017). Deep learning in bioinformatics. Briefings in bioinformatics, 18(5), 851-869.

Anexo

Código R:

```
#~~~~~~~~~~~~#
# Análisis 1 #
#~~~~~~~~~~~~#

#============================#
# Importación de los datos: #
#============================#

# Estudio comparativo DIAN-ADNI:
adni_dian<-read.csv("ADNi_DIAN_Comparison_Data.csv",
    header = T)

#============================#
# Exploración de los datos: #
#============================#

str(adni_dian,list.len=ncol(adni_dian)) # Mostramos la estructura del dataset

## 'data.frame': 3225 obs. of  130 variables:
##  $ visit                          : Factor w/ 13 levels "v00","v01","v02",..: 1 2 3 4
##  $ visitage                       : num  76.3 76.8 77.3 78.4 79.4 ...
##  $ cdrglob                        : num  0 0 0 0.5 0 0.5 0.5 0.5 0.5 1 ...
##  $ visit_date                     : Factor w/ 1733 levels "1-Apr-08","1-Apr-09",..: 1
##  $ CSFDATE                        : Factor w/ 398 levels "","1/13/2016",..: 1 1 1 1 1
##  $ EDUC                           : int  18 18 18 18 18 17 17 17 17 17 ...
##  $ MRI_SCANDATE                   : Factor w/ 636 levels "","1/10/2014",..: 1 1 140 1
##  $ FLUF                           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ WORDIM                         : int  8 12 4 10 4 5 1 1 0 2 ...
##  $ LOGIMEM                        : int  18 NA 15 17 17 8 NA 6 NA 5 ...
##  $ DIGIF                          : int  12 11 12 6 12 7 8 7 7 7 ...
##  $ DIGIFLEN                       : int  8 8 8 5 8 6 6 6 6 7 ...
```

```
##  $ DIGIB                              : int  8 6 12 10 7 6 6 6 7 7 ...
##  $ DIGIBLEN                           : int  6 4 7 7 5 5 4 4 5 6 ...
##  $ ANIMALS                            : int  22 24 23 16 20 12 12 15 15 9 ...
##  $ VEG                                : int  15 18 18 14 16 8 5 7 6 6 ...
##  $ TRAILA                             : int  26 23 19 21 29 21 18 33 38 28 ...
##  $ TRAILARR                           : int  0 0 0 0 1 0 0 0 0 1 ...
##  $ TRAILB                             : int  52 39 62 39 34 98 70 90 66 60 ...
##  $ TRAILBRR                           : int  1 0 0 0 0 1 0 0 0 0 ...
##  $ WAIS                               : int  61 65 64 58 59 51 48 48 47 48 ...
##  $ MEMUNITS                           : int  17 NA 13 16 15 2 NA 0 NA 0 ...
##  $ BOSTON                             : int  29 28 29 29 30 26 30 28 27 27 ...
##  $ WORDDEL                            : int  5 7 5 9 4 2 2 3 0 1 ...
##  $ MR_TOTV_WMHYPOINTENSITIES          : num  NA NA 4097 4552 4824 ...
##  $ MR_TOTV_INTRACRANIAL               : num  NA NA 1621654 1631666 1624062 ...
##  $ MR_TOTV_HIPPOCAMPUS                : num  NA NA 6874 7104 6896 ...
##  $ FDG_fSUVR_rsf_TOT_CTX_PRECUNEUS    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ FDG_fSUVR_rsf_TOT_HIPPOCAMPUS      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ FDG_fSUVR_rsf_TOT_CORTMEAN         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ PIB_fSUVR_rsf_TOT_CTX_PRECUNEUS    : num  NA NA 1.25 1.33 1.36 ...
##  $ PIB_fSUVR_rsf_TOT_HIPPOCAMPUS      : num  NA NA 1.23 1.3 1.2 ...
##  $ PIB_fSUVR_rsf_TOT_CORTMEAN         : num  NA NA 1.05 1.07 1 ...
##  $ BIRTHMO                            : int  7 7 7 7 7 6 6 6 6 6 ...
##  $ BIRTHYR                            : int  1930 1930 1930 1930 1930 1934 1934 1934 1934
##  $ HISPANIC                           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RACE                               : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ PRIMLANG                           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ MARISTAT                           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ HANDED                             : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ MOMDEM                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MOMAUTO                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DADDEM                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DADAUTO                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ABRUPT                             : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ STEPWISE                           : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ SOMATIC                            : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ EMOT                               : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ HXHYPER                            : int  1 NA NA NA NA 1 NA NA NA NA ...
##  $ HXSTROKE                           : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ FOCLSYM                            : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ FOCLSIGN                           : int  0 NA NA NA NA 0 NA NA NA NA ...
##  $ HACHIN                             : int  1 NA NA NA NA 1 NA NA NA NA ...
##  $ CDRSUM                             : num  0 0 0 2 0.5 1 1 2 2.5 5 ...
```

```
##  $ GDS                          : int  2 NA 2 1 2 1 NA 0 NA 1 ...
##  $ BILLS                        : int  0 0 0 0 0 3 3 3 4 5 ...
##  $ TAXES                        : int  0 0 0 5 0 0 0 3 3 5 ...
##  $ SHOPPING                     : int  0 0 0 0 0 0 1 1 1 2 ...
##  $ GAMES                        : int  0 0 0 0 0 0 1 1 2 4 ...
##  $ STOVE                        : int  0 0 0 0 0 0 1 1 1 2 ...
##  $ MEALPREP                     : int  0 0 0 0 0 0 1 2 1 2 ...
##  $ EVENTS                       : int  0 0 0 0 0 0 1 1 2 4 ...
##  $ PAYATTN                      : int  0 0 0 0 0 0 0 0 0 3 ...
##  $ REMDATES                     : int  0 0 0 0 0 4 4 4 4 5 ...
##  $ TRAVEL                       : int  0 0 0 0 0 0 0 0 3 3 ...
##  $ DECSUB                       : int  0 NA 0 0 0 1 NA 0 NA 1 ...
##  $ VASC                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ALCDEM                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FTD                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PPAPH                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PSP                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CORT                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ HUNT                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PRION                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DEP                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PARK                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ HYCEPH                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ COGOTH                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ COGOTHX                      : Factor w/ 2 levels "","UNK": 1 1 1 1 1 1 1 1 1 1
##  $ gender                       : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2
##  $ CSF_ELC_AB42                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ CSF_ELC_ptau                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CSF_ELC_tau                  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ CSF_ELC_AB40                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ CSF_ELC_AB4240               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MSP_AB38                     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MSP_AB40                     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MSP_AB42                     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ csf_date_bl                  : Factor w/ 173 levels "","1-Apr-08",..: 1 1 1 1 1
##  $ mri_date_bl                  : Factor w/ 289 levels "","1-Apr-11",..: 23 23 23 2
##  $ csf_cdr_bl                   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ mri_cdr_bl                   : num  0 0 0 0 0 NA NA NA NA NA ...
##  $ csf_cdrsb_bl                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ mri_cdrsb_bl                 : num  0 0 0 0 0 NA NA NA NA NA ...
##  $ visit_date_bl                : Factor w/ 445 levels "","1-Apr-11",..: 53 53 53 5
##  $ MR_TOTT_PRECUNEUS            : num  NA NA 4.06 4.14 4.13 ...
```

```
##  $ last_visit_date                  : Factor w/ 449 levels "1-Apr-08","1-Apr-13",..: 75
##  $ time_cognitive_followup          : num  3.09 3.09 3.09 3.09 3.09 ...
##  $ time_csf_followup                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ time_mri_followup                : num  2.09 2.09 2.09 2.09 2.09 ...
##  $ cdr_baseline                     : num  0 0 0 0 0 0.5 0.5 0.5 0.5 0.5 ...
##  $ group_CSF                        : int  99 99 99 99 99 99 99 99 99 99 ...
##  $ group_MRI                        : int  1 1 1 1 1 99 99 99 99 99 ...
##  $ cohort                           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RID                              : int  1002 1002 1002 1002 1002 1007 1007 1007 1007
##  $ pet_type                         : Factor w/ 4 levels "","AV45 only",..: 1 1 1 1 1 1
##  $ PiB_mSUVR_TOT_PRECUNEUS          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ PiB_mSUVR_TOT_CORTMEAN           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ FDG_mSUVR_TOT_PRECUNEUS          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ FDG_mSUVR_TOT_CORTMEAN           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ AV45_fSUVR_rsf_TOT_CTX_PRECUNEUS: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ AV45_fSUVR_rsf_TOT_HIPPOCAMPUS   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ AV45_fSUVR_rsf_TOT_CORTMEAN      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ AV45_mSUVR_TOT_PRECUNEUS         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ AV45_mSUVR_TOT_CORTMEAN          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ pet_date_bl                      : Factor w/ 160 levels "","1-Jun-11",..: 1 1 1 1 1
##  $ pet_cdr_bl                       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ pet_cdrsb_bl                     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ time_pet_followup                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ ADgroup                          : int  0 0 0 0 0 1 1 1 1 1 ...
##  $ group_PET                        : int  99 99 99 99 99 99 99 99 99 99 ...
##  $ PET_scandate                     : Factor w/ 410 levels "","1-Aug-08",..: 1 1 1 1 1
##  $ origin                           : Factor w/ 1 level "ADNI": 1 1 1 1 1 1 1 1 1 1 ...
##  $ DIAN_ID                          : int  1 1 1 1 1 3 3 3 3 3 ...
##  $ DIAN_APOE                        : int  33 33 33 33 33 33 33 33 33 33 ...
##  $ DIAN_GROUP                       : int  1 1 1 1 1 2 2 2 2 2 ...
##  $ DIAN_CDRSB_BL                    : num  0 0 0 0 0 1 1 1 1 1 ...
##  $ DIAN_years_bl                    : num  0 0.485 1.024 2.092 3.088 ...
##  $ DIAN_MMSE                        : int  30 30 29 30 29 30 30 29 30 25 ...
##  $ nEYO                             : num  -2.092 -1.607 -1.068 0 0.997 ...
```

```
summary(adni_dian) # Mostramos el resumen estadístico del dataset
```

```
##      visit        visitage        cdrglob          visit_date
##  v00    :571   Min.   :55.00   Min.   :0.000   12-Apr-12:  10
##  v01    :556   1st Qu.:71.22   1st Qu.:0.500   11-Dec-13:   8
##  v02    :527   Median :76.40   Median :0.500   14-Mar-13:   8
##  v03    :416   Mean   :76.01   Mean   :0.631   15-Feb-12:   8
##  v04    :331   3rd Qu.:81.20   3rd Qu.:1.000   4-Dec-12 :   8
```

```
##  v05    :261   Max.   :94.66   Max.   :3.000   10-Oct-12:   7
##  (Other):563                   NA's   :37      (Other)  :3176
##        CSFDATE          EDUC           MRI_SCANDATE       FLUF
##            :2788   Min.   : 8.00             :2458   Min.   : 0.00
##  11/10/2010:   3   1st Qu.:14.00   4/19/2013 :   4   1st Qu.: 9.00
##  4/14/2014 :   3   Median :16.00   1/23/2013 :   3   Median :12.00
##  1/29/2014 :   2   Mean   :15.83   10/17/2011:   3   Mean   :12.29
##  10/19/2011:   2   3rd Qu.:18.00   10/5/2010 :   3   3rd Qu.:16.00
##  10/25/2013:   2   Max.   :20.00   11/6/2013 :   3   Max.   :30.00
##  (Other)   : 425                   (Other)   : 751   NA's   :991
##      WORDIM          LOGIMEM          DIGIF           DIGIFLEN
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :3.000
##  1st Qu.: 1.000   1st Qu.: 4.000   1st Qu.: 7.000   1st Qu.:6.000
##  Median : 3.000   Median : 8.000   Median : 8.000   Median :6.000
##  Mean   : 3.802   Mean   : 8.212   Mean   : 8.082   Mean   :6.478
##  3rd Qu.: 6.000   3rd Qu.:12.000   3rd Qu.:10.000   3rd Qu.:7.000
##  Max.   :15.000   Max.   :23.000   Max.   :12.000   Max.   :8.000
##  NA's   :141      NA's   :751      NA's   :2346     NA's   :2347
##      DIGIB           DIGIBLEN        ANIMALS           VEG
##  Min.   : 0.000   Min.   :2.000   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 5.000   1st Qu.:4.000   1st Qu.:11.00   1st Qu.: 8.00
##  Median : 6.000   Median :5.000   Median :15.00   Median :11.00
##  Mean   : 6.264   Mean   :4.593   Mean   :15.24   Mean   :10.92
##  3rd Qu.: 7.000   3rd Qu.:5.000   3rd Qu.:19.00   3rd Qu.:13.00
##  Max.   :12.000   Max.   :7.000   Max.   :39.00   Max.   :27.00
##  NA's   :2354     NA's   :2355    NA's   :114     NA's   :2346
##      TRAILA          TRAILARR         TRAILB         TRAILBRR
##  Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.000
##  1st Qu.: 30.00   1st Qu.: 0.00   1st Qu.: 76.0   1st Qu.: 0.000
##  Median : 38.00   Median : 0.00   Median :105.0   Median : 1.000
##  Mean   : 47.36   Mean   : 0.16   Mean   :135.1   Mean   : 1.041
##  3rd Qu.: 53.00   3rd Qu.: 0.00   3rd Qu.:172.0   3rd Qu.: 2.000
##  Max.   :301.00   Max.   :10.00   Max.   :510.0   Max.   :12.000
##  NA's   :148      NA's   :149     NA's   :301     NA's   :301
##       WAIS          MEMUNITS         BOSTON         WORDDEL
##  Min.   : 0.00   Min.   : 0.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.:32.00   1st Qu.: 0.000   1st Qu.:24.00   1st Qu.: 0.000
##  Median :39.00   Median : 5.000   Median :27.00   Median : 1.000
##  Mean   :39.19   Mean   : 5.561   Mean   :25.45   Mean   : 2.587
##  3rd Qu.:47.00   3rd Qu.: 9.000   3rd Qu.:29.00   3rd Qu.: 4.000
##  Max.   :78.00   Max.   :24.000   Max.   :30.00   Max.   :15.000
##  NA's   :2357    NA's   :771      NA's   :132     NA's   :145
```

```
##  MR_TOTV_WMHYPOINTENSITIES MR_TOTV_INTRACRANIAL MR_TOTV_HIPPOCAMPUS
##  Min.   :  865.6           Min.   : 961509      Min.   : 2382
##  1st Qu.: 2916.7           1st Qu.:1405187      1st Qu.: 5374
##  Median : 4552.2           Median :1506249      Median : 6258
##  Mean   : 7248.1           Mean   :1522652      Mean   : 6284
##  3rd Qu.: 8294.8           3rd Qu.:1635076      3rd Qu.: 7258
##  Max.   :54100.1           Max.   :3384699      Max.   :10563
##  NA's   :2458              NA's   :2458         NA's   :2458
##  FDG_fSUVR_rsf_TOT_CTX_PRECUNEUS FDG_fSUVR_rsf_TOT_HIPPOCAMPUS
##  Min.   :1.043                   Min.   :0.5560
##  1st Qu.:1.617                   1st Qu.:0.7135
##  Median :1.737                   Median :0.7690
##  Mean   :1.731                   Mean   :0.7715
##  3rd Qu.:1.856                   3rd Qu.:0.8330
##  Max.   :2.196                   Max.   :0.9770
##  NA's   :2966                    NA's   :2966
##  FDG_fSUVR_rsf_TOT_CORTMEAN PIB_fSUVR_rsf_TOT_CTX_PRECUNEUS
##  Min.   :1.170              Min.   :0.991
##  1st Qu.:1.430              1st Qu.:1.304
##  Median :1.502              Median :2.948
##  Mean   :1.511              Mean   :2.791
##  3rd Qu.:1.595              3rd Qu.:3.959
##  Max.   :1.905              Max.   :6.394
##  NA's   :2966               NA's   :3174
##  PIB_fSUVR_rsf_TOT_HIPPOCAMPUS PIB_fSUVR_rsf_TOT_CORTMEAN    BIRTHMO
##  Min.   :0.895                 Min.   :0.895              Min.   : 1.000
##  1st Qu.:1.234                 1st Qu.:1.054              1st Qu.: 4.000
##  Median :1.326                 Median :2.124              Median : 7.000
##  Mean   :1.369                 Mean   :2.203              Mean   : 6.671
##  3rd Qu.:1.509                 3rd Qu.:3.015              3rd Qu.:10.000
##  Max.   :2.024                 Max.   :5.160              Max.   :12.000
##  NA's   :3174                  NA's   :3174
##     BIRTHYR        HISPANIC            RACE           PRIMLANG
##  Min.   :1918   Min.   :0.00000   Min.   : 1.000   Min.   :1.000
##  1st Qu.:1930   1st Qu.:0.00000   1st Qu.: 1.000   1st Qu.:1.000
##  Median :1935   Median :0.00000   Median : 1.000   Median :1.000
##  Mean   :1935   Mean   :0.04868   Mean   : 1.427   Mean   :1.054
##  3rd Qu.:1940   3rd Qu.:0.00000   3rd Qu.: 1.000   3rd Qu.:1.000
##  Max.   :1957   Max.   :9.00000   Max.   :50.000   Max.   :8.000
##
##     MARISTAT        HANDED          MOMDEM          MOMAUTO
##  Min.   :1.00   Min.   :1.000   Min.   :0.0000   Min.   :0.000
```

```
##  1st Qu.:1.00   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :1.00   Median :2.000   Median :0.0000   Median :1.000
##  Mean   :1.33   Mean   :1.909   Mean   :0.6083   Mean   :1.562
##  3rd Qu.:1.00   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :9.00   Max.   :2.000   Max.   :9.0000   Max.   :9.000
##                                 NA's   :3        NA's   :1360
##      DADDEM          DADAUTO         ABRUPT         STEPWISE
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00   Median :0.000   Median :0.0000
##  Mean   :0.5634   Mean   :1.12   Mean   :0.014   Mean   :0.0105
##  3rd Qu.:0.0000   3rd Qu.:1.00   3rd Qu.:0.000   3rd Qu.:0.0000
##  Max.   :9.0000   Max.   :9.00   Max.   :2.000   Max.   :1.0000
##  NA's   :46       NA's   :1811   NA's   :2654    NA's   :2654
##     SOMATIC            EMOT          HXHYPER          HXSTROKE
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.000   Median :0.0000   Median :0.0000
##  Mean   :0.0123   Mean   :0.014   Mean   :0.4991   Mean   :0.0245
##  3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :2.0000
##  NA's   :2654     NA's   :2654    NA's   :2654     NA's   :2654
##     FOCLSYM         FOCLSIGN         HACHIN           CDRSUM
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   : 0.00
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.: 1.00
##  Median :0.000   Median :0.000   Median :1.0000   Median : 2.50
##  Mean   :0.028   Mean   :0.021   Mean   :0.6235   Mean   : 3.12
##  3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.: 4.50
##  Max.   :2.000   Max.   :2.000   Max.   :4.0000   Max.   :18.00
##  NA's   :2654    NA's   :2654    NA's   :2654     NA's   :37
##       GDS             BILLS           TAXES          SHOPPING
##  Min.   : 0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.: 1.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
##  Median : 1.000   Median :2.000   Median :2.000   Median :0.000
##  Mean   : 1.903   Mean   :2.099   Mean   :2.269   Mean   :1.615
##  3rd Qu.: 3.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:3.000
##  Max.   :13.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##  NA's   :325      NA's   :25      NA's   :25      NA's   :23
##      GAMES            STOVE          MEALPREP         EVENTS
##  Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.000   Median :0.0000   Median :0.000   Median :0.000
```

```
##  Mean   :1.454   Mean   :0.7424   Mean   :1.482   Mean   :1.606
##  3rd Qu.:3.000   3rd Qu.:0.0000   3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   :5.000   Max.   :5.0000   Max.   :5.000   Max.   :5.000
##  NA's   :23      NA's   :22       NA's   :22      NA's   :22
##     PAYATTN         REMDATES         TRAVEL          DECSUB
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.000   Median :3.000   Median :0.000   Median :1.0000
##  Mean   :1.374   Mean   :2.375   Mean   :1.953   Mean   :0.5045
##  3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:1.0000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :1.0000
##  NA's   :23      NA's   :21      NA's   :22      NA's   :323
##       VASC              ALCDEM       FTD                 PPAPH        PSP
##  Min.   :0.000000   Min.   :0   Min.   :0.0000000   Min.   :0   Min.   :0
##  1st Qu.:0.000000   1st Qu.:0   1st Qu.:0.0000000   1st Qu.:0   1st Qu.:0
##  Median :0.000000   Median :0   Median :0.0000000   Median :0   Median :0
##  Mean   :0.002791   Mean   :0   Mean   :0.0009302   Mean   :0   Mean   :0
##  3rd Qu.:0.000000   3rd Qu.:0   3rd Qu.:0.0000000   3rd Qu.:0   3rd Qu.:0
##  Max.   :1.000000   Max.   :0   Max.   :1.0000000   Max.   :0   Max.   :0
##
##       CORT              HUNT       PRION        DEP
##  Min.   :0.00000   Min.   :0   Min.   :0   Min.   :0.0000000
##  1st Qu.:0.00000   1st Qu.:0   1st Qu.:0   1st Qu.:0.0000000
##  Median :0.00000   Median :0   Median :0   Median :0.0000000
##  Mean   :0.00186   Mean   :0   Mean   :0   Mean   :0.0009302
##  3rd Qu.:0.00000   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0.0000000
##  Max.   :1.00000   Max.   :0   Max.   :0   Max.   :1.0000000
##
##       PARK               HYCEPH              COGOTH         COGOTHX
##  Min.   :0.0000000   Min.   :0.000000   Min.   :0.000000      :3195
##  1st Qu.:0.0000000   1st Qu.:0.000000   1st Qu.:0.000000   UNK:  30
##  Median :0.0000000   Median :0.000000   Median :0.000000
##  Mean   :0.0003101   Mean   :0.003721   Mean   :0.009302
##  3rd Qu.:0.0000000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.0000000   Max.   :1.000000   Max.   :1.000000
##
##     gender      CSF_ELC_AB42     CSF_ELC_ptau     CSF_ELC_tau
##  Female:1308   Min.   : 238.0   Min.   :  8.50   Min.   :110.0
##  Male  :1917   1st Qu.: 542.5   1st Qu.: 22.10   1st Qu.:248.2
##                Median : 685.5   Median : 29.70   Median :318.0
##                Mean   : 845.6   Mean   : 34.76   Mean   :350.3
##                3rd Qu.: 948.0   3rd Qu.: 43.00   3rd Qu.:425.8
```

```
##                Max.   :2982.0   Max.   :117.70   Max.   :946.0
##                NA's   :2803     NA's   :2804     NA's   :2803
##   CSF_ELC_AB40   CSF_ELC_AB4240      MSP_AB38       MSP_AB40
##  Min.   : 5550   Min.   :0.0199   Min.   : 561   Min.   : 2709
##  1st Qu.:13848   1st Qu.:0.0323   1st Qu.:1450   1st Qu.: 6408
##  Median :17825   Median :0.0403   Median :1846   Median : 8229
##  Mean   :17957   Mean   :0.0475   Mean   :1876   Mean   : 8103
##  3rd Qu.:21550   3rd Qu.:0.0501   3rd Qu.:2272   3rd Qu.: 9553
##  Max.   :37590   Max.   :0.1036   Max.   :4107   Max.   :16505
##  NA's   :2803    NA's   :2803     NA's   :2799   NA's   :2799
##     MSP_AB42         csf_date_bl      mri_date_bl     csf_cdr_bl
##  Min.   : 244.0            :2058            :1050   Min.   :0.0000
##  1st Qu.: 593.2   8-Feb-11 :  24   16-Feb-11:  28   1st Qu.:0.5000
##  Median : 753.5   10-Nov-10:  19   13-Oct-10:  24   Median :0.5000
##  Mean   : 892.9   17-Mar-11:  13   10-Aug-11:  22   Mean   :0.4589
##  3rd Qu.:1038.0   20-Oct-10:  13   8-Mar-11 :  21   3rd Qu.:0.5000
##  Max.   :3738.0   22-Sep-11:  13   3-May-07 :  20   Max.   :2.0000
##  NA's   :2799     (Other)  :1085   (Other)  :2060   NA's   :2058
##    mri_cdr_bl      csf_cdrsb_bl     mri_cdrsb_bl      visit_date_bl
##  Min.   :0.0000   Min.   : 0.000   Min.   : 0.000   11-Oct-06:  31
##  1st Qu.:0.5000   1st Qu.: 0.500   1st Qu.: 0.500   4-Dec-06 :  30
##  Median :0.5000   Median : 1.500   Median : 1.500   14-Dec-06:  29
##  Mean   :0.4891   Mean   : 1.789   Mean   : 2.052   28-Mar-06:  28
##  3rd Qu.:0.5000   3rd Qu.: 2.500   3rd Qu.: 3.000   12-Apr-06:  26
##  Max.   :2.0000   Max.   :10.000   Max.   :14.000   1-Feb-06 :  25
##  NA's   :1060     NA's   :2058     NA's   :1060     (Other)  :3056
##  MR_TOTT_PRECUNEUS  last_visit_date time_cognitive_followup time_csf_followup
##  Min.   :2.928     10-Feb-16:  48   Min.   : 0.000          Min.   :0.9508
##  1st Qu.:3.946     4-Apr-16 :  25   1st Qu.: 3.004          1st Qu.:3.0200
##  Median :4.157     16-Nov-16:  24   Median : 4.197          Median :4.0219
##  Mean   :4.139     27-Jan-16:  24   Mean   : 5.128          Mean   :4.1965
##  3rd Qu.:4.383     29-Mar-16:  24   3rd Qu.: 8.000          3rd Qu.:5.0055
##  Max.   :5.044     11-Aug-15:  23   Max.   :10.178          Max.   :9.0835
##  NA's   :2458      (Other)  :3057   NA's   :10              NA's   :2058
##  time_mri_followup  cdr_baseline      group_CSF       group_MRI         cohort
##  Min.   :-0.0548   Min.   :0.0000   Min.   : 1.00   Min.   : 1.00   Min.   :0
##  1st Qu.: 2.9476   1st Qu.:0.5000   1st Qu.: 2.00   1st Qu.: 2.00   1st Qu.:0
##  Median : 4.0466   Median :0.5000   Median :99.00   Median : 2.00   Median :0
##  Mean   : 4.1101   Mean   :0.4636   Mean   :63.91   Mean   :33.94   Mean   :0
##  3rd Qu.: 5.0696   3rd Qu.:0.5000   3rd Qu.:99.00   3rd Qu.:99.00   3rd Qu.:0
##  Max.   : 9.0835   Max.   :2.0000   Max.   :99.00   Max.   :99.00   Max.   :0
##  NA's   :1050
```

```
##       RID            pet_type    PiB_mSUVR_TOT_PRECUNEUS PiB_mSUVR_TOT_CORTMEAN
##  Min.   :  42            :2742   Min.   :1.190           Min.   :1.169
##  1st Qu.: 863   AV45 only: 397   1st Qu.:1.434           1st Qu.:1.177
##  Median :2216   PIB only :  35   Median :1.678           Median :1.184
##  Mean   :2627   PIB&AV45 :  51   Mean   :1.653           Mean   :1.282
##  3rd Qu.:4542                    3rd Qu.:1.884           3rd Qu.:1.339
##  Max.   :5295                    Max.   :2.090           Max.   :1.494
##                                  NA's   :3222            NA's   :3222
##  FDG_mSUVR_TOT_PRECUNEUS FDG_mSUVR_TOT_CORTMEAN
##  Min.   :1.023           Min.   :0.866
##  1st Qu.:1.096           1st Qu.:0.932
##  Median :1.172           Median :0.990
##  Mean   :1.163           Mean   :0.986
##  3rd Qu.:1.216           3rd Qu.:1.043
##  Max.   :1.363           Max.   :1.086
##  NA's   :3192            NA's   :3192
##  AV45_fSUVR_rsf_TOT_CTX_PRECUNEUS AV45_fSUVR_rsf_TOT_HIPPOCAMPUS
##  Min.   :0.495                    Min.   :0.765
##  1st Qu.:1.087                    1st Qu.:1.096
##  Median :2.065                    Median :1.190
##  Mean   :1.998                    Mean   :1.197
##  3rd Qu.:2.672                    3rd Qu.:1.294
##  Max.   :4.422                    Max.   :1.773
##  NA's   :2687                     NA's   :2687
##  AV45_fSUVR_rsf_TOT_CORTMEAN AV45_mSUVR_TOT_PRECUNEUS AV45_mSUVR_TOT_CORTMEAN
##  Min.   :0.590               Min.   :0.827            Min.   :0.881
##  1st Qu.:1.063               1st Qu.:1.195            1st Qu.:1.125
##  Median :1.838               Median :1.558            Median :1.407
##  Mean   :1.771               Mean   :1.498            Mean   :1.395
##  3rd Qu.:2.321               3rd Qu.:1.724            3rd Qu.:1.603
##  Max.   :3.684               Max.   :2.213            Max.   :2.031
##  NA's   :2687                NA's   :3133             NA's   :3133
##     pet_date_bl     pet_cdr_bl      pet_cdrsb_bl    time_pet_followup
##           :2058   Min.   :0.0000   Min.   : 0.000   Min.   :0.7142
##  10-Nov-10:  22   1st Qu.:0.5000   1st Qu.: 0.500   1st Qu.:3.0221
##  25-Feb-11:  17   Median :0.5000   Median : 1.500   Median :4.0191
##  15-Sep-10:  16   Mean   :0.4589   Mean   : 1.789   Mean   :4.1933
##  16-Aug-11:  16   3rd Qu.:0.5000   3rd Qu.: 2.500   3rd Qu.:5.0092
##  16-Feb-12:  15   Max.   :2.0000   Max.   :10.000   Max.   :9.0232
##  (Other)  :1081   NA's   :2058     NA's   :2058     NA's   :2058
##     ADgroup         group_PET        PET_scandate   origin        DIAN_ID
##  Min.   :0.0000   Min.   : 1.00            :2756   ADNI:3225   Min.   :  1.0
```

```
##  1st Qu.:1.0000   1st Qu.: 2.00   12-Jun-13:   3              1st Qu.:108.0
##  Median :1.0000   Median :99.00   24-Mar-14:   3              Median :241.0
##  Mean   :0.8335   Mean   :63.91   24-Oct-11:   3              Mean   :263.6
##  3rd Qu.:1.0000   3rd Qu.:99.00   3-Oct-12 :   3              3rd Qu.:414.0
##  Max.   :1.0000   Max.   :99.00   10-Apr-14:   2              Max.   :571.0
##                                   (Other)  : 455
##    DIAN_APOE       DIAN_GROUP   DIAN_CDRSB_BL   DIAN_years_bl
##  Min.   :22.00   Min.   :1.00   Min.   : 0.00   Min.   : 0.0000
##  1st Qu.:33.00   1st Qu.:2.00   1st Qu.: 0.50   1st Qu.: 0.4983
##  Median :34.00   Median :2.00   Median : 1.50   Median : 1.1280
##  Mean   :34.11   Mean   :1.98   Mean   : 1.78   Mean   : 2.2442
##  3rd Qu.:34.00   3rd Qu.:2.00   3rd Qu.: 2.50   3rd Qu.: 3.1102
##  Max.   :44.00   Max.   :4.00   Max.   :10.00   Max.   :10.1766
##  NA's   :1
##    DIAN_MMSE          nEYO
##  Min.   : 0.00   Min.   :-9.0842
##  1st Qu.:24.00   1st Qu.: 0.5147
##  Median :27.00   Median : 2.8765
##  Mean   :25.59   Mean   : 3.0610
##  3rd Qu.:29.00   3rd Qu.: 5.3838
##  Max.   :30.00   Max.   :17.5816
##  NA's   :81
```

```
dim(adni_dian) # Mostramos el número de variables y registros
```

```
## [1] 3225  130
```

```
#=====================================#
# Limpieza y preparación de los datos: #
#=====================================#

# Seleccionamos las vatiables que nos interesan
adni_dian<-adni_dian[,c(3,54,55,2,80,6,36,37,38,39,40,41,43)]
adni_dian<-na.omit(adni_dian) # Eliminamos los datos perdidos (NA)
dim(adni_dian) # Mostramos el número de variables y registros
```

```
## [1] 2831   13
```

```
adni_dian$gender<-factor(adni_dian$gender,levels = c("Male","Female"),
labels = c("1","0"))
# Transformamos la varaible "gender" a numérica
adni_dian$gender<-as.numeric(adni_dian$gender)
```

```
#======================#
# Análisis de regresión #
#======================#

# Realizamos diferentes modelos de regresión con las distintas variables
# respuesta:
adni_dian_lm1<-lm(CDRSUM ~ visitage + gender + EDUC + HISPANIC + RACE +
PRIMLANG + MARISTAT + HANDED + MOMDEM + DADDEM,adni_dian) # CDRSUM
adni_dian_lm2<-lm(GDS ~ visitage + gender + EDUC + HISPANIC + RACE + PRIMLANG +
MARISTAT + HANDED + MOMDEM + DADDEM,adni_dian) # GDS
adni_dian_lm3<-lm(cdrglob ~ visitage + gender + EDUC + HISPANIC + RACE +
PRIMLANG + MARISTAT + HANDED + MOMDEM + DADDEM,adni_dian) # cdrglob
# Visualizamos los resultados de los modelos:
summary(adni_dian_lm1)

##
## Call:
## lm(formula = CDRSUM ~ visitage + gender + EDUC + HISPANIC + RACE +
##     PRIMLANG + MARISTAT + HANDED + MOMDEM + DADDEM, data = adni_dian)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.582 -1.940 -0.635  1.361 14.289
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.143658   0.765571   2.800  0.00514 **
## visitage     0.019715   0.006912   2.852  0.00437 **
## gender       0.148656   0.107894   1.378  0.16838
## EDUC        -0.028398   0.018795  -1.511  0.13091
## HISPANIC     0.218911   0.102873   2.128  0.03343 *
## RACE        -0.007766   0.011834  -0.656  0.51171
## PRIMLANG     0.038007   0.092654   0.410  0.68169
## MARISTAT    -0.228845   0.057938  -3.950 8.01e-05 ***
## HANDED      -0.051174   0.178611  -0.287  0.77451
## MOMDEM      -0.030858   0.037049  -0.833  0.40498
## DADDEM       0.067535   0.028038   2.409  0.01607 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.714 on 2820 degrees of freedom
```

```
## Multiple R-squared:  0.01295,Adjusted R-squared:  0.009448
## F-statistic: 3.699 on 10 and 2820 DF,  p-value: 6.119e-05

summary(adni_dian_lm2)

##
## Call:
## lm(formula = GDS ~ visitage + gender + EDUC + HISPANIC + RACE +
##     PRIMLANG + MARISTAT + HANDED + MOMDEM + DADDEM, data = adni_dian)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3676 -1.2086 -0.7567  0.8596 10.8754
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.519517   0.556151   0.934  0.35032
## visitage     0.001710   0.005021   0.341  0.73339
## gender       0.357555   0.078380   4.562 5.29e-06 ***
## EDUC         0.006129   0.013653   0.449  0.65355
## HISPANIC     0.116091   0.074732   1.553  0.12043
## RACE        -0.012785   0.008597  -1.487  0.13708
## PRIMLANG    -0.034401   0.067309  -0.511  0.60933
## MARISTAT     0.034585   0.042089   0.822  0.41131
## HANDED       0.350641   0.129752   2.702  0.00693 **
## MOMDEM      -0.006085   0.026914  -0.226  0.82114
## DADDEM       0.017800   0.020369   0.874  0.38225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.971 on 2820 degrees of freedom
## Multiple R-squared:  0.01274,Adjusted R-squared:  0.009241
## F-statistic:  3.64 on 10 and 2820 DF,  p-value: 7.734e-05

summary(adni_dian_lm3)

##
## Call:
## lm(formula = cdrglob ~ visitage + gender + EDUC + HISPANIC +
##     RACE + PRIMLANG + MARISTAT + HANDED + MOMDEM + DADDEM, data = adni_dian)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -0.7470 -0.1381 -0.1088  0.2733  2.4276
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.6260557  0.1182104   5.296 1.27e-07 ***
## visitage     0.0011849  0.0010672   1.110  0.26700
## gender       0.0076938  0.0166598   0.462  0.64424
## EDUC        -0.0058369  0.0029021  -2.011  0.04439 *
## HISPANIC     0.0124705  0.0158844   0.785  0.43247
## RACE        -0.0007872  0.0018273  -0.431  0.66664
## PRIMLANG    -0.0050930  0.0143066  -0.356  0.72187
## MARISTAT    -0.0230144  0.0089461  -2.573  0.01015 *
## HANDED       0.0093678  0.0275790   0.340  0.73413
## MOMDEM      -0.0073795  0.0057207  -1.290  0.19717
## DADDEM       0.0120054  0.0043294   2.773  0.00559 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.419 on 2820 degrees of freedom
## Multiple R-squared:  0.007155,Adjusted R-squared:  0.003634
## F-statistic: 2.032 on 10 and 2820 DF,  p-value: 0.02672

# Realizamos una ANOVA con cada uno:
anova(adni_dian_lm1)

## Analysis of Variance Table
##
## Response: CDRSUM
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## visitage     1    53.5  53.513  7.2662  0.007068 **
## gender       1    10.7  10.744  1.4589  0.227213
## EDUC         1    10.1  10.104  1.3720  0.241567
## HISPANIC     1    34.1  34.120  4.6329  0.031449 *
## RACE         1     3.8   3.847  0.5223  0.469923
## PRIMLANG     1     1.7   1.731  0.2350  0.627895
## MARISTAT     1   114.2 114.207 15.5073 8.418e-05 ***
## HANDED       1     1.2   1.217  0.1653  0.684340
## MOMDEM       1     0.2   0.227  0.0308  0.860778
## DADDEM       1    42.7  42.727  5.8016  0.016075 *
## Residuals 2820 20768.4   7.365
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(adni_dian_lm2)
```

```
## Analysis of Variance Table
##
## Response: GDS
##             Df  Sum Sq Mean Sq F value   Pr(>F)
## visitage     1     0.0   0.003  0.0007 0.978176
## gender       1    88.7  88.732 22.8304 1.86e-06 ***
## EDUC         1     1.4   1.429  0.3677 0.544283
## HISPANIC     1     8.6   8.582  2.2081 0.137400
## RACE         1     9.7   9.689  2.4929 0.114477
## PRIMLANG     1     0.9   0.913  0.2350 0.627894
## MARISTAT     1     1.6   1.590  0.4091 0.522455
## HANDED       1    27.5  27.547  7.0876 0.007806 **
## MOMDEM       1     0.0   0.001  0.0002 0.987617
## DADDEM       1     3.0   2.968  0.7637 0.382246
## Residuals 2820 10960.2   3.887
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(adni_dian_lm3)

## Analysis of Variance Table
##
## Response: cdrglob
##             Df Sum Sq Mean Sq F value   Pr(>F)
## visitage     1   0.20 0.19811  1.1282 0.288241
## gender       1   0.05 0.05072  0.2889 0.590997
## EDUC         1   0.57 0.57128  3.2535 0.071377 .
## HISPANIC     1   0.11 0.10616  0.6046 0.436904
## RACE         1   0.05 0.04641  0.2643 0.607221
## PRIMLANG     1   0.02 0.01649  0.0939 0.759264
## MARISTAT     1   1.17 1.16949  6.6604 0.009908 **
## HANDED       1   0.01 0.00719  0.0410 0.839601
## MOMDEM       1   0.05 0.05242  0.2985 0.584840
## DADDEM       1   1.35 1.35021  7.6896 0.005590 **
## Residuals 2820 495.16 0.17559
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Visualización de los datos según las variables
cdr<-as.factor(adni_dian$cdrglob)
HISPANIC<-as.factor(adni_dian$HISPANIC)
table(adni_dian$gender)
```

```
##
##    1    2
## 1705 1126
```

```
#~~~~~~~~~~~~#
# Análisis 2 #
#~~~~~~~~~~~~#

#============================#
# Importación de los datos: #
#============================#

# Signos vitales:
vitals<-read.csv("VITALS.csv",header = T)

#Historial médico:
medhist<-read.csv("MEDHIST.csv",header = T)

# CDR (Clinical Dementia Rate):
cdr<-read.csv("CDR.csv",header = T)

#============================#
# Exploración de los datos: #
#============================#

# Signos vitales:
str(vitals,list.len=ncol(vitals)) # Mostramos la estructura del dataset

## 'data.frame': 14455 obs. of  22 variables:
##  $ Phase       : Factor w/ 4 levels "ADNI1","ADNI2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ID          : int  14 16 18 20 22 24 26 28 30 32 ...
##  $ RID         : int  2 3 4 5 7 9 8 11 13 2 ...
##  $ SITEID      : int  107 107 10 107 10 10 107 107 10 107 ...
##  $ VISCODE     : Factor w/ 26 levels "bl","f","init",..: 12 12 12 12 12 2 12 2 2 1 ...
##  $ VISCODE2    : Factor w/ 35 levels "","bl","f","m06",..: 34 34 34 34 34 3 34 3 3 2 .
##  $ USERDATE    : Factor w/ 3773 levels "2005-08-17","2005-08-18",..: 1 2 2 3 4 5 6 7 8
##  $ USERDATE2   : Factor w/ 1405 levels "","2009-10-21",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ EXAMDATE    : Factor w/ 1146 levels "","2005-01-09",..: 4 5 5 6 7 8 9 10 12 11 ...
##  $ VSWEIGHT    : num  89 74 184 88 179 ...
##  $ VSWTUNIT    : int  2 2 1 2 1 1 2 1 1 2 ...
##  $ VSHEIGHT    : num  71 69 69 71 64 54 65 62 66 -4 ...
##  $ VSHTUNIT    : int  1 1 1 1 1 1 1 1 1 -4 ...
```

```
##  $ VSBPSYS     : int  110 148 140 150 160 128 124 144 139 150 ...
##  $ VSBPDIA     : int  80 70 70 80 75 80 74 90 80 70 ...
##  $ VSPULSE     : int  60 60 54 52 67 80 80 60 68 64 ...
##  $ VSRESP      : int  20 18 16 18 15 16 20 16 20 18 ...
##  $ VSTEMP      : num  96.3 97.9 97 96.7 97 98.2 96 97.9 98.6 97.5 ...
##  $ VSTMPSRC    : int  2 2 1 2 1 1 2 2 1 2 ...
##  $ VSTMPUNT    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ VSCOMM      : Factor w/ 1053 levels "","- REGULAR RHYTHM ON AUSCULTATION\n- Tempera
##  $ update_stamp: Factor w/ 3403 levels "2005-08-17 00:00:00.0",..: 1 2 2 3 4 5 6 7 8 8
```

```
summary(vitals) # Mostramos el resumen estadístico del dataset
```

```
##     Phase            ID              RID           SITEID          VISCODE
##  ADNI1 :5044   Min.   :     7   Min.   :   2   Min.   :  1.00   sc     :1984
##  ADNI2 :5756   1st Qu.:  2853   1st Qu.: 779   1st Qu.: 15.00   bl     :1551
##  ADNI3 :2918   Median :  6410   Median :2337   Median : 33.00   v01    :1119
##  ADNIGO: 737   Mean   : 18941   Mean   :2958   Mean   : 40.03   v11    : 944
##                3rd Qu.:  9958   3rd Qu.:4799   3rd Qu.: 52.00   m06    : 898
##                Max.   :143754   Max.   :7016   Max.   :127.00   v21    : 817
##                                                                 (Other):7142
##     VISCODE2          USERDATE          USERDATE2           EXAMDATE
##  sc     :3101   2006-11-16:   19             :11090             :9411
##  bl     :2333   2007-01-31:   18   2018-07-11:   15   2006-12-12:  20
##  m12    :1680   2006-05-02:   17   2018-05-31:   13   2006-12-05:  18
##  m06    :1617   2006-12-07:   17   2018-07-17:   12   2006-09-11:  17
##  m24    :1492   2007-02-21:   17   2018-06-11:   11   2007-02-22:  17
##  m36    : 857   2007-03-05:   17   2018-07-18:   11   2006-12-04:  16
##  (Other):3375   (Other)   :14350   (Other)   : 3303   (Other)   :4956
##     VSWEIGHT        VSWTUNIT         VSHEIGHT         VSHTUNIT
##  Min.   : -1.0   Min.   :-1.000   Min.   : -4.00   Min.   :-4.000
##  1st Qu.:121.9   1st Qu.: 1.000   1st Qu.: -4.00   1st Qu.:-4.000
##  Median :157.0   Median : 1.000   Median : -4.00   Median :-4.000
##  Mean   :150.1   Mean   : 1.183   Mean   : 25.44   Mean   :-1.403
##  3rd Qu.:183.5   3rd Qu.: 1.000   3rd Qu.: 63.00   3rd Qu.: 1.000
##  Max.   :493.0   Max.   : 2.000   Max.   :194.00   Max.   : 2.000
##  NA's   :51      NA's   :168      NA's   :1706     NA's   :6844
##     VSBPSYS         VSBPDIA          VSPULSE           VSRESP
##  Min.   : -1.0   Min.   : -1.00   Min.   : -1.00   Min.   :-1.00
##  1st Qu.:122.0   1st Qu.: 68.00   1st Qu.: 58.00   1st Qu.:15.00
##  Median :133.0   Median : 74.00   Median : 64.00   Median :16.00
##  Mean   :133.6   Mean   : 73.71   Mean   : 65.18   Mean   :16.31
##  3rd Qu.:144.0   3rd Qu.: 80.00   3rd Qu.: 72.00   3rd Qu.:18.00
##  Max.   :233.0   Max.   :132.00   Max.   :190.00   Max.   :48.00
```

```
##  NA's   :126     NA's   :127      NA's   :135      NA's   :182
##      VSTEMP          VSTMPSRC         VSTMPUNT
##  Min.   : -1.00   Min.   :-1.000   Min.   :-1.000
##  1st Qu.: 36.90   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 97.30   Median : 1.000   Median : 1.000
##  Mean   : 80.66   Mean   : 1.223   Mean   : 1.253
##  3rd Qu.: 97.90   3rd Qu.: 1.000   3rd Qu.: 2.000
##  Max.   :100.40   Max.   : 3.000   Max.   : 2.000
##  NA's   :95       NA's   :213      NA's   :212
##                                VSCOMM                     update_stamp
##  -4                               :10569   2012-11-16 11:42:52.0: 1280
##                                   : 2536   2012-11-16 11:42:53.0:  549
##  Vital signs taken by CRU nurse.:   87   2012-11-16 09:43:54.0:  542
##  temperature taken temporally   :   48   2012-11-16 11:42:51.0:  532
##  Vital signs taken by CRU nurse :   37   2021-07-29 04:20:37.0:  319
##  temporal                       :   24   2012-11-16 09:43:53.0:  153
##  (Other)                        : 1154   (Other)              :11080

dim(vitals) # Mostramos el número de variables y registros

## [1] 14455    22

# Historial médico:
str(medhist,list.len=ncol(medhist)) # Mostramos la estructura del dataset

## 'data.frame': 3083 obs. of  39 variables:
##  $ Phase       : Factor w/ 3 levels "ADNI1","ADNI2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ID          : int  8 10 12 14 16 18 20 22 24 26 ...
##  $ RID         : int  2 1 3 4 5 7 9 8 11 13 ...
##  $ SITEID      : int  107 10 107 10 107 10 10 107 107 10 ...
##  $ VISCODE     : Factor w/ 7 levels "f","m36","m48",..: 5 1 5 5 5 5 1 5 1 1 ...
##  $ VISCODE2    : Factor w/ 9 levels "f","m12","m24",..: 9 1 9 9 9 9 1 9 1 1 ...
##  $ USERDATE    : Factor w/ 1104 levels "2005-08-17","2005-08-18",..: 1 2 2 2 3 4 5 6 7
##  $ USERDATE2   : Factor w/ 413 levels "","2009-12-16",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ EXAMDATE    : Factor w/ 358 levels "","2005-08-17",..: 2 3 3 3 4 5 6 7 8 9 ...
##  $ MHSOURCE    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ MHPSYCH     : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ MH2NEURL    : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ MH3HEAD     : int  1 1 1 0 1 1 0 1 1 0 ...
##  $ MH4CARD     : int  0 0 1 1 0 1 0 0 1 0 ...
##  $ MH5RESP     : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ MH6HEPAT    : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ MH7DERM     : int  0 0 0 0 1 0 1 0 0 0 ...
```

```
##  $ MH8MUSCL    : int  0 0 1 1 0 0 1 1 1 0 ...
##  $ MH9ENDO     : int  1 0 1 0 0 1 0 0 0 0 ...
##  $ MH10GAST    : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ MH11HEMA    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH12RENA    : int  0 0 1 1 1 0 0 0 0 0 ...
##  $ MH13ALLE    : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ MH14ALCH    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH14AALCH   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH14BALCH   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH14CALCH   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH15DRUG    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH15ADRUG   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH15BDRUG   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH16SMOK    : int  0 0 1 1 1 1 0 1 1 1 ...
##  $ MH16ASMOK   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH16BSMOK   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH16CSMOK   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MH17MALI    : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ MH18SURG    : int  0 0 1 1 1 0 0 1 1 0 ...
##  $ MH19OTHR    : int  0 1 0 0 0 1 0 0 0 0 ...
##  $ MHCOMMEN    : Factor w/ 246 levels "-4","\"I love my good health\"",..: 1 1 1 1 1 1
##  $ update_stamp: Factor w/ 584 levels "2005-08-17 00:00:00.0",..: 1 2 2 2 3 4 5 6 7 8
```

```
summary(medhist) # Mostramos el resumen estadístico del dataset
```

```
##     Phase            ID             RID             SITEID       VISCODE
##  ADNI1 :1155   Min.   :   4   Min.   :   1.0   Min.   :  1.00   f  : 333
##  ADNI2 :1463   1st Qu.: 523   1st Qu.: 668.5   1st Qu.: 15.00   m36:   9
##  ADNIGO: 465   Median :1086   Median :1337.0   Median : 33.00   m48: 143
##                Mean   :1207   Mean   :2259.8   Mean   : 40.11   m60:  56
##                3rd Qu.:1858   3rd Qu.:4356.5   3rd Qu.: 50.00   sc :1079
##                Max.   :2934   Max.   :5296.0   Max.   :127.00   v01:1079
##                                                                 v06: 384
##     VISCODE2          USERDATE         USERDATE2          EXAMDATE
##  sc     :2158   2006-05-22:  11             :2438             :1928
##  f      : 333   2006-06-20:  11   2012-03-02:   7   2006-03-13:  11
##  m60    : 213   2006-12-07:  11   2013-02-05:   7   2006-05-10:  10
##  m48    : 163   2012-01-11:  11   2012-06-25:   6   2006-03-16:   9
##  m12    : 112   2006-03-28:   9   2012-07-12:   6   2006-05-04:   9
##  m72    :  85   2006-05-11:   9   2011-11-08:   5   2006-05-11:   9
##  (Other):  19   (Other)   :3021   (Other)   : 614   (Other)   :1107
##     MHSOURCE        MHPSYCH          MH2NEURL         MH3HEAD
##  Min.   :1.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
```

```
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :1.000   Median :0.0000   Median :0.0000   Median :1.000
##  Mean   :1.007   Mean   :0.3506   Mean   :0.3146   Mean   :0.628
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :2.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##
##     MH4CARD          MH5RESP          MH6HEPAT          MH7DERM
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.00000   Median :0.0000
##  Mean   :0.6643   Mean   :0.2222   Mean   :0.03892   Mean   :0.3185
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##
##     MH8MUSCL         MH9ENDO          MH10GAST         MH11HEMA
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :-4.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.00000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median : 0.00000
##  Mean   :0.6695   Mean   :0.4236   Mean   :0.4453   Mean   : 0.09471
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   : 1.00000
##
##     MH12RENA        MH13ALLE         MH14ALCH         MH14AALCH
##  Min.   :0.000   Min.   :0.0000   Min.   :0.00000   Min.   : 0.000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 3.000
##  Median :0.000   Median :0.0000   Median :0.00000   Median : 5.000
##  Mean   :0.434   Mean   :0.4194   Mean   :0.04411   Mean   : 6.344
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.: 8.000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.00000   Max.   :25.000
##                                                     NA's   :2990
##    MH14BALCH        MH14CALCH         MH15DRUG          MH15ADRUG
##  Min.   :-4.000   Min.   :-4.000   Min.   :0.000000   Min.   :-4.000
##  1st Qu.:-4.000   1st Qu.:-4.000   1st Qu.:0.000000   1st Qu.:-4.000
##  Median :-4.000   Median :-4.000   Median :0.000000   Median :-4.000
##  Mean   :-3.016   Mean   :-2.791   Mean   :0.009731   Mean   :-3.783
##  3rd Qu.:-4.000   3rd Qu.:-4.000   3rd Qu.:0.000000   3rd Qu.:-4.000
##  Max.   :51.000   Max.   :46.000   Max.   :1.000000   Max.   :40.000
##  NA's   :1155     NA's   :1155                        NA's   :1620
##    MH15BDRUG        MH16SMOK         MH16ASMOK         MH16BSMOK
##  Min.   :-4.000   Min.   :0.0000   Min.   :-4.000   Min.   :-4.000
##  1st Qu.:-4.000   1st Qu.:0.0000   1st Qu.:-4.000   1st Qu.:-4.000
##  Median :-4.000   Median :0.0000   Median :-4.000   Median :-4.000
```

```
##  Mean   :-3.606   Mean   :0.3983   Mean   :-1.861   Mean   : 6.473
##  3rd Qu.:-4.000   3rd Qu.:1.0000   3rd Qu.: 1.000   3rd Qu.:15.000
##  Max.   :50.000   Max.   :1.0000   Max.   :10.000   Max.   :66.000
##  NA's   :1620                      NA's   :1155     NA's   :1155
##    MH16CSMOK         MH17MALI         MH18SURG         MH19OTHR
##  Min.   :-4.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-4.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :-4.000   Median :0.0000   Median :1.0000   Median :0.0000
##  Mean   : 9.392   Mean   :0.2601   Mean   :0.7486   Mean   :0.2138
##  3rd Qu.:26.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :68.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  NA's   :1155
##                              MHCOMMEN                    update_stamp
##  -4                              :2813   2012-11-16 12:24:57.0:1033
##  None                            :  14   2012-11-16 09:41:02.0: 454
##  Fall                            :   4   2012-11-16 12:24:56.0:  77
##  none                            :   4   2012-11-16 12:24:58.0:  40
##  Allergic to tetracyclin (rash):   2   2013-04-09 19:17:29.0:  15
##  Generally healthy.              :   2   2006-05-22 00:00:00.0:  11
##  (Other)                         : 244   (Other)              :1453
```

```
dim(medhist) # Mostramos el número de variables y registros
```

```
## [1] 3083   39
```

```
# CDR (Clinical Dementia Rate):
str(cdr,list.len=ncol(cdr)) # Mostramos la estructura del dataset
```

```
## 'data.frame': 12414 obs. of  21 variables:
##  $ Phase       : Factor w/ 4 levels "ADNI1","ADNI2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ID          : int  8 10 12 14 16 18 20 22 24 26 ...
##  $ RID         : int  2 3 4 5 7 9 8 11 13 14 ...
##  $ SITEID      : int  107 107 10 107 10 10 107 107 10 10 ...
##  $ VISCODE     : Factor w/ 24 levels "f","init","m06",..: 11 11 11 11 11 1 11 1 1 11 .
##  $ VISCODE2    : Factor w/ 35 levels "","f","m06","m102",..: 34 34 34 34 34 2 34 2 2 3
##  $ USERDATE    : Factor w/ 3744 levels "2005-08-17","2005-08-18",..: 1 2 2 3 4 5 6 7 8
##  $ USERDATE2   : Factor w/ 1508 levels "","2009-10-05",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ EXAMDATE    : Factor w/ 1132 levels "","2005-08-17",..: 2 3 3 4 5 6 7 8 9 10 ...
##  $ CDSOURCE    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ CDVERSION   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ CDMEMORY    : num  0 1 0.5 0 1 0.5 0 0.5 0 0 ...
##  $ CDORIENT    : num  0 1 0 0 1 1 0 0 0 0 ...
##  $ CDJUDGE     : num  0 1 0.5 0 1 1 0 0.5 0 0 ...
```

```
##  $ CDCOMMUN    : num  0 1 0 0 1 1 0 0 0 0 ...
##  $ CDHOME      : num  0 0.5 0 0 1 1 0 0 0 0 ...
##  $ CDCARE      : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ CDGLOBAL    : num  0 1 0.5 0 1 1 0 0.5 0 0 ...
##  $ CDRSB       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CDSOB       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ update_stamp: Factor w/ 1995 levels "2009-10-05 00:00:00.0",..: 1 1 1 1 1 1 1 1 1 1
```

```
summary(cdr) # Mostramos el resumen estadístico del dataset
```

```
##     Phase            ID              RID           SITEID         VISCODE
##  ADNI1 :4205   Min.   :     3   Min.   :   2   Min.   :  1.0   sc     :1999
##  ADNI2 :5075   1st Qu.:  2484   1st Qu.: 746   1st Qu.: 15.0   v01    :1113
##  ADNI3 :2538   Median :  5544   Median :2238   Median : 33.0   v11    : 962
##  ADNIGO: 596   Mean   : 19250   Mean   :2860   Mean   : 39.6   m06    : 897
##                3rd Qu.:  8770   3rd Qu.:4713   3rd Qu.: 52.0   v21    : 850
##                Max.   :144070   Max.   :7020   Max.   :127.0   m12    : 740
##                                                                (Other):5853
##     VISCODE2          USERDATE          USERDATE2          EXAMDATE
##  sc     :3110   2006-12-07:   16             :5366             :8209
##  m12    :1700   2007-02-13:   15   2009-10-05:3827   2007-02-22:  18
##  m06    :1616   2007-02-22:   15   2018-07-11:  12   2006-12-05:  15
##  m24    :1543   2013-02-13:   15   2018-02-15:  10   2006-03-13:  13
##  m36    : 868   2006-05-22:   14   2018-05-31:   9   2006-10-18:  12
##  m48    : 695   2007-02-26:   14   2018-07-17:   9   2006-11-06:  12
##  (Other):2882   (Other)   :12325   (Other)   :3181   (Other)   :4135
##     CDSOURCE        CDVERSION        CDMEMORY          CDORIENT
##  Min.   :-1.000   Min.   :1.000   Min.   :-1.0000   Min.   :-1.0000
##  1st Qu.: 1.000   1st Qu.:1.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 1.000   Median :1.000   Median : 0.5000   Median : 0.0000
##  Mean   : 1.084   Mean   :1.081   Mean   : 0.5038   Mean   : 0.3342
##  3rd Qu.: 1.000   3rd Qu.:1.000   3rd Qu.: 1.0000   3rd Qu.: 0.5000
##  Max.   : 2.000   Max.   :3.000   Max.   : 3.0000   Max.   : 3.0000
##  NA's   :109      NA's   :4918    NA's   :33        NA's   :33
##     CDJUDGE          CDCOMMUN          CDHOME           CDCARE
##  Min.   :-1.0000   Min.   :-1.0000   Min.   :-1.0000   Min.   :-1.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.5000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.3773   Mean   : 0.2753   Mean   : 0.3155   Mean   : 0.1247
##  3rd Qu.: 0.5000   3rd Qu.: 0.5000   3rd Qu.: 0.5000   3rd Qu.: 0.0000
##  Max.   : 3.0000   Max.   : 3.0000   Max.   : 3.0000   Max.   : 3.0000
##  NA's   :33        NA's   :33        NA's   :33        NA's   :34
##     CDGLOBAL          CDRSB            CDSOB
```

```
##  Min.   :-1.0000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.5000   Median : 0.500   Median : 0.500
##  Mean   : 0.4249   Mean   : 1.567   Mean   : 1.565
##  3rd Qu.: 0.5000   3rd Qu.: 2.000   3rd Qu.: 2.000
##  Max.   : 3.0000   Max.   :18.000   Max.   :18.000
##  NA's   :35        NA's   :9910     NA's   :9910
##                 update_stamp
##  2009-10-05 00:00:00.0:3827
##  2012-11-16 12:23:27.0: 946
##  2012-11-16 12:23:28.0: 872
##  2012-11-16 09:39:34.0: 569
##  2018-10-24 22:44:39.0: 508
##  2021-07-29 04:20:49.0: 362
##  (Other)              :5330
```

```
dim(cdr) # Mostramos el número de variables y registros
```

```
## [1] 12414    21
```

```
#=====================================#
# Limpieza y preparación de los datos: #
#=====================================#

# Signos vitales:

vitals<-subset(vitals,vitals$Phase=="ADNI1")
# Seleccionamos las vatiables que nos interesan
vitals<-vitals[,c(-1,-3,-4,-5,-6,-7,-8,-9,-11,-13,-22,-21)]
vitals<-na.omit(vitals) # Eliminamos los datos perdidos (NA)
# Transformamos a centímetros aquellas medidas de altura que se encuentren en
# pulgadas
vitals$VSHEIGHT[vitals$VSHTUNIT==1]<-round(
vitals$VSHEIGHT[vitals$VSHTUNIT==1]*2.54,digits = 1)
# Transformamos a kilogramos aquellas medidas de peso que se encuentren en
# libras
vitals$VSWEIGHT[vitals$VSWTUNIT==1]<-round(
vitals$VSWEIGHT[vitals$VSWTUNIT==1]*0.454,digits = 1)
dim(vitals) # Mostramos el número de variables y registros
```

```
## [1] 5044   10
```

```
# Historial médico:
```

```
medhist<-subset(medhist,medhist$Phase=="ADNI1")
# Seleccionamos las vatiables que nos interesan
medhist<-medhist[,c(2,11,12,13,14,15,16,17,18,19,20,21,22,23,24,28,31)]
medhist<-na.omit(medhist) # Eliminamos los datos perdidos (NA)
dim(medhist) # Mostramos el número de variables y registros

## [1] 1155   17

# CDR (Clinical Dementia Rate):

cdr<-subset(cdr,cdr$Phase=="ADNI1")
cdr<-subset(cdr,cdr$CDGLOBAL!=-1)
# Seleccionamos las vatiables que nos interesan
cdr<-cdr[,c(2,18)]
cdr<-na.omit(cdr) # Eliminamos los datos perdidos (NA)
dim(cdr) # Mostramos el número de variables y registros

## [1] 4158    2

# Fusión de los tres datasets en uno único mediante la variable ID (ID del
# paciente):

vitals_medhist<-merge(vitals,medhist,by="ID")
cdr_vitals_medhist<-merge(cdr,vitals_medhist,by="ID")
cdr_vitals_medhist<-cdr_vitals_medhist[,-1]
str(cdr_vitals_medhist)

## 'data.frame': 1138 obs. of  26 variables:
##  $ CDGLOBAL: num  0 1 1 0 0.5 0 0 0 1 0 ...
##  $ VSWEIGHT: num  89 74 184 88 179 ...
##  $ VSHEIGHT: num  71 69 69 71 64 54 65 62 66 -4 ...
##  $ VSBPSYS : int  110 148 140 150 160 128 124 144 139 150 ...
##  $ VSBPDIA : int  80 70 70 80 75 80 74 90 80 70 ...
##  $ VSPULSE : int  60 60 54 52 67 80 80 60 68 64 ...
##  $ VSRESP  : int  20 18 16 18 15 16 20 16 20 18 ...
##  $ VSTEMP  : num  96.3 97.9 97 96.7 97 98.2 96 97.9 98.6 97.5 ...
##  $ VSTMPSRC: int  2 2 1 2 1 1 2 2 1 2 ...
##  $ VSTMPUNT: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ MHPSYCH : int  0 0 1 0 0 0 0 1 0 0 ...
##  $ MH2NEURL: int  0 0 0 1 0 0 0 0 0 0 ...
##  $ MH3HEAD : int  0 1 1 0 1 1 0 0 1 1 ...
##  $ MH4CARD : int  1 0 1 0 0 1 0 0 1 1 ...
##  $ MH5RESP : int  0 0 0 0 0 1 0 0 1 0 ...
```

```
##  $ MH6HEPAT: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH7DERM : int  0 1 0 1 0 0 0 0 1 0 ...
##  $ MH8MUSCL: int  1 0 0 1 1 1 0 0 0 0 ...
##  $ MH9ENDO : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ MH10GAST: int  0 1 0 0 0 0 0 0 0 0 ...
##  $ MH11HEMA: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH12RENA: int  1 1 0 0 0 0 0 0 1 1 ...
##  $ MH13ALLE: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH14ALCH: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH15DRUG: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH16SMOK: int  1 1 1 0 1 1 1 0 1 0 ...

#=======================#
# Análisis de regresión #
#=======================#

# Creamos el modelo de regresión:
cdr_vitals_medhist_lm<-lm(CDGLOBAL ~. ,cdr_vitals_medhist)
# Visualizamos los resultados estadísticos:
summary(cdr_vitals_medhist_lm)

##
## Call:
## lm(formula = CDGLOBAL ~ ., data = cdr_vitals_medhist)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73009 -0.33582  0.06163  0.14348  1.55774
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.582e-01  1.376e-01   4.056 5.34e-05 ***
## VSWEIGHT    -9.848e-05  2.290e-04  -0.430 0.667242
## VSHEIGHT    -3.145e-04  1.933e-04  -1.627 0.104087
## VSBPSYS      8.911e-04  6.230e-04   1.430 0.152889
## VSBPDIA     -2.150e-03  1.138e-03  -1.889 0.059202 .
## VSPULSE      4.506e-04  9.299e-04   0.485 0.628059
## VSRESP      -4.273e-03  3.130e-03  -1.365 0.172536
## VSTEMP      -1.828e-04  8.770e-04  -0.208 0.834954
## VSTMPSRC    -1.871e-02  2.424e-02  -0.772 0.440264
## VSTMPUNT    -4.457e-02  5.163e-02  -0.863 0.388210
## MHPSYCH      1.886e-02  2.128e-02   0.886 0.375680
## MH2NEURL     1.741e-02  2.211e-02   0.788 0.431110
```

```
## MH3HEAD      7.278e-02  2.094e-02   3.476 0.000529 ***
## MH4CARD     -1.024e-02  2.150e-02  -0.476 0.633942
## MH5RESP      7.226e-02  2.485e-02   2.908 0.003704 **
## MH6HEPAT    -3.982e-02  6.011e-02  -0.662 0.507816
## MH7DERM      2.572e-02  2.268e-02   1.134 0.256907
## MH8MUSCL    -4.494e-03  2.145e-02  -0.210 0.834059
## MH9ENDO     -2.678e-02  2.049e-02  -1.307 0.191417
## MH10GAST    -2.112e-02  2.123e-02  -0.995 0.320076
## MH11HEMA    -9.796e-03  3.432e-02  -0.285 0.775378
## MH12RENA     3.071e-02  2.054e-02   1.495 0.135163
## MH13ALLE    -3.008e-02  2.081e-02  -1.446 0.148489
## MH14ALCH     9.315e-02  5.442e-02   1.712 0.087262 .
## MH15DRUG     1.399e-01  1.709e-01   0.819 0.413137
## MH16SMOK    -9.305e-03  2.086e-02  -0.446 0.655632
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3289 on 1112 degrees of freedom
## Multiple R-squared:  0.04494,Adjusted R-squared:  0.02347
## F-statistic: 2.093 on 25 and 1112 DF,  p-value: 0.001324

# Realizamos una ANOVA:
anova(cdr_vitals_medhist_lm)

## Analysis of Variance Table
##
## Response: CDGLOBAL
##             Df  Sum Sq Mean Sq F value   Pr(>F)
## VSWEIGHT     1   0.006 0.00552  0.0510  0.82132
## VSHEIGHT     1   0.299 0.29906  2.7648  0.09664 .
## VSBPSYS      1   0.000 0.00025  0.0023  0.96142
## VSBPDIA      1   0.371 0.37116  3.4314  0.06423 .
## VSPULSE      1   0.006 0.00572  0.0529  0.81813
## VSRESP       1   0.308 0.30765  2.8443  0.09198 .
## VSTEMP       1   0.100 0.10032  0.9275  0.33573
## VSTMPSRC     1   0.137 0.13719  1.2683  0.26032
## VSTMPUNT     1   0.063 0.06289  0.5814  0.44591
## MHPSYCH      1   0.089 0.08903  0.8231  0.36447
## MH2NEURL     1   0.184 0.18435  1.7043  0.19199
## MH3HEAD      1   1.654 1.65403 15.2917 9.77e-05 ***
## MH4CARD      1   0.014 0.01389  0.1284  0.72014
## MH5RESP      1   0.908 0.90839  8.3982  0.00383 **
## MH6HEPAT     1   0.047 0.04672  0.4319  0.51119
```

```
## MH7DERM      1   0.134 0.13394  1.2383  0.26605
## MH8MUSCL     1   0.032 0.03165  0.2926  0.58867
## MH9ENDO      1   0.246 0.24627  2.2768  0.13161
## MH10GAST     1   0.115 0.11524  1.0654  0.30222
## MH11HEMA     1   0.013 0.01254  0.1160  0.73350
## MH12RENA     1   0.195 0.19476  1.8006  0.17992
## MH13ALLE     1   0.254 0.25424  2.3505  0.12553
## MH14ALCH     1   0.391 0.39134  3.6180  0.05742 .
## MH15DRUG     1   0.072 0.07235  0.6689  0.41362
## MH16SMOK     1   0.022 0.02152  0.1990  0.65563
## Residuals 1112 120.279 0.10816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#~~~~~~~~~~~~~~#
# Predicción 1 #
#~~~~~~~~~~~~~~#

#==========================#
# Preparación de los datos #
#==========================#

# Extraemos los factores del riesgo que hemos obtenido en el análisis del
# dataset 1 para crear un nuevo dataset:

adni_dian<-adni_dian[,c(1,4,5,6,7,10,11,13)]
adni_dian_pred<-adni_dian[,c(2,3,4,5,6,7,8)]
str(adni_dian_pred)

## 'data.frame': 2831 obs. of  7 variables:
##  $ visitage: num  76.3 77.3 78.4 79.4 72.3 ...
##  $ gender  : num  2 2 2 2 1 1 1 1 1 1 ...
##  $ EDUC    : int  18 18 18 18 17 17 17 17 17 18 ...
##  $ HISPANIC: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MARISTAT: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ HANDED  : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ DADDEM  : int  0 0 0 0 0 0 0 0 0 0 ...

# Separamos los datos en dos grupos:
smp_size<-floor(0.67 * nrow(adni_dian_pred))
set.seed(123)
train_ind<-sample(seq_len(nrow(adni_dian_pred)),size = smp_size)
```

```
train_sample<-adni_dian_pred[train_ind, ] # Datos de entrenamiento 1
test_sample<-adni_dian_pred[-train_ind, ] # Datos de prueba 1

# Etiquetas de clase de los datos de entrenamiento:
train_sample_labels<-adni_dian[train_ind,1]

# Etiquetas de clase de los datos de prueba:
test_sample_labels<-adni_dian[-train_ind,1]

# Comprobamos que los datos se hayan dividido correctamente observando el
# número de registros:
dim(train_sample)

## [1] 1896    7

dim(test_sample)

## [1] 935   7

# Normalización de los datos:
normalize <- function(x) {
return((x - min(x)) / (max(x) - min(x)))
}
adni_dian_norm<-as.data.frame(lapply(adni_dian_pred,normalize))
str(adni_dian_norm)

## 'data.frame': 2831 obs. of  7 variables:
##  $ visitage: num  0.54 0.566 0.593 0.618 0.438 ...
##  $ gender  : num  1 1 1 1 0 0 0 0 0 0 ...
##  $ EDUC    : num  0.833 0.833 0.833 0.833 0.75 ...
##  $ HISPANIC: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ MARISTAT: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ HANDED  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ DADDEM  : num  0 0 0 0 0 0 0 0 0 0 ...

#=======================#
# Modelos de predicción #
#=======================#

#+++++++++++++++++#
# Algoritmo k-NN #
#+++++++++++++++++#
```

```r
# Entrenamiento del modelo #

library(class) # Cargamos el paquete "class"

# Utilizamos la función "knn" con diferentes valores de k:
# k = 1
knn_pred1<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=1)
# k = 3
knn_pred3<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=3)
# k = 5
knn_pred5<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=5)
# k = 7
knn_pred7<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=7)
# k = 11
knn_pred11<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,
k=11)
# k = 21
knn_pred21<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,
k=21)

# Evaluación del rendimiento del modelo #

library(gmodels) # Cargamos el paquete "gmodels"

# Utilizamos la función "CrossTable" para comprobar el funcionamiento de cada
# modelo:
CrossTable(x=test_sample_labels,y=knn_pred1,prop.chisq = FALSE) # k = 1
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                    | knn_pred1
```

```
## test_sample_labels |         0 |       0.5 |         1 |         2 |         3 | Row T
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  0 |        20 |        55 |        11 |         2 |         1 |
##                    |     0.225 |     0.618 |     0.124 |     0.022 |     0.011 |      0
##                    |     0.206 |     0.090 |     0.057 |     0.059 |     0.250 |
##                    |     0.021 |     0.059 |     0.012 |     0.002 |     0.001 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                0.5 |        63 |       421 |       107 |        13 |         2 |
##                    |     0.104 |     0.695 |     0.177 |     0.021 |     0.003 |      0
##                    |     0.649 |     0.692 |     0.557 |     0.382 |     0.500 |
##                    |     0.067 |     0.450 |     0.114 |     0.014 |     0.002 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  1 |        11 |       117 |        61 |        16 |         1 |
##                    |     0.053 |     0.568 |     0.296 |     0.078 |     0.005 |      0
##                    |     0.113 |     0.192 |     0.318 |     0.471 |     0.250 |
##                    |     0.012 |     0.125 |     0.065 |     0.017 |     0.001 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  2 |         2 |        14 |        12 |         2 |         0 |
##                    |     0.067 |     0.467 |     0.400 |     0.067 |     0.000 |      0
##                    |     0.021 |     0.023 |     0.062 |     0.059 |     0.000 |
##                    |     0.002 |     0.015 |     0.013 |     0.002 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  3 |         1 |         1 |         1 |         1 |         0 |
##                    |     0.250 |     0.250 |     0.250 |     0.250 |     0.000 |      0
##                    |     0.010 |     0.002 |     0.005 |     0.029 |     0.000 |
##                    |     0.001 |     0.001 |     0.001 |     0.001 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##       Column Total |        97 |       608 |       192 |        34 |         4 |
##                    |     0.104 |     0.650 |     0.205 |     0.036 |     0.004 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##
##
```

```
CrossTable(x=test_sample_labels,y=knn_pred3,prop.chisq = FALSE) # k = 3
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
```

```
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                      | knn_pred3
## test_sample_labels |         0 |       0.5 |         1 |         2 |         3 | Row T
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  0 |        14 |        64 |        10 |         1 |         0 |
##                    |     0.157 |     0.719 |     0.112 |     0.011 |     0.000 |      0
##                    |     0.215 |     0.092 |     0.065 |     0.048 |     0.000 |
##                    |     0.015 |     0.068 |     0.011 |     0.001 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                0.5 |        40 |       470 |        88 |         7 |         1 |
##                    |     0.066 |     0.776 |     0.145 |     0.012 |     0.002 |      0
##                    |     0.615 |     0.677 |     0.571 |     0.333 |     1.000 |
##                    |     0.043 |     0.503 |     0.094 |     0.007 |     0.001 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  1 |        10 |       136 |        50 |        10 |         0 |
##                    |     0.049 |     0.660 |     0.243 |     0.049 |     0.000 |      0
##                    |     0.154 |     0.196 |     0.325 |     0.476 |     0.000 |
##                    |     0.011 |     0.145 |     0.053 |     0.011 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  2 |         1 |        21 |         6 |         2 |         0 |
##                    |     0.033 |     0.700 |     0.200 |     0.067 |     0.000 |      0
##                    |     0.015 |     0.030 |     0.039 |     0.095 |     0.000 |
##                    |     0.001 |     0.022 |     0.006 |     0.002 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##                  3 |         0 |         3 |         0 |         1 |         0 |
##                    |     0.000 |     0.750 |     0.000 |     0.250 |     0.000 |      0
##                    |     0.000 |     0.004 |     0.000 |     0.048 |     0.000 |
##                    |     0.000 |     0.003 |     0.000 |     0.001 |     0.000 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##       Column Total |        65 |       694 |       154 |        21 |         1 |
##                    |     0.070 |     0.742 |     0.165 |     0.022 |     0.001 |
## -------------------|-----------|-----------|-----------|-----------|-----------|------
##
##

CrossTable(x=test_sample_labels,y=knn_pred5,prop.chisq = FALSE) # k = 5

##
```

```
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                     | knn_pred5
## test_sample_labels |         0 |       0.5 |         1 |         2 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                  0 |        15 |        62 |        12 |         0 |        89 |
##                    |     0.169 |     0.697 |     0.135 |     0.000 |     0.095 |
##                    |     0.326 |     0.083 |     0.094 |     0.000 |           |
##                    |     0.016 |     0.066 |     0.013 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                0.5 |        20 |       520 |        64 |         2 |       606 |
##                    |     0.033 |     0.858 |     0.106 |     0.003 |     0.648 |
##                    |     0.435 |     0.692 |     0.500 |     0.200 |           |
##                    |     0.021 |     0.556 |     0.068 |     0.002 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                  1 |         9 |       145 |        45 |         7 |       206 |
##                    |     0.044 |     0.704 |     0.218 |     0.034 |     0.220 |
##                    |     0.196 |     0.193 |     0.352 |     0.700 |           |
##                    |     0.010 |     0.155 |     0.048 |     0.007 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                  2 |         1 |        22 |         7 |         0 |        30 |
##                    |     0.033 |     0.733 |     0.233 |     0.000 |     0.032 |
##                    |     0.022 |     0.029 |     0.055 |     0.000 |           |
##                    |     0.001 |     0.024 |     0.007 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                  3 |         1 |         2 |         0 |         1 |         4 |
##                    |     0.250 |     0.500 |     0.000 |     0.250 |     0.004 |
##                    |     0.022 |     0.003 |     0.000 |     0.100 |           |
##                    |     0.001 |     0.002 |     0.000 |     0.001 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##       Column Total |        46 |       751 |       128 |        10 |       935 |
```

```
##                     |     0.049 |     0.803 |     0.137 |     0.011 |           |
## --------------------|-----------|-----------|-----------|-----------|-----------|
##
##

CrossTable(x=test_sample_labels,y=knn_pred7,prop.chisq = FALSE) # k = 7

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                     | knn_pred7
## test_sample_labels |         0 |       0.5 |         1 |         2 | Row Total |
## --------------------|-----------|-----------|-----------|-----------|-----------|
##                   0 |        11 |        66 |        11 |         1 |        89 |
##                     |     0.124 |     0.742 |     0.124 |     0.011 |     0.095 |
##                     |     0.355 |     0.084 |     0.092 |     1.000 |           |
##                     |     0.012 |     0.071 |     0.012 |     0.001 |           |
## --------------------|-----------|-----------|-----------|-----------|-----------|
##                 0.5 |        14 |       529 |        63 |         0 |       606 |
##                     |     0.023 |     0.873 |     0.104 |     0.000 |     0.648 |
##                     |     0.452 |     0.675 |     0.529 |     0.000 |           |
##                     |     0.015 |     0.566 |     0.067 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|-----------|
##                   1 |         5 |       161 |        40 |         0 |       206 |
##                     |     0.024 |     0.782 |     0.194 |     0.000 |     0.220 |
##                     |     0.161 |     0.205 |     0.336 |     0.000 |           |
##                     |     0.005 |     0.172 |     0.043 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|-----------|
##                   2 |         1 |        25 |         4 |         0 |        30 |
##                     |     0.033 |     0.833 |     0.133 |     0.000 |     0.032 |
##                     |     0.032 |     0.032 |     0.034 |     0.000 |           |
##                     |     0.001 |     0.027 |     0.004 |     0.000 |           |
```

```
## -------------------|-----------|-----------|-----------|-----------|-----------|
##                  3 |         0 |         3 |         1 |         0 |         4 |
##                    |     0.000 |     0.750 |     0.250 |     0.000 |     0.004 |
##                    |     0.000 |     0.004 |     0.008 |     0.000 |           |
##                    |     0.000 |     0.003 |     0.001 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##       Column Total |        31 |       784 |       119 |         1 |       935 |
##                    |     0.033 |     0.839 |     0.127 |     0.001 |           |
## -------------------|-----------|-----------|-----------|-----------|-----------|
##
##
```

```
CrossTable(x=test_sample_labels,y=knn_pred11,prop.chisq = FALSE) # k = 11
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                     | knn_pred11
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|
##                  0 |         7 |        74 |         8 |        89 |
##                    |     0.079 |     0.831 |     0.090 |     0.095 |
##                    |     0.438 |     0.089 |     0.088 |           |
##                    |     0.007 |     0.079 |     0.009 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                0.5 |         7 |       552 |        47 |       606 |
##                    |     0.012 |     0.911 |     0.078 |     0.648 |
##                    |     0.438 |     0.667 |     0.516 |           |
##                    |     0.007 |     0.590 |     0.050 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |         1 |       174 |        31 |       206 |
##                    |     0.005 |     0.845 |     0.150 |     0.220 |
```

```
##                     |     0.062 |     0.210 |     0.341 |           |
##                     |     0.001 |     0.186 |     0.033 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   2 |         1 |        25 |         4 |        30 |
##                     |     0.033 |     0.833 |     0.133 |     0.032 |
##                     |     0.062 |     0.030 |     0.044 |           |
##                     |     0.001 |     0.027 |     0.004 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   3 |         0 |         3 |         1 |         4 |
##                     |     0.000 |     0.750 |     0.250 |     0.004 |
##                     |     0.000 |     0.004 |     0.011 |           |
##                     |     0.000 |     0.003 |     0.001 |           |
## --------------------|-----------|-----------|-----------|-----------|
##        Column Total |        16 |       828 |        91 |       935 |
##                     |     0.017 |     0.886 |     0.097 |           |
## --------------------|-----------|-----------|-----------|-----------|
##
##
```

```
CrossTable(x=test_sample_labels,y=knn_pred21,prop.chisq = FALSE) # k = 13
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  935
##
##
##                     | knn_pred21
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## --------------------|-----------|-----------|-----------|-----------|
##                   0 |         4 |        80 |         5 |        89 |
##                     |     0.045 |     0.899 |     0.056 |     0.095 |
##                     |     0.500 |     0.090 |     0.119 |           |
##                     |     0.004 |     0.086 |     0.005 |           |
## --------------------|-----------|-----------|-----------|-----------|
```

```
##                 0.5 |         2 |       589 |        15 |       606 |
##                     |     0.003 |     0.972 |     0.025 |     0.648 |
##                     |     0.250 |     0.666 |     0.357 |           |
##                     |     0.002 |     0.630 |     0.016 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   1 |         1 |       186 |        19 |       206 |
##                     |     0.005 |     0.903 |     0.092 |     0.220 |
##                     |     0.125 |     0.210 |     0.452 |           |
##                     |     0.001 |     0.199 |     0.020 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   2 |         1 |        27 |         2 |        30 |
##                     |     0.033 |     0.900 |     0.067 |     0.032 |
##                     |     0.125 |     0.031 |     0.048 |           |
##                     |     0.001 |     0.029 |     0.002 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   3 |         0 |         3 |         1 |         4 |
##                     |     0.000 |     0.750 |     0.250 |     0.004 |
##                     |     0.000 |     0.003 |     0.024 |           |
##                     |     0.000 |     0.003 |     0.001 |           |
## --------------------|-----------|-----------|-----------|-----------|
##       Column Total |         8 |       885 |        42 |       935 |
##                     |     0.009 |     0.947 |     0.045 |           |
## --------------------|-----------|-----------|-----------|-----------|
##
##
```

```
library(caret) # Cargamos el paquete "caret"
```

```
## Loading required package: lattice
## Loading required package: ggplot2
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
had status 1
```

```
knn_res1<-table(knn_pred1,test_sample_labels)
knn_res3<-table(knn_pred3,test_sample_labels)
knn_res5<-table(knn_pred5,test_sample_labels)
knn_res7<-table(knn_pred7,test_sample_labels)
knn_res11<-table(knn_pred11,test_sample_labels)
knn_res21<-table(knn_pred21,test_sample_labels)
knn_cmatrix1 <- confusionMatrix(knn_res1)
knn_cmatrix1
```

```
## Confusion Matrix and Statistics
```

```
##
##          test_sample_labels
## knn_pred1   0 0.5   1   2   3
##       0    20  63  11   2   1
##       0.5  55 421 117  14   1
##       1    11 107  61  12   1
##       2     2  13  16   2   1
##       3     1   2   1   0   0
##
## Overall Statistics
##
##                Accuracy : 0.539
##                  95% CI : (0.5065, 0.5714)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 1.000
##
##                   Kappa : 0.1173
##
##  Mcnemar's Test P-Value : 0.983
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.22472     0.6947  0.29612 0.066667 0.000000
## Specificity           0.90898     0.4316  0.82030 0.964641 0.995704
## Pos Pred Value        0.20619     0.6924  0.31771 0.058824 0.000000
## Neg Pred Value        0.91766     0.4343  0.80485 0.968923 0.995704
## Prevalence            0.09519     0.6481  0.22032 0.032086 0.004278
## Detection Rate        0.02139     0.4503  0.06524 0.002139 0.000000
## Detection Prevalence  0.10374     0.6503  0.20535 0.036364 0.004278
## Balanced Accuracy     0.56685     0.5632  0.55821 0.515654 0.497852
```

```
knn_cmatrix3 <- confusionMatrix(knn_res3)
knn_cmatrix3
```

```
## Confusion Matrix and Statistics
##
##          test_sample_labels
## knn_pred3   0 0.5   1   2   3
##       0    14  40  10   1   0
##       0.5  64 470 136  21   3
##       1    10  88  50   6   0
##       2     1   7  10   2   1
```

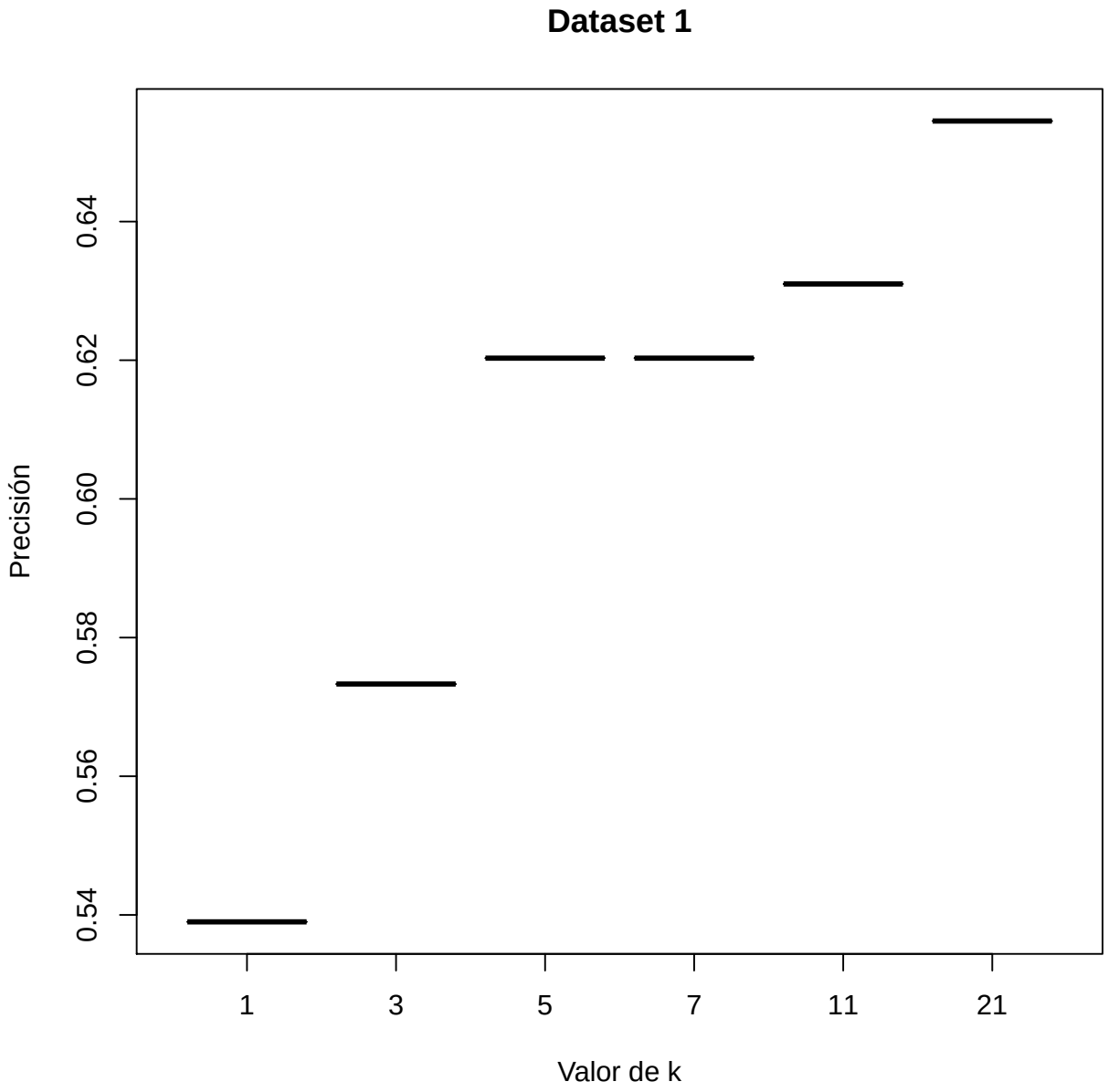
```
##       3     0   1   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5733
##                  95% CI : (0.5408, 0.6052)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1022
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.15730     0.7756  0.24272 0.066667 0.000000
## Specificity           0.93972     0.3191  0.85734 0.979006 0.998926
## Pos Pred Value        0.21538     0.6772  0.32468 0.095238 0.000000
## Neg Pred Value        0.91379     0.4357  0.80026 0.969365 0.995717
## Prevalence            0.09519     0.6481  0.22032 0.032086 0.004278
## Detection Rate        0.01497     0.5027  0.05348 0.002139 0.000000
## Detection Prevalence  0.06952     0.7422  0.16471 0.022460 0.001070
## Balanced Accuracy     0.54851     0.5474  0.55003 0.522836 0.499463

knn_cmatrix5 <- confusionMatrix(knn_res5)
knn_cmatrix5

## Confusion Matrix and Statistics
##
##          test_sample_labels
## knn_pred5   0 0.5   1   2   3
##       0    15  20   9   1   1
##       0.5  62 520 145  22   2
##       1    12  64  45   7   0
##       2     0   2   7   0   1
##       3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6203
##                  95% CI : (0.5883, 0.6515)
##     No Information Rate : 0.6481
```

```
##     P-Value [Acc > NIR] : 0.9647
##
##                   Kappa : 0.1453
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.16854     0.8581  0.21845  0.00000 0.000000
## Specificity           0.96336     0.2979  0.88615  0.98895 1.000000
## Pos Pred Value        0.32609     0.6924  0.35156  0.00000      NaN
## Neg Pred Value        0.91676     0.5326  0.80050  0.96757 0.995722
## Prevalence            0.09519     0.6481  0.22032  0.03209 0.004278
## Detection Rate        0.01604     0.5561  0.04813  0.00000 0.000000
## Detection Prevalence  0.04920     0.8032  0.13690  0.01070 0.000000
## Balanced Accuracy     0.56595     0.5780  0.55230  0.49448 0.500000

knn_cmatrix7 <- confusionMatrix(knn_res7)
knn_cmatrix7

## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred7   0 0.5   1   2   3
##       0    11  14   5   1   0
##       0.5  66 529 161  25   3
##       1    11  63  40   4   1
##       2     1   0   0   0   0
##       3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6203
##                  95% CI : (0.5883, 0.6515)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.9647
##
##                   Kappa : 0.1073
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.12360     0.8729  0.19417  0.00000 0.000000
## Specificity           0.97636     0.2249  0.89163  0.99890 1.000000
## Pos Pred Value        0.35484     0.6747  0.33613  0.00000      NaN
## Neg Pred Value        0.91372     0.4901  0.79657  0.96788 0.995722
## Prevalence            0.09519     0.6481  0.22032  0.03209 0.004278
## Detection Rate        0.01176     0.5658  0.04278  0.00000 0.000000
## Detection Prevalence  0.03316     0.8385  0.12727  0.00107 0.000000
## Balanced Accuracy     0.54998     0.5489  0.54290  0.49945 0.500000

knn_cmatrix11 <- confusionMatrix(knn_res11)
knn_cmatrix11

## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred11   0 0.5   1   2   3
##        0     7   7   1   1   0
##        0.5  74 552 174  25   3
##        1     8  47  31   4   1
##        2     0   0   0   0   0
##        3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.631
##                  95% CI : (0.5992, 0.662)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.8706
##
##                   Kappa : 0.0843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.078652     0.9109  0.15049  0.00000 0.000000
## Specificity          0.989362     0.1611  0.91770  1.00000 1.000000
## Pos Pred Value       0.437500     0.6667  0.34066      NaN      NaN
## Neg Pred Value       0.910773     0.4953  0.79265  0.96791 0.995722
## Prevalence           0.095187     0.6481  0.22032  0.03209 0.004278
```

```
## Detection Rate       0.007487     0.5904  0.03316  0.00000 0.000000
## Detection Prevalence 0.017112     0.8856  0.09733  0.00000 0.000000
## Balanced Accuracy    0.534007     0.5360  0.53409  0.50000 0.500000

knn_cmatrix21 <- confusionMatrix(knn_res21)
knn_cmatrix21

## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred21   0 0.5   1   2   3
##        0     4   2   1   1   0
##        0.5  80 589 186  27   3
##        1     5  15  19   2   1
##        2     0   0   0   0   0
##        3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6545
##                  95% CI : (0.6231, 0.685)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.3543
##
##                   Kappa : 0.0808
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.044944     0.9719  0.09223  0.00000 0.000000
## Specificity          0.995272     0.1003  0.96845  1.00000 1.000000
## Pos Pred Value       0.500000     0.6655  0.45238      NaN      NaN
## Neg Pred Value       0.908306     0.6600  0.79059  0.96791 0.995722
## Prevalence           0.095187     0.6481  0.22032  0.03209 0.004278
## Detection Rate       0.004278     0.6299  0.02032  0.00000 0.000000
## Detection Prevalence 0.008556     0.9465  0.04492  0.00000 0.000000
## Balanced Accuracy    0.520108     0.5361  0.53034  0.50000 0.500000

# Representación de la precisión:

k<-factor(c("1","3","5","7","11","21"),levels = c("1","3","5","7","11","21"))
```

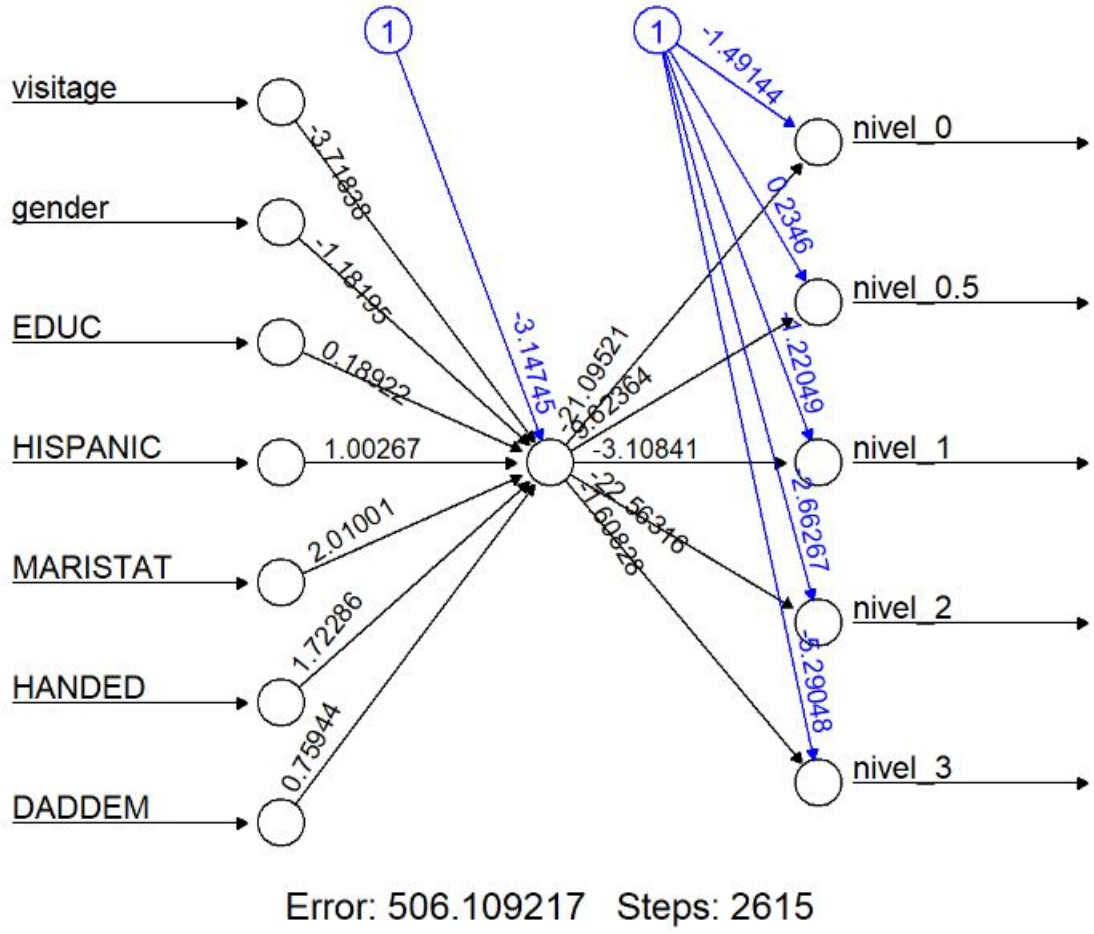
```
precision<-c(0.539,0.5733,0.6203,0.6203,0.631,0.6545)
plot(k,precision,type="p",xlab="Valor de k",ylab="Precisión",
main="Dataset 1",color="red")
```



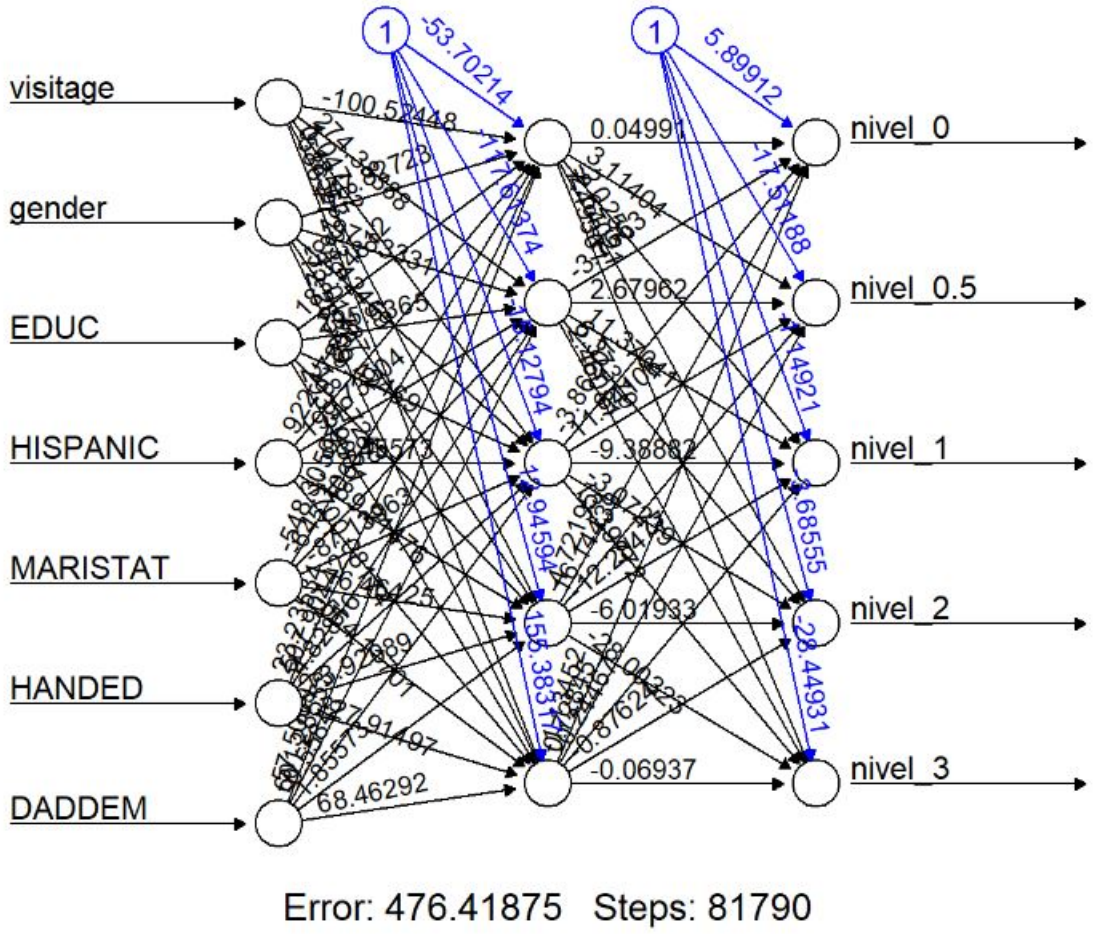

```
#++++++++++++++++++++++++++++++++++#
# Algoritmo Red Neuronal Artificial #
#++++++++++++++++++++++++++++++++++#
```

```
# Preparación de los datos #

# Dividimos la variable "cdrglob" en 5 variables según sus posibles resultados:
adni_dian_norm$nivel_0<-adni_dian$cdrglob==0
adni_dian_norm$nivel_0.5<-adni_dian$cdrglob==0.5
adni_dian_norm$nivel_1<-adni_dian$cdrglob==1
adni_dian_norm$nivel_2<-adni_dian$cdrglob==2
adni_dian_norm$nivel_3<-adni_dian$cdrglob==3
str(adni_dian_norm)

## 'data.frame': 2831 obs. of  12 variables:
##  $ visitage : num  0.54 0.566 0.593 0.618 0.438 ...
##  $ gender   : num  1 1 1 1 0 0 0 0 0 0 ...
##  $ EDUC     : num  0.833 0.833 0.833 0.833 0.75 ...
##  $ HISPANIC : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ MARISTAT : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ HANDED   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ DADDEM   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ nivel_0  : logi  TRUE TRUE FALSE TRUE FALSE FALSE ...
##  $ nivel_0.5: logi  FALSE FALSE TRUE FALSE TRUE TRUE ...
##  $ nivel_1  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ nivel_2  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ nivel_3  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...

# Separación de los datos normalizados:
train_sample_norm<-adni_dian_norm[train_ind, ] # Datos de entrenamiento
test_sample_norm<-adni_dian_norm[-train_ind, ] # Datos de prueba
```

```
# Entrenamiento del modelo #

# Cargamos el paquete "neuralnet":

library(neuralnet)
set.seed(123)
# Creamos el modelo de 1 nodo:
red_pred1<-neuralnet(nivel_0 + nivel_0.5 + nivel_1 + nivel_2 + nivel_3 ~
visitage + gender + EDUC + HISPANIC + MARISTAT + HANDED + DADDEM,
data = train_sample_norm,hidden=1,linear.output=FALSE)
# Representamos el modelo de 1 nodo:
plot(red_pred1,rep="best")
```




```
# Creamos el modelo de 5 nodos:
red_pred2<-neuralnet(nivel_0 + nivel_0.5 + nivel_1 + nivel_2 + nivel_3 ~
visitage + gender + EDUC + HISPANIC + MARISTAT + HANDED + DADDEM,
data = train_sample_norm,hidden=5,linear.output=FALSE)
# Representamos el modelo de 5 nodos:
plot(red_pred2,rep="best")
```




```
# Predicción y evaluación del modelo #

# Realizamos la predicción con el modelo de 1 nodo:
red_pred1_results <- compute(red_pred1, test_sample_norm[,1:7])$net.result
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx_1 <- apply(red_pred1_results, 1, maxidx)
prediction_1<- c("0","0.5","1","2","3")[idx_1]
res_1 <- table(prediction_1, test_sample_labels)
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-res_1[c(2,1,3,4,5),]
rownames(res_1) <-c("0","0.5","1","2","3")
res_1
```

```
# Realizamos la evaluación del rendimiento del modelo:
library(caret)
require(caret)
cmatrix1 <- confusionMatrix(res_1)
cmatrix1
```

```
## Confusion Matrix and Statistics
##
##       0 0.5   1   2   3
## 0     0   0   0   0   0
## 0.5  89 606 206  30   4
## 1     0   0   0   0   0
## 2     0   0   0   0   0
## 3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6481
##                  95% CI : (0.6166, 0.6788)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.515
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.00000     1.0000   0.0000  0.00000 0.000000
## Specificity           1.00000     0.0000   1.0000  1.00000 1.000000
## Pos Pred Value            NaN     0.6481      NaN      NaN      NaN
## Neg Pred Value        0.90481        NaN   0.7797  0.96791 0.995722
## Prevalence            0.09519     0.6481   0.2203  0.03209 0.004278
## Detection Rate        0.00000     0.6481   0.0000  0.00000 0.000000
## Detection Prevalence  0.00000     1.0000   0.0000  0.00000 0.000000
## Balanced Accuracy     0.50000     0.5000   0.5000  0.50000 0.500000
```

```
# Realizamos la predicción con el modelo de 5 nodos:
red_pred2_results <- neuralnet::compute(red_pred2, test_sample_norm[,1:7])$net.result
maxidx2 <- function(arr) {
  return(which(arr == max(arr)))
}
```

```
idx_2 <- apply(red_pred2_results, 1, maxidx2)
prediction_2<- c("0","0.5","1","2","3")[idx_2]
res_2 <- table(prediction_2, test_sample_labels)
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-res_2[c(2,1,3,4,5),]
rownames(res_2) <-c("0","0.5","1","2","3")
res_2

# Realizamos la evaluación del rendimiento del modelo:
library(caret)
require(caret)
cmatrix2 <- confusionMatrix(res_2)
cmatrix2
```

```
## Confusion Matrix and Statistics
##
##       0 0.5   1   2   3
## 0    85 589 194  25   4
## 0.5   2   2   1   0   0
## 1     2  15  11   5   0
## 2     0   0   0   0   0
## 3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.1048
##                  95% CI : (0.0859, 0.1262)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0025
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.95506   0.003300  0.05340  0.00000 0.000000
## Specificity           0.04019   0.990881  0.96982  1.00000 1.000000
## Pos Pred Value        0.09476   0.400000  0.33333      NaN      NaN
## Neg Pred Value        0.89474   0.350538  0.78381  0.96791 0.995722
## Prevalence            0.09519   0.648128  0.22032  0.03209 0.004278
## Detection Rate        0.09091   0.002139  0.01176  0.00000 0.000000
## Detection Prevalence  0.95936   0.005348  0.03529  0.00000 0.000000
## Balanced Accuracy     0.49762   0.497091  0.51161  0.50000 0.500000
```

```r
#++++++++++++++++++++++#
# Algoritmo Naive Bayes #
#++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos el paquete "e1071":
library(e1071)
set.seed(123)
# Creamos el modelo:
naive_bayes<-naiveBayes(train_sample,train_sample_labels,laplace=0)
# Visualizamos tablas formadas en nuestro modelo:
naive_bayes$tables
```

```
## $visitage
##                     visitage
## train_sample_labels     [,1]     [,2]
##                 0   77.44839 5.865848
##                 0.5 75.49990 7.635183
##                 1   76.17893 7.235930
##                 2   79.42785 7.074043
##                 3   72.99798 6.630297
##
## $gender
##                     gender
## train_sample_labels     [,1]      [,2]
##                 0   1.516129 0.5008953
##                 0.5 1.384167 0.4866004
##                 1   1.396985 0.4898886
##                 2   1.444444 0.5003911
##                 3   1.666667 0.5000000
##
## $EDUC
##                     EDUC
## train_sample_labels     [,1]     [,2]
##                 0   15.65899 2.819217
##                 0.5 15.88583 2.787042
##                 1   15.79146 2.675427
##                 2   15.27778 2.748737
##                 3   15.77778 3.231787
##
## $HISPANIC
##                     HISPANIC
```

```
## train_sample_labels        [,1]       [,2]
##                 0   0.004608295 0.06788442
##                 0.5 0.053333333 0.53919492
##                 1   0.070351759 0.65443681
##                 2   0.041666667 0.20122862
##                 3   0.000000000 0.00000000
##
## $MARISTAT
##                     MARISTAT
## train_sample_labels     [,1]      [,2]
##                 0   1.396313 0.8105698
##                 0.5 1.346667 0.9568622
##                 1   1.223618 0.5746491
##                 2   1.319444 0.5771808
##                 3   1.333333 0.7071068
##
## $HANDED
##                     HANDED
## train_sample_labels     [,1]      [,2]
##                 0   1.861751 0.3459590
##                 0.5 1.918333 0.2739704
##                 1   1.917085 0.2761001
##                 2   1.888889 0.3164751
##                 3   2.000000 0.0000000
##
## $DADDEM
##                     DADDEM
## train_sample_labels      [,1]     [,2]
##                 0   0.2626728 1.092965
##                 0.5 0.6116667 1.913264
##                 1   0.5678392 1.824593
##                 2   0.6944444 2.066765
##                 3   1.3333333 2.915476

# Predicción y evaluación del modelo #

# Realizamos la predicción de los datos de prueba utilizando el modelo:
naive_bayes_pred<-predict(naive_bayes,test_sample)
# Evaluamos su eficiencia mediante una matriz de confusión:
naive_bayes_res <- table(naive_bayes_pred, adni_dian$cdrglob[-train_ind])
library(caret)
naive_bayes_cmatrix <- confusionMatrix(naive_bayes_res)
# Mostramos el resultado:
```

```
naive_bayes_cmatrix

## Confusion Matrix and Statistics
##
##
## naive_bayes_pred   0 0.5   1   2   3
##              0     5  48  16   1   1
##              0.5   0  18   8   0   0
##              1     0   1   2   0   0
##              2     0   0   0   0   0
##              3    84 539 180  29   3
##
## Overall Statistics
##
##                Accuracy : 0.0299
##                  95% CI : (0.02, 0.043)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 2e-04
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.056180    0.02970 0.009709  0.00000 0.750000
## Specificity          0.921986    0.97568 0.998628  1.00000 0.106337
## Pos Pred Value       0.070423    0.69231 0.666667      NaN 0.003593
## Neg Pred Value       0.902778    0.35314 0.781116  0.96791 0.990000
## Prevalence           0.095187    0.64813 0.220321  0.03209 0.004278
## Detection Rate       0.005348    0.01925 0.002139  0.00000 0.003209
## Detection Prevalence 0.075936    0.02781 0.003209  0.00000 0.893048
## Balanced Accuracy    0.489083    0.50269 0.504168  0.50000 0.428169

# Probaremos un nuevo valor de la opción "laplace" #

# Creamos el nuevo modelo:
naive_bayes2<-naiveBayes(train_sample,train_sample_labels,laplace=1)
# Mostramos las tablas formadas en el modelo:
naive_bayes2$tables

## $visitage
```

```
##                     visitage
## train_sample_labels     [,1]     [,2]
##                 0   77.44839 5.865848
##                 0.5 75.49990 7.635183
##                 1   76.17893 7.235930
##                 2   79.42785 7.074043
##                 3   72.99798 6.630297
##
## $gender
##                     gender
## train_sample_labels     [,1]      [,2]
##                 0   1.516129 0.5008953
##                 0.5 1.384167 0.4866004
##                 1   1.396985 0.4898886
##                 2   1.444444 0.5003911
##                 3   1.666667 0.5000000
##
## $EDUC
##                     EDUC
## train_sample_labels     [,1]     [,2]
##                 0   15.65899 2.819217
##                 0.5 15.88583 2.787042
##                 1   15.79146 2.675427
##                 2   15.27778 2.748737
##                 3   15.77778 3.231787
##
## $HISPANIC
##                     HISPANIC
## train_sample_labels        [,1]       [,2]
##                 0   0.004608295 0.06788442
##                 0.5 0.053333333 0.53919492
##                 1   0.070351759 0.65443681
##                 2   0.041666667 0.20122862
##                 3   0.000000000 0.00000000
##
## $MARISTAT
##                     MARISTAT
## train_sample_labels     [,1]      [,2]
##                 0   1.396313 0.8105698
##                 0.5 1.346667 0.9568622
##                 1   1.223618 0.5746491
##                 2   1.319444 0.5771808
```

```
##                 3   1.333333 0.7071068
##
## $HANDED
##                    HANDED
## train_sample_labels     [,1]      [,2]
##                 0   1.861751 0.3459590
##                 0.5 1.918333 0.2739704
##                 1   1.917085 0.2761001
##                 2   1.888889 0.3164751
##                 3   2.000000 0.0000000
##
## $DADDEM
##                    DADDEM
## train_sample_labels      [,1]     [,2]
##                 0   0.2626728 1.092965
##                 0.5 0.6116667 1.913264
##                 1   0.5678392 1.824593
##                 2   0.6944444 2.066765
##                 3   1.3333333 2.915476
```

```
# Realizamos la predicción:
naive_bayes_pred2<-predict(naive_bayes2,test_sample)
# Analizamos el rendimiento del mismo mediante una matriz de confusión:
naive_bayes_res2 <- table(naive_bayes_pred2, adni_dian$cdrglob[-train_ind])
library(caret)
naive_bayes_cmatrix2 <- confusionMatrix(naive_bayes_res2)
# Mostramos el resultado:
naive_bayes_cmatrix2
```

```
## Confusion Matrix and Statistics
##
##
## naive_bayes_pred2   0 0.5   1   2   3
##               0     5  48  16   1   1
##               0.5   0  18   8   0   0
##               1     0   1   2   0   0
##               2     0   0   0   0   0
##               3    84 539 180  29   3
##
## Overall Statistics
##
##                Accuracy : 0.0299
##                  95% CI : (0.02, 0.043)
```

```
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 2e-04
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.056180    0.02970 0.009709  0.00000 0.750000
## Specificity          0.921986    0.97568 0.998628  1.00000 0.106337
## Pos Pred Value       0.070423    0.69231 0.666667      NaN 0.003593
## Neg Pred Value       0.902778    0.35314 0.781116  0.96791 0.990000
## Prevalence           0.095187    0.64813 0.220321  0.03209 0.004278
## Detection Rate       0.005348    0.01925 0.002139  0.00000 0.003209
## Detection Prevalence 0.075936    0.02781 0.003209  0.00000 0.893048
## Balanced Accuracy    0.489083    0.50269 0.504168  0.50000 0.428169
```

```
#++++++++++++++++++++++++++++++++++#
# Algoritmo Support Vector Machine #
#++++++++++++++++++++++++++++++++++#

# Preparación de datos #
set.seed(123)
adni_dian$cdrglob<-as.factor(adni_dian$cdrglob)
train_ind2<-sample(seq_len(nrow(adni_dian)),size = smp_size)
train_sample2<-adni_dian[train_ind2, ] # Datos de entrenamiento 2
test_sample2<-adni_dian[-train_ind2, ] # Datos de prueba 2

# Entrenamiento del modelo #

# Cargamos el paquete "kernlab":
library(kernlab)

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

# Fijamos la semilla aleatoria:
set.seed(123)
```

```
# Creamos un modelo con la función lineal:
svm <- ksvm(cdrglob~.,data=train_sample2, kernel='vanilladot')

##  Setting default kernel parameters

# Mostramos la información básica del modelo:
svm

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 1330
##
## Objective Function Value : -434 -433.3045 -140.5521 -18 -796 -144 -18 -144 -18 -18
## Training error : 0.367089

# Predicción y evaluación del modelo #

# LLevamos a cabo la predicción de nuestros datos de prueba utilizando el
# modelo SVM
# de la función lineal:
svm_pred<-predict(svm,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
library(caret)
svm_res<-table(svm_pred,test_sample2$cdrglob)
svm_cmatrix<-confusionMatrix(svm_res)
# Mostramos el resultado:
svm_cmatrix

## Confusion Matrix and Statistics
##
##
## svm_pred   0 0.5   1   2   3
##      0     0   0   0   0   0
##      0.5  89 606 206  30   4
##      1     0   0   0   0   0
##      2     0   0   0   0   0
##      3     0   0   0   0   0
##
```

```
## Overall Statistics
##
##                Accuracy : 0.6481
##                  95% CI : (0.6166, 0.6788)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.515
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.00000     1.0000   0.0000  0.00000 0.000000
## Specificity           1.00000     0.0000   1.0000  1.00000 1.000000
## Pos Pred Value            NaN     0.6481      NaN      NaN      NaN
## Neg Pred Value        0.90481        NaN   0.7797  0.96791 0.995722
## Prevalence            0.09519     0.6481   0.2203  0.03209 0.004278
## Detection Rate        0.00000     0.6481   0.0000  0.00000 0.000000
## Detection Prevalence  0.00000     1.0000   0.0000  0.00000 0.000000
## Balanced Accuracy     0.50000     0.5000   0.5000  0.50000 0.500000
```

```
# Mejora del modelo #

# Cargamos el paquete "kernlab":
library(kernlab)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos un modelo con la función lineal:
svm_rbf <- ksvm(cdrglob~.,data=train_sample2, kernel='rbfdot')
# Mostramos la información básica del modelo:
svm_rbf
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.320017903205958
##
## Number of Support Vectors : 1513
```

```
##
## Objective Function Value : -418.9446 -350.5727 -125.0681 -17.0821 -776.1257 -141.1386
## Training error : 0.353903

# LLevamos a cabo la predicción de nuestros datos de prueba utilizando el
# modelo SVM
# de la función lineal:
svm_rbf_pred<-predict(svm_rbf,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
library(caret)
svm_rbf_res<-table(svm_rbf_pred,test_sample2$cdrglob)
svm_rbf_cmatrix<-confusionMatrix(svm_rbf_res)
svm_rbf_cmatrix

## Confusion Matrix and Statistics
##
##
## svm_rbf_pred   0 0.5   1   2   3
##          0     1   1   0   0   0
##          0.5  87 605 204  28   3
##          1     1   0   2   2   1
##          2     0   0   0   0   0
##          3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6503
##                  95% CI : (0.6187, 0.6809)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.4604
##
##                   Kappa : 0.0171
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.011236    0.99835 0.009709  0.00000 0.000000
## Specificity          0.998818    0.02128 0.994513  1.00000 1.000000
## Pos Pred Value       0.500000    0.65264 0.333333      NaN      NaN
## Neg Pred Value       0.905681    0.87500 0.780409  0.96791 0.995722
## Prevalence           0.095187    0.64813 0.220321  0.03209 0.004278
```

```
## Detection Rate       0.001070    0.64706 0.002139  0.00000 0.000000
## Detection Prevalence 0.002139    0.99144 0.006417  0.00000 0.000000
## Balanced Accuracy    0.505027    0.50981 0.502111  0.50000 0.500000
```

```
#+++++++++++++++++++++++++++++++#
# Algoritmo Árbol de Decisión  #
#+++++++++++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos las librerías necesarias para llevar a cabo este algoritmo:
library(C50)
library(caret)
library(tidyverse)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos el modelo:
arbol<-C5.0(train_sample2[-1],train_sample2$cdrglob)
# Mostramos la información del modelo:
arbol
```

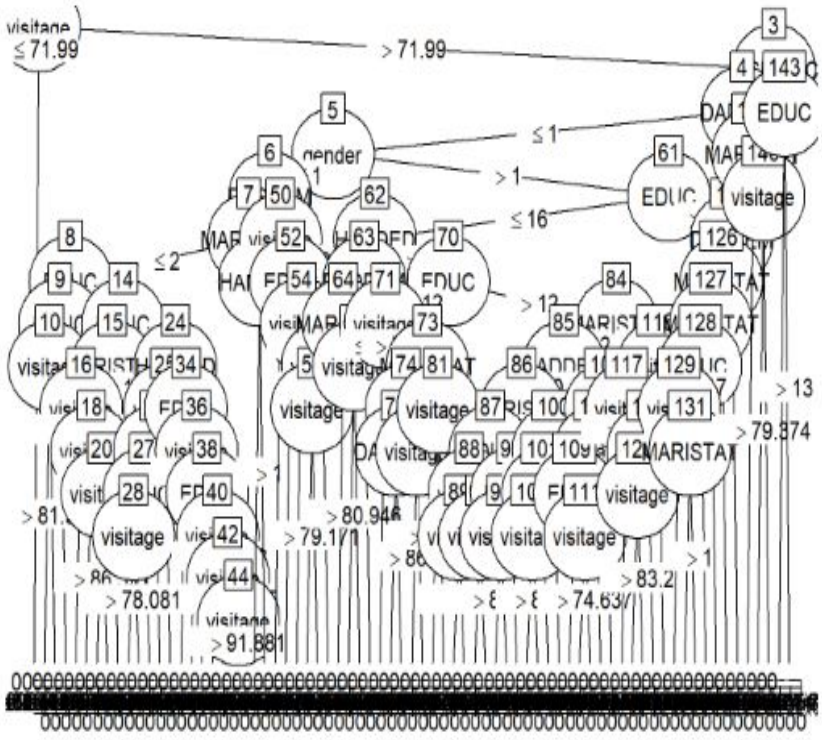
```
##
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$cdrglob)
##
## Classification Tree
## Number of samples: 1896
## Number of predictors: 7
##
## Tree size: 73
##
## Non-standard options: attempt to group attributes
```

```
# Hacemos un resumen estadístico del modelo:
summary(arbol)
```

```
##
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$cdrglob)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Dec 23 21:09:23 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 1896 cases (8 attributes) from undefined.data
##
## Decision tree:
##
## visitage <= 71.98973: 0.5 (518/146)
## visitage > 71.98973:
## :...HISPANIC > 0:
##     :...EDUC <= 18: 1 (3)
##     :   EDUC > 18: 0.5 (28/8)
##     HISPANIC <= 0:
##     :...DADDEM > 1:
##         :...MARISTAT <= 1: 0.5 (24/9)
##         :   MARISTAT > 1:
##         :   :...visitage <= 79.37399: 0.5 (9/3)
##         :       visitage > 79.37399: 1 (7/1)
##         DADDEM <= 1:
##         :...gender > 1:
##             :...EDUC > 16:
##             :   :...DADDEM > 0: 0.5 (4/2)
##             :   :   DADDEM <= 0:
##             :   :   :...MARISTAT <= 2: 0.5 (7)
##             :   :       MARISTAT <= 2:
##             :   :       :...MARISTAT > 2: 0 (6/1)
##             :   :           MARISTAT <= 2:
##             :   :           :...EDUC > 17: 0.5 (67/23)
##             :   :               EDUC <= 17:
##             :   :               :...visitage <= 77.1: 0.5 (5/1)
##             :   :                   visitage > 77.1:
##             :   :                   :...MARISTAT <= 1: 0 (6)
##             :   :                       MARISTAT > 1: 0.5 (5/1)
##             :   EDUC <= 16:
##             :   :...HANDED <= 1:
##             :       :...DADDEM > 0: 0.5 (10/2)
##             :       :   DADDEM <= 0:
##             :       :   :...MARISTAT <= 1: 0.5 (19/10)
##             :       :       MARISTAT > 1:
##             :       :       :...visitage <= 80.94648: 0 (14/2)
##             :       :           visitage > 80.94648: 0.5 (9/1)
##             :       HANDED > 1:
##             :       :...EDUC <= 12:
##             :           :...visitage <= 72.59125: 1 (5/3)
##             :           :   visitage > 72.59125:
##             :           :   :...MARISTAT > 1:
##             :           :       :...visitage <= 82.06427: 1 (15/6)
##             :           :       :   visitage > 82.06427: 0.5 (16/2)
##             :           :       MARISTAT <= 1:
##             :           :       :...EDUC <= 11:
##             :           :           :...DADDEM <= 0: 1 (6)
##             :           :           :   DADDEM > 0: 0.5 (2)
##             :           :           EDUC > 11:
##             :           :           :...visitage <= 86.31643: 0.5 (52/13)
##             :           :               visitage > 86.31643: 1 (4/1)
##             :           EDUC > 12:
##             :           :...MARISTAT > 2:
##             :               :...visitage > 86.02259: 2 (2)
##             :               :   visitage <= 86.02259:
##             :               :   :...visitage <= 77.22259: 0.5 (5)
##             :               :       visitage > 77.22259:
##             :               :       :...EDUC > 14: 1 (8/2)
##             :               :           EDUC <= 14:
##             :               :           :...visitage <= 89.2: 0.5 (8)
##             :               :               visitage > 89.2: 1 (3)
##             :               MARISTAT <= 2:
##             :               :...DADDEM <= 0:
##             :                   :...MARISTAT <= 1:
##             :                   :   :...EDUC <= 13:
##             :                   :   :   :...visitage <= 81.71232:
##             :                   :   :   :   :...visitage <= 79.2297: 0.5 (4)
##             :                   :   :   :   :   visitage > 79.2297: 1 (7/3)
##             :                   :   :   :   visitage > 81.71232:
##             :                   :   :   :   :...visitage <= 90.19795: 0 (11/3)
##             :                   :   :   :       visitage > 90.19795: 0.5 (2)
##             :                   :   :   EDUC > 13:
##             :                   :   :   :...EDUC > 14: 0.5 (45/20)
##             :                   :   :       EDUC <= 14:
##             :                   :   :       :...visitage <= 86.62943: 1 (19/6)
##             :                   :   :           visitage > 86.62943: 0.5 (4)
##             :                   :   MARISTAT > 1:
##             :                   :   :...visitage > 89.07536: 2 (3/1)
##             :                   :       visitage <= 89.07536:
##             :                   :       :...EDUC > 15: 0.5 (12/2)
##             :                   :           EDUC <= 15:
##             :                   :           :...visitage <= 83.2: 0 (23/9)
##             :                   :               visitage > 83.2: 0.5 (5)
##             :                   DADDEM > 0:
##             :                   :...EDUC > 15: 0.5 (2/1)
##             :                       EDUC <= 15:
##             :                       :...visitage > 84.14579: 0.5 (5)
##             :                           visitage <= 84.14579:
##             :                           :...EDUC <= 14: 0 (28/11)
##             :                               EDUC > 14:
##             :                               :...visitage <= 74.63662: 0 (2)
##             :                                   visitage > 74.63662: 0.5 (14/2)
##             gender <= 1:
##             :...DADDEM > 0:
##                 :...visitage <= 72.31848: 1 (7/2)
##                 :   visitage > 72.31848:
##                 :...EDUC <= 19: 0.5 (92/22)
##                     EDUC > 19:
##                     :...visitage <= 73.6: 1 (3)
##                         visitage > 73.6:
##                         :...visitage > 86.92296: 0.5 (2)
##                             visitage <= 86.92296:
##                             :...visitage <= 79.17125: 0.5 (13/4)
##                                 visitage > 79.17125: 1 (13/5)
##                 DADDEM <= 0:
##                 :...MARISTAT > 2:
##                     :...HANDED <= 1: 0 (4)
##                     :   HANDED > 1: 0.5 (17/3)
##                     MARISTAT <= 2:
##                     :...EDUC <= 12:
##                         :...EDUC > 8: 0.5 (103/24)
##                         :   EDUC <= 8:
##                         :   :...visitage <= 81.21269: 0 (4)
##                         :       visitage > 81.21269: 0.5 (5/1)
```

```
##                           EDUC > 12:
##                           :...EDUC <= 14:
##                               :...MARISTAT > 1: 0 (4/1)
##                               :   MARISTAT <= 1:
##                               :       :...visitage <= 79.89227: 0.5 (46/20)
##                               :       :   visitage > 79.89227:
##                               :       :...visitage <= 82.2: 1 (17/5)
##                               :       :   visitage > 82.2:
##                               :       :...visitage <= 86.33149: 2 (5/4)
##                               :       :   visitage > 86.33149: 0 (2/1)
##                               :   EDUC > 14:
##                               :...HANDED <= 1:
##                                   :...visitage > 88: 2 (2)
##                                   :   visitage <= 88:
##                                   :...EDUC > 17: 0.5 (24/11)
##                                   :   EDUC <= 17:
##                                   :       :...EDUC > 16: 1 (3)
##                                   :       :   EDUC <= 16:
##                                   :       :       :...visitage <= 78.05053: 0.5 (7/2)
##                                   :       :           visitage > 78.05053: 1 (11/3)
##                                   :   HANDED > 1:
##                                   :...EDUC <= 18: 0.5 (299/111)
##                                       EDUC > 18:
##                                       :...visitage <= 85.00103: 0.5 (87/21)
##                                           visitage > 85.00103:
##                                           :...EDUC <= 19: 0.5 (5/1)
##                                               EDUC > 19:
##                                               :...visitage <= 87.6027: 1 (14/6)
##                                                   visitage > 87.6027:
##                                                   :...visitage <= 89.39521: 2 (2/1)
##                                                       visitage > 89.39521: [S1]
##
## SubTree [S1]
##
## visitage <= 91.8808: 0 (4/1)
## visitage > 91.8808: 0.5 (2)
##
##
## Evaluation on training data (1596 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      72  546(28.8%)   <<
##
##
##     (a)   (b)   (c)   (d)   (e)    <-classified as
##    ----  ----  ----  ----  ----
##      50   129     6     2          (a): class 0
##      23  1152    22     2          (b): class 0.5
##       5   256   105     2          (c): class 1
##       1    46    12    12          (d): class 2
##             7     2                (e): class 3
##
##
##  Attribute usage:
##
##  100.00% visitage
##   72.68% HISPANIC
##   71.04% DADDEM
##   68.67% EDUC
##   68.41% gender
##   60.12% MARISTAT
##   44.99% HANDED
##
##
## Time: 0.0 secs
```

```
# Realizamos una representación gráfica del modelo:
plot(arbol)
```



```
# Predicción y evaluación del modelo #

# Realizamos la prediccion de los datos de prueba utilizando el modelo creado:
arbol_pred<-predict(arbol,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
arbol_res<-table(arbol_pred,test_sample_labels)
arbol_cmatrix<-confusionMatrix(arbol_res)
# Mostramos la matriz de confusión:
arbol_cmatrix
```

```
## Confusion Matrix and Statistics
##
##           test_sample_labels
## arbol_pred   0 0.5   1   2   3
##        0    22  24   3   1   0
##        0.5  66 555 176  24   3
##        1     0  25  21   4   1
##        2     1   2   6   1   0
##        3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6406
##                  95% CI : (0.6089, 0.6714)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.6971
##
##                   Kappa : 0.1264
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.24719     0.9158  0.10194  0.03333 0.000000
## Specificity           0.96690     0.1824  0.95885  0.99006 1.000000
## Pos Pred Value        0.44000     0.6735  0.41176  0.10000      NaN
## Neg Pred Value        0.92429     0.5405  0.79072  0.96865 0.995722
## Prevalence            0.09519     0.6481  0.22032  0.03209 0.004278
## Detection Rate        0.02353     0.5936  0.02246  0.00107 0.000000
## Detection Prevalence  0.05348     0.8813  0.05455  0.01070 0.000000
## Balanced Accuracy     0.60705     0.5491  0.53039  0.51169 0.500000
```

```
# Mejora del modelo #

# Creamos el nuevo modelo:
boost10<-C5.0(train_sample2[-1],train_sample2$cdrglob,trials=10)
# Mostramos la información básica del modelo:
boost10
```

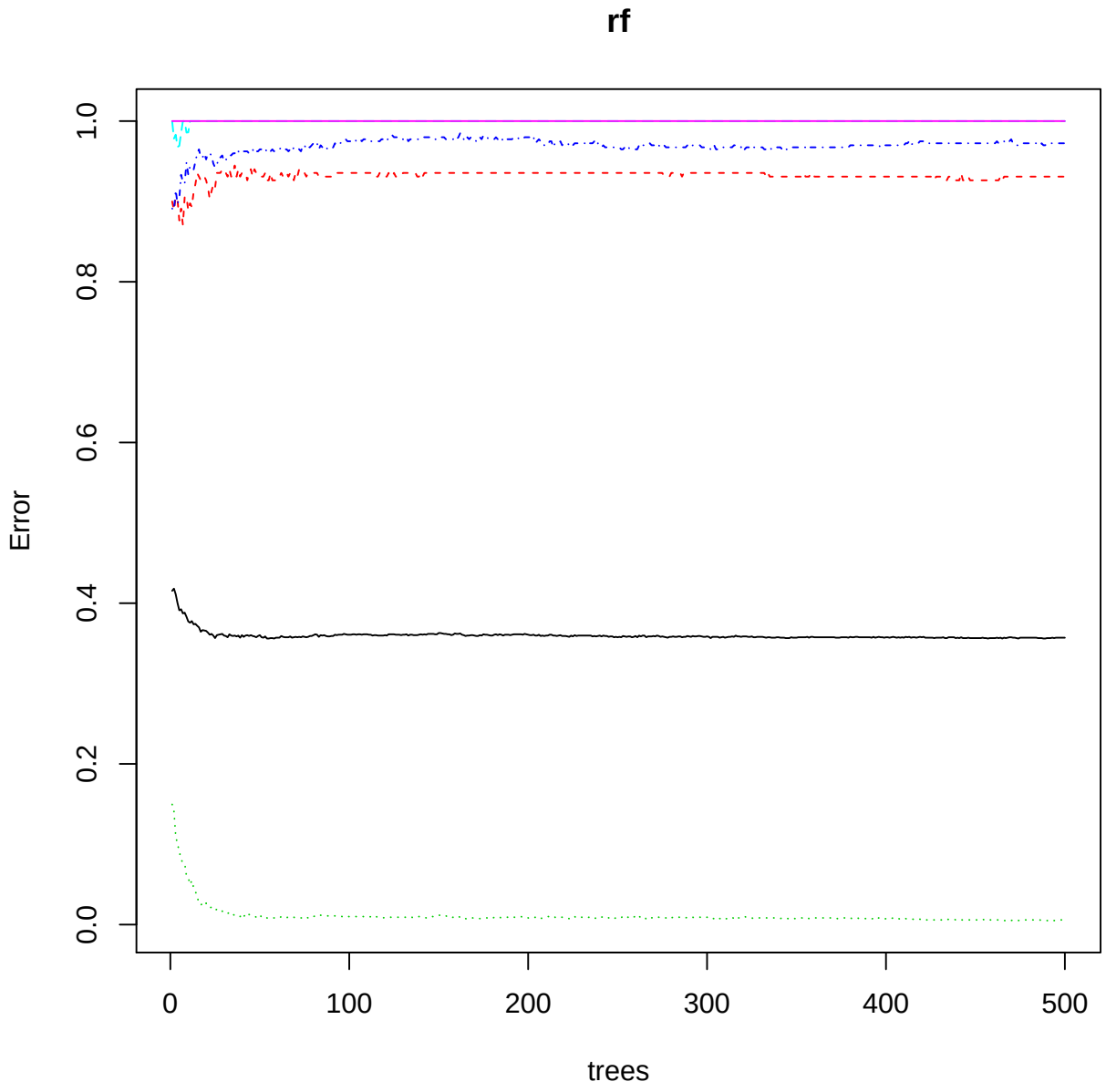
```
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$cdrglob, trials = 10)
##
## Classification Tree
## Number of samples: 1896
## Number of predictors: 7
##
## Number of boosting iterations: 10 requested;  1 used due to early stopping
##
## Non-standard options: attempt to group attributes
```

```
# Realizamos la predicción utilizando el nuevo modelo:
boost10_pred<-predict(boost10,test_sample2)
# Realizamos la comparativa de los datos predichos con los reales utilizando una matriz
# de confusión:
boost10_res<-table(boost10_pred,test_sample_labels)
boost10_cmatrix<-confusionMatrix(boost10_res)
# Mostramos el resultado:
boost10_cmatrix
```

```
## Confusion Matrix and Statistics
##
##             test_sample_labels
## boost10_pred   0 0.5   1   2   3
##          0    22  24   3   1   0
##          0.5  66 555 176  24   3
##          1     0  25  21   4   1
##          2     1   2   6   1   0
##          3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6406
##                  95% CI : (0.6089, 0.6714)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.6971
##
##                   Kappa : 0.1264
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity           0.24719     0.9158  0.10194  0.03333 0.000000
## Specificity           0.96690     0.1824  0.95885  0.99006 1.000000
## Pos Pred Value        0.44000     0.6735  0.41176  0.10000      NaN
## Neg Pred Value        0.92429     0.5405  0.79072  0.96865 0.995722
## Prevalence            0.09519     0.6481  0.22032  0.03209 0.004278
## Detection Rate        0.02353     0.5936  0.02246  0.00107 0.000000
## Detection Prevalence  0.05348     0.8813  0.05455  0.01070 0.000000
## Balanced Accuracy     0.60705     0.5491  0.53039  0.51169 0.500000
```

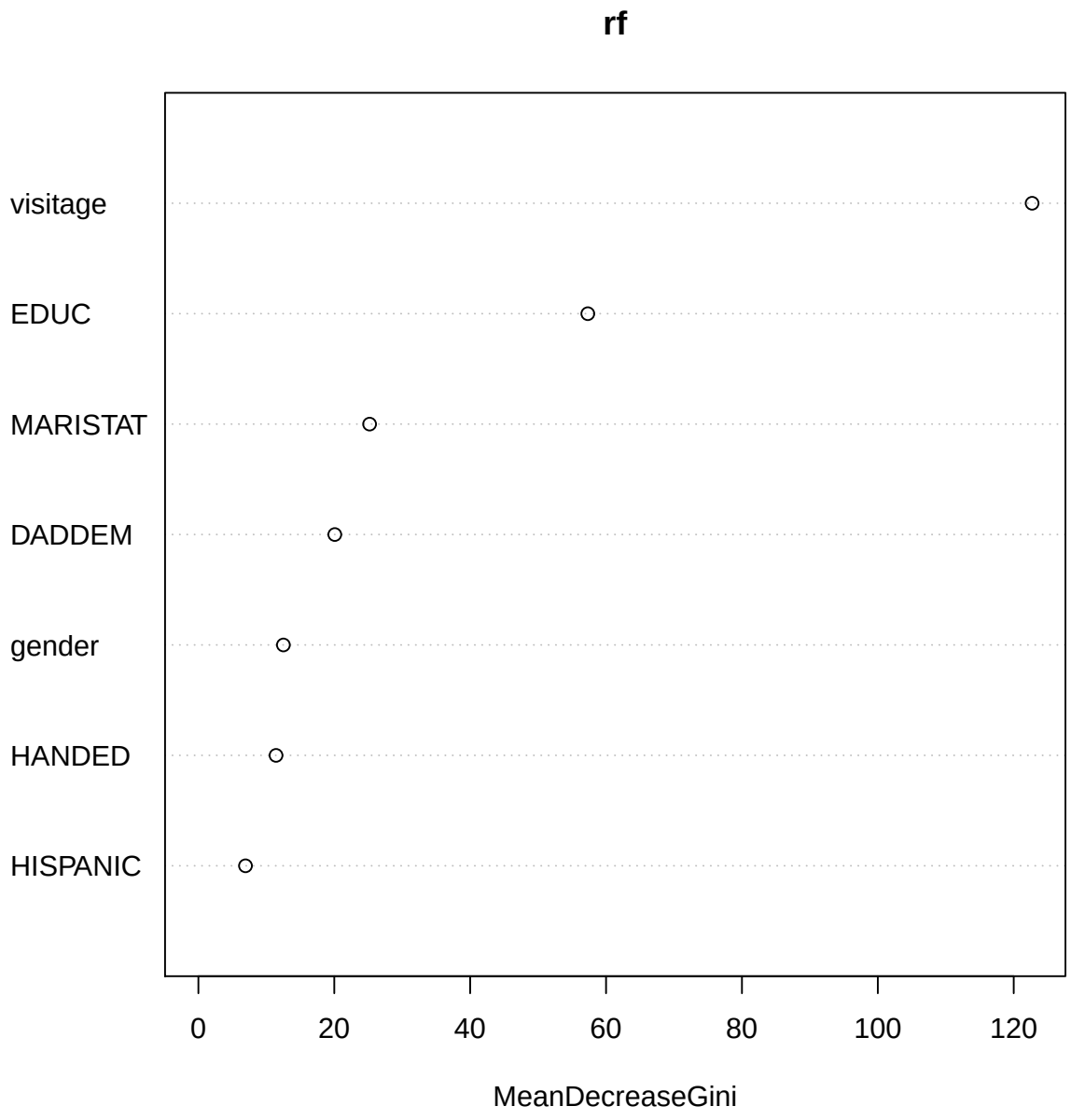
```
#++++++++++++++++++++++++++#
# Algoritmo Random Forest #
#++++++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos el paquete "randomForest":
library(randomForest)

## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin

# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos el modelo:
rf <- randomForest(cdrglob ~ ., data = train_sample2)
# Mostramos el modelo:
print(rf)

##
## Call:
##  randomForest(formula = cdrglob ~ ., data = train_sample2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 35.71%
## Confusion matrix:
##      0  0.5  1 2 3 class.error
## 0   15  202  0 0 0 0.930875576
## 0.5  4 1193  2 1 0 0.005833333
## 1    0  387 11 0 0 0.972361809
## 2    0   71  1 0 0 1.000000000
## 3    0    9  0 0 0 1.000000000

# Representamos gráficamente el modelo:
plot(rf)
```




```
# Mostramos la importancia de las variables:
importance(rf)

##           MeanDecreaseGini
## visitage        122.707148
## gender           12.510610
## EDUC             57.314772
## HISPANIC          6.933743
## MARISTAT         25.196883
```

```
## HANDED          11.428900
## DADDEM          20.075519

# Representamos gráficamente la importancia de las variables:
varImpPlot(rf)
```




```
# Predicción y evaluación del modelo #
```

```
# Realizamos la predicción:
rf_pred<-predict(rf,test_sample2[-9])
# Calculamos la matriz de confusión:
rf_res<-table(rf_pred,test_sample_labels)
rf_cmatrix<-confusionMatrix(rf_res)
# Mostramos el resultado:
rf_cmatrix

## Confusion Matrix and Statistics
##
##        test_sample_labels
## rf_pred   0 0.5   1   2   3
##     0     1   1   0   0   0
##     0.5  88 605 205  29   3
##     1     0   0   1   1   1
##     2     0   0   0   0   0
##     3     0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6492
##                  95% CI : (0.6176, 0.6798)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.4877
##
##                   Kappa : 0.0102
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.011236    0.99835 0.004854  0.00000 0.000000
## Specificity          0.998818    0.01216 0.997257  1.00000 1.000000
## Pos Pred Value       0.500000    0.65054 0.333333      NaN      NaN
## Neg Pred Value       0.905681    0.80000 0.780043  0.96791 0.995722
## Prevalence           0.095187    0.64813 0.220321  0.03209 0.004278
## Detection Rate       0.001070    0.64706 0.001070  0.00000 0.000000
## Detection Prevalence 0.002139    0.99465 0.003209  0.00000 0.000000
## Balanced Accuracy    0.505027    0.50525 0.501055  0.50000 0.500000

# Mejora del modelo #
```

```
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos el modelo:
rf.1000<-randomForest(cdrglob ~ ., data = train_sample2,ntree=1000)
# Mostramos la información básica del modelo:
rf.1000

##
## Call:
##  randomForest(formula = cdrglob ~ ., data = train_sample2, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 35.76%
## Confusion matrix:
##      0  0.5   1 2 3 class.error
## 0   14  203   0 0 0   0.9354839
## 0.5  3 1194   3 0 0   0.0050000
## 1    0  388 10 0 0   0.9748744
## 2    0   71   1 0 0   1.0000000
## 3    0    9   0 0 0   1.0000000

# Realizamos la predicción:
rf.1000_pred<-predict(rf.1000,test_sample2[-9])
# Contrastamos el rendimiento del mismo con resultados verdaderos de los
# datos de prueba utilizando una matriz de confusión:
rf.1000_res<-table(rf.1000_pred,test_sample_labels)
rf.1000_cmatrix<-confusionMatrix(rf.1000_res)
# Mostramos los resultados:
rf.1000_cmatrix

## Confusion Matrix and Statistics
##
##             test_sample_labels
## rf.1000_pred   0 0.5   1   2   3
##          0     1   1   0   0   0
##          0.5  88 605 205  29   3
##          1     0   0   1   1   1
##          2     0   0   0   0   0
##          3     0   0   0   0   0
##
## Overall Statistics
```

```
##
##                Accuracy : 0.6492
##                  95% CI : (0.6176, 0.6798)
##     No Information Rate : 0.6481
##     P-Value [Acc > NIR] : 0.4877
##
##                   Kappa : 0.0102
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2 Class: 3
## Sensitivity          0.011236    0.99835 0.004854  0.00000 0.000000
## Specificity          0.998818    0.01216 0.997257  1.00000 1.000000
## Pos Pred Value       0.500000    0.65054 0.333333      NaN      NaN
## Neg Pred Value       0.905681    0.80000 0.780043  0.96791 0.995722
## Prevalence           0.095187    0.64813 0.220321  0.03209 0.004278
## Detection Rate       0.001070    0.64706 0.001070  0.00000 0.000000
## Detection Prevalence 0.002139    0.99465 0.003209  0.00000 0.000000
## Balanced Accuracy    0.505027    0.50525 0.501055  0.50000 0.500000
```

```
#~~~~~~~~~~~~~~~#
# Predicción 2 #
#~~~~~~~~~~~~~~~#

#==========================#
# Preparación de los datos #
#==========================#

# Extraemos los factores del riesgo que hemos obtenido en el análisis del
# dataset 1 para crear un nuevo dataset:

cdr_vitals_medhist<-cdr_vitals_medhist[,c(1,3,5,13,15,23,24)]
cdr_vitals_medhist_pred<-cdr_vitals_medhist[,c(2,3,4,5,6,7)]
str(cdr_vitals_medhist_pred)

## 'data.frame': 1138 obs. of  6 variables:
##  $ VSHEIGHT: num  71 69 69 71 64 54 65 62 66 -4 ...
##  $ VSBPDIA : int  80 70 70 80 75 80 74 90 80 70 ...
##  $ MH3HEAD : int  0 1 1 0 1 1 0 0 1 1 ...
##  $ MH5RESP : int  0 0 0 0 0 1 0 0 1 0 ...
```

```
##  $ MH13ALLE: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH14ALCH: int  0 0 0 0 0 0 0 0 0 0 ...

# Separamos los datos en dos grupos:
smp_size<-floor(0.67 * nrow(cdr_vitals_medhist_pred))
set.seed(111)
train_ind<-sample(seq_len(nrow(cdr_vitals_medhist_pred)),size = smp_size)
train_sample<-cdr_vitals_medhist_pred[train_ind, ] # Datos de entrenamiento 1
test_sample<-cdr_vitals_medhist_pred[-train_ind, ] # Datos de prueba 1


# Etiquetas de clase de los datos de entrenamiento:
train_sample_labels<-cdr_vitals_medhist[train_ind,1]
table(train_sample_labels)

## train_sample_labels
##   0 0.5   1   2
## 233 433  92   4

# Etiquetas de clase de los datos de prueba:
test_sample_labels<-cdr_vitals_medhist[-train_ind,1]
table(test_sample_labels)

## test_sample_labels
##   0 0.5   1   2
## 128 209  37   2

# Comprobamos que los datos se hayan dividido correctamente observando el
# número de registros:
dim(train_sample)

## [1] 762   6

dim(test_sample)

## [1] 376   6

# Normalización de los datos:
normalize <- function(x) {
return((x - min(x)) / (max(x) - min(x)))
}
cdr_vitals_medhist_norm<-as.data.frame(lapply(cdr_vitals_medhist_pred,normalize))
str(cdr_vitals_medhist_norm)
```

```
## 'data.frame': 1138 obs. of  6 variables:
##  $ VSHEIGHT: num  0.385 0.374 0.374 0.385 0.349 ...
##  $ VSBPDIA : num  0.73 0.64 0.64 0.73 0.685 ...
##  $ MH3HEAD : num  0 1 1 0 1 1 0 0 1 1 ...
##  $ MH5RESP : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ MH13ALLE: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH14ALCH: num  0 0 0 0 0 0 0 0 0 0 ...
```

```
#=======================#
# Modelos de predicción #
#=======================#

#+++++++++++++++++#
# Algoritmo k-NN #
#+++++++++++++++++#

# Entrenamiento del modelo #

library(class) # Cargamos el paquete "class"
set.seed(123)
# Utilizamos la función "knn" con diferentes valores de k:
# k = 1
knn_pred1<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=1)
# k = 3
knn_pred3<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=3)
# k = 5
knn_pred5<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=5)
# k = 7
knn_pred7<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=7)
# k = 10
knn_pred10<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=10)
# k = 18
knn_pred18<-knn(train=train_sample,test=test_sample,cl=train_sample_labels,k=18)

# Evaluación del rendimiento del modelo #

library(gmodels) # Cargamos el paquete "gmodels"

# Utilizamos la función "CrossTable" para comprobar el funcionamiento de
# cada modelo:
# k = 1
CrossTable(x=test_sample_labels,y=knn_pred1,prop.chisq = FALSE)
```

```
##
```

```
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                     | knn_pred1
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|
##                  0 |        40 |        79 |         9 |       128 |
##                    |     0.312 |     0.617 |     0.070 |     0.340 |
##                    |     0.351 |     0.343 |     0.281 |           |
##                    |     0.106 |     0.210 |     0.024 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                0.5 |        62 |       130 |        17 |       209 |
##                    |     0.297 |     0.622 |     0.081 |     0.556 |
##                    |     0.544 |     0.565 |     0.531 |           |
##                    |     0.165 |     0.346 |     0.045 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |        11 |        21 |         5 |        37 |
##                    |     0.297 |     0.568 |     0.135 |     0.098 |
##                    |     0.096 |     0.091 |     0.156 |           |
##                    |     0.029 |     0.056 |     0.013 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  2 |         1 |         0 |         1 |         2 |
##                    |     0.500 |     0.000 |     0.500 |     0.005 |
##                    |     0.009 |     0.000 |     0.031 |           |
##                    |     0.003 |     0.000 |     0.003 |           |
## -------------------|-----------|-----------|-----------|-----------|
##       Column Total |       114 |       230 |        32 |       376 |
##                    |     0.303 |     0.612 |     0.085 |           |
## -------------------|-----------|-----------|-----------|-----------|
##
##

# k = 3
```

```
CrossTable(x=test_sample_labels,y=knn_pred3,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                     | knn_pred3
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|
##                  0 |        38 |        84 |         6 |       128 |
##                    |     0.297 |     0.656 |     0.047 |     0.340 |
##                    |     0.437 |     0.317 |     0.250 |           |
##                    |     0.101 |     0.223 |     0.016 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                0.5 |        43 |       150 |        16 |       209 |
##                    |     0.206 |     0.718 |     0.077 |     0.556 |
##                    |     0.494 |     0.566 |     0.667 |           |
##                    |     0.114 |     0.399 |     0.043 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |         5 |        30 |         2 |        37 |
##                    |     0.135 |     0.811 |     0.054 |     0.098 |
##                    |     0.057 |     0.113 |     0.083 |           |
##                    |     0.013 |     0.080 |     0.005 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  2 |         1 |         1 |         0 |         2 |
##                    |     0.500 |     0.500 |     0.000 |     0.005 |
##                    |     0.011 |     0.004 |     0.000 |           |
##                    |     0.003 |     0.003 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|
##       Column Total |        87 |       265 |        24 |       376 |
##                    |     0.231 |     0.705 |     0.064 |           |
## -------------------|-----------|-----------|-----------|-----------|
##
```

```
##

# k = 5
CrossTable(x=test_sample_labels,y=knn_pred5,prop.chisq = FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                     | knn_pred5
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|
##                  0 |        39 |        86 |         3 |       128 |
##                    |     0.305 |     0.672 |     0.023 |     0.340 |
##                    |     0.481 |     0.301 |     0.333 |           |
##                    |     0.104 |     0.229 |     0.008 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                0.5 |        37 |       167 |         5 |       209 |
##                    |     0.177 |     0.799 |     0.024 |     0.556 |
##                    |     0.457 |     0.584 |     0.556 |           |
##                    |     0.098 |     0.444 |     0.013 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |         4 |        32 |         1 |        37 |
##                    |     0.108 |     0.865 |     0.027 |     0.098 |
##                    |     0.049 |     0.112 |     0.111 |           |
##                    |     0.011 |     0.085 |     0.003 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  2 |         1 |         1 |         0 |         2 |
##                    |     0.500 |     0.500 |     0.000 |     0.005 |
##                    |     0.012 |     0.003 |     0.000 |           |
##                    |     0.003 |     0.003 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|
##       Column Total |        81 |       286 |         9 |       376 |
```

```
##                    |     0.215 |     0.761 |     0.024 |           |
## -------------------|-----------|-----------|-----------|-----------|
##
##

# k = 7
CrossTable(x=test_sample_labels,y=knn_pred7,prop.chisq = FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                    | knn_pred7
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## -------------------|-----------|-----------|-----------|-----------|
##                  0 |        33 |        95 |         0 |       128 |
##                    |     0.258 |     0.742 |     0.000 |     0.340 |
##                    |     0.440 |     0.320 |     0.000 |           |
##                    |     0.088 |     0.253 |     0.000 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                0.5 |        39 |       167 |         3 |       209 |
##                    |     0.187 |     0.799 |     0.014 |     0.556 |
##                    |     0.520 |     0.562 |     0.750 |           |
##                    |     0.104 |     0.444 |     0.008 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |         3 |        33 |         1 |        37 |
##                    |     0.081 |     0.892 |     0.027 |     0.098 |
##                    |     0.040 |     0.111 |     0.250 |           |
##                    |     0.008 |     0.088 |     0.003 |           |
## -------------------|-----------|-----------|-----------|-----------|
##                  2 |         0 |         2 |         0 |         2 |
##                    |     0.000 |     1.000 |     0.000 |     0.005 |
##                    |     0.000 |     0.007 |     0.000 |           |
```

```
##                      |     0.000 |     0.005 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|
##       Column Total |        75 |       297 |         4 |       376 |
##                      |     0.199 |     0.790 |     0.011 |           |
## --------------------|-----------|-----------|-----------|-----------|
##
##
```

```
# k = 11
CrossTable(x=test_sample_labels,y=knn_pred10,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                      | knn_pred10
## test_sample_labels |         0 |       0.5 |         1 | Row Total |
## --------------------|-----------|-----------|-----------|-----------|
##                   0 |        30 |        98 |         0 |       128 |
##                     |     0.234 |     0.766 |     0.000 |     0.340 |
##                     |     0.500 |     0.311 |     0.000 |           |
##                     |     0.080 |     0.261 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                 0.5 |        26 |       182 |         1 |       209 |
##                     |     0.124 |     0.871 |     0.005 |     0.556 |
##                     |     0.433 |     0.578 |     1.000 |           |
##                     |     0.069 |     0.484 |     0.003 |           |
## --------------------|-----------|-----------|-----------|-----------|
##                   1 |         4 |        33 |         0 |        37 |
##                     |     0.108 |     0.892 |     0.000 |     0.098 |
##                     |     0.067 |     0.105 |     0.000 |           |
##                     |     0.011 |     0.088 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|
```

```
##                   2 |         0 |         2 |         0 |         2 |
##                     |     0.000 |     1.000 |     0.000 |     0.005 |
##                     |     0.000 |     0.006 |     0.000 |           |
##                     |     0.000 |     0.005 |     0.000 |           |
## --------------------|-----------|-----------|-----------|-----------|
##       Column Total |        60 |       315 |         1 |       376 |
##                     |     0.160 |     0.838 |     0.003 |           |
## --------------------|-----------|-----------|-----------|-----------|
##
##
```

```
# k = 17
CrossTable(x=test_sample_labels,y=knn_pred18,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  376
##
##
##                     | knn_pred18
## test_sample_labels |         0 |       0.5 | Row Total |
## --------------------|-----------|-----------|-----------|
##                   0 |        18 |       110 |       128 |
##                     |     0.141 |     0.859 |     0.340 |
##                     |     0.545 |     0.321 |           |
##                     |     0.048 |     0.293 |           |
## --------------------|-----------|-----------|-----------|
##                 0.5 |        12 |       197 |       209 |
##                     |     0.057 |     0.943 |     0.556 |
##                     |     0.364 |     0.574 |           |
##                     |     0.032 |     0.524 |           |
## --------------------|-----------|-----------|-----------|
##                   1 |         3 |        34 |        37 |
##                     |     0.081 |     0.919 |     0.098 |
```

```
##                     |     0.091 |     0.099 |           |
##                     |     0.008 |     0.090 |           |
## --------------------|-----------|-----------|-----------|
##                   2 |         0 |         2 |         2 |
##                     |     0.000 |     1.000 |     0.005 |
##                     |     0.000 |     0.006 |           |
##                     |     0.000 |     0.005 |           |
## --------------------|-----------|-----------|-----------|
##       Column Total  |        33 |       343 |       376 |
##                     |     0.088 |     0.912 |           |
## --------------------|-----------|-----------|-----------|
##
##
```

```
library(caret) # Cargamos el paquete "caret"
knn_res1<-table(knn_pred1,test_sample_labels)
knn_res3<-table(knn_pred3,test_sample_labels)
knn_res5<-table(knn_pred5,test_sample_labels)
knn_res7<-table(knn_pred7,test_sample_labels)
knn_res10<-table(knn_pred10,test_sample_labels)
knn_res18<-table(knn_pred18,test_sample_labels)
knn_cmatrix1 <- confusionMatrix(knn_res1)
knn_cmatrix1
```

```
## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred1   0 0.5   1   2
##       0    40  62  11   1
##       0.5  79 130  21   0
##       1     9  17   5   1
##       2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.4654
##                  95% CI : (0.4141, 0.5173)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.0252
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.3125     0.6220  0.13514 0.000000
## Specificity            0.7016     0.4012  0.92035 1.000000
## Pos Pred Value         0.3509     0.5652  0.15625      NaN
## Neg Pred Value         0.6641     0.4589  0.90698 0.994681
## Prevalence             0.3404     0.5559  0.09840 0.005319
## Detection Rate         0.1064     0.3457  0.01330 0.000000
## Detection Prevalence   0.3032     0.6117  0.08511 0.000000
## Balanced Accuracy      0.5071     0.5116  0.52774 0.500000
```

```
knn_cmatrix3 <- confusionMatrix(knn_res3)
knn_cmatrix3
```

```
## Confusion Matrix and Statistics
##
##          test_sample_labels
## knn_pred3   0 0.5   1   2
##       0    38  43   5   1
##       0.5  84 150  30   1
##       1     6  16   2   0
##       2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5053
##                  95% CI : (0.4536, 0.557)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.9783
##
##                   Kappa : 0.0545
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.2969     0.7177 0.054054 0.000000
## Specificity            0.8024     0.3114 0.935103 1.000000
## Pos Pred Value         0.4368     0.5660 0.083333      NaN
## Neg Pred Value         0.6886     0.4685 0.900568 0.994681
```

```
## Prevalence             0.3404     0.5559 0.098404 0.005319
## Detection Rate         0.1011     0.3989 0.005319 0.000000
## Detection Prevalence   0.2314     0.7048 0.063830 0.000000
## Balanced Accuracy      0.5496     0.5145 0.494579 0.500000

knn_cmatrix5 <- confusionMatrix(knn_res5)
knn_cmatrix5

## Confusion Matrix and Statistics
##
##          test_sample_labels
## knn_pred5   0 0.5   1   2
##       0    39  37   4   1
##       0.5  86 167  32   1
##       1     3   5   1   0
##       2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5505
##                  95% CI : (0.4987, 0.6016)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.603
##
##                   Kappa : 0.1038
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.3047     0.7990  0.02703 0.000000
## Specificity            0.8306     0.2874  0.97640 1.000000
## Pos Pred Value         0.4815     0.5839  0.11111      NaN
## Neg Pred Value         0.6983     0.5333  0.90191 0.994681
## Prevalence             0.3404     0.5559  0.09840 0.005319
## Detection Rate         0.1037     0.4441  0.00266 0.000000
## Detection Prevalence   0.2154     0.7606  0.02394 0.000000
## Balanced Accuracy      0.5677     0.5432  0.50171 0.500000

knn_cmatrix7 <- confusionMatrix(knn_res7)
knn_cmatrix7
```

```
## Confusion Matrix and Statistics
##
##          test_sample_labels
## knn_pred7   0 0.5   1   2
##       0    33  39   3   0
##       0.5  95 167  33   2
##       1     0   3   1   0
##       2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5346
##                  95% CI : (0.4827, 0.5859)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.8113
##
##                   Kappa : 0.054
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.25781     0.7990  0.02703 0.000000
## Specificity           0.83065     0.2216  0.99115 1.000000
## Pos Pred Value        0.44000     0.5623  0.25000      NaN
## Neg Pred Value        0.68439     0.4684  0.90323 0.994681
## Prevalence            0.34043     0.5559  0.09840 0.005319
## Detection Rate        0.08777     0.4441  0.00266 0.000000
## Detection Prevalence  0.19947     0.7899  0.01064 0.000000
## Balanced Accuracy     0.54423     0.5103  0.50909 0.500000

knn_cmatrix10 <- confusionMatrix(knn_res10)
knn_cmatrix10

## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred10   0 0.5   1   2
##        0    30  26   4   0
##        0.5  98 182  33   2
##        1     0   1   0   0
##        2     0   0   0   0
```

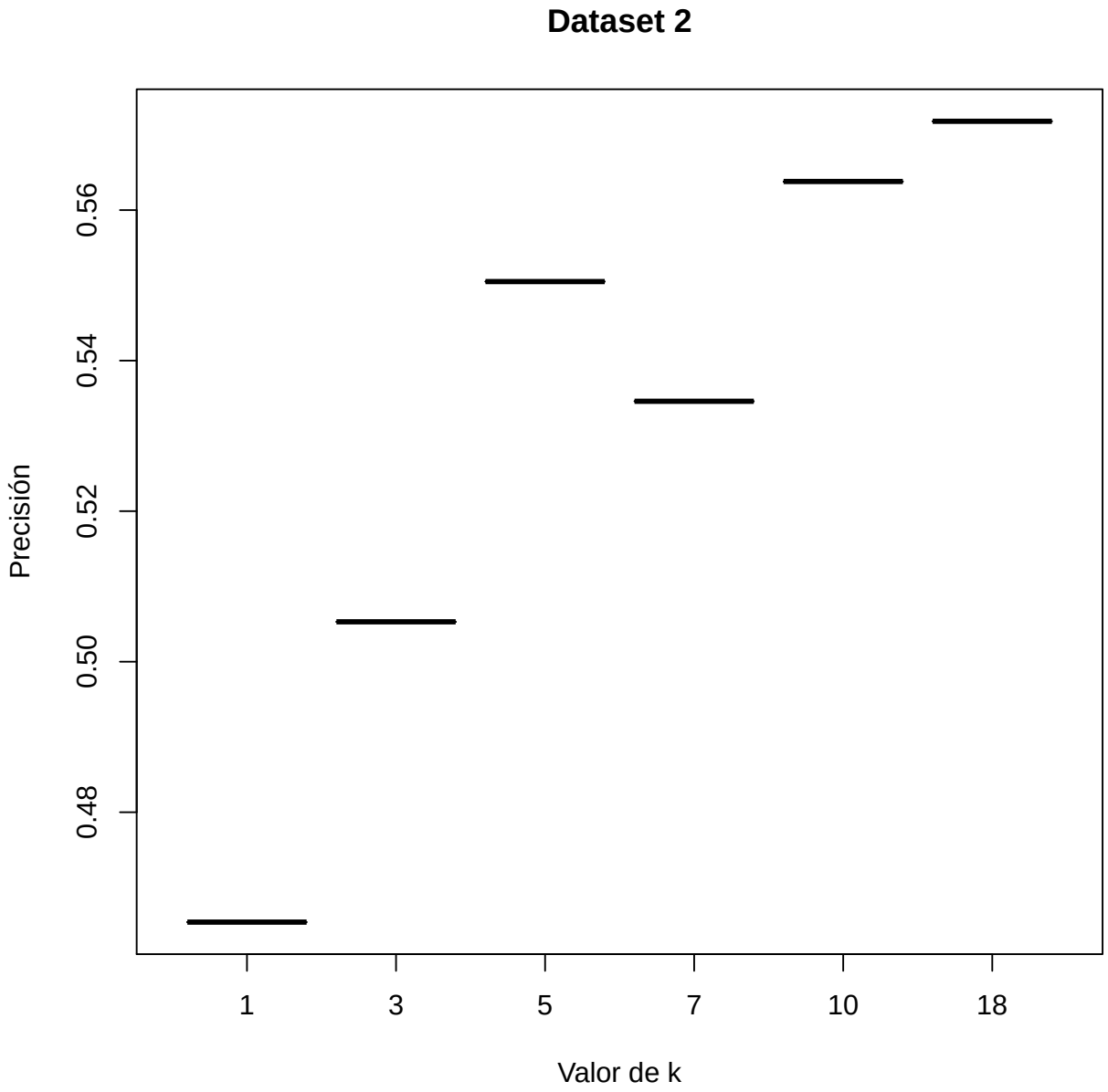
```
##
## Overall Statistics
##
##                Accuracy : 0.5638
##                  95% CI : (0.512, 0.6146)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.3984
##
##                   Kappa : 0.0908
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.23438     0.8708  0.00000 0.000000
## Specificity           0.87903     0.2036  0.99705 1.000000
## Pos Pred Value        0.50000     0.5778  0.00000      NaN
## Neg Pred Value        0.68987     0.5574  0.90133 0.994681
## Prevalence            0.34043     0.5559  0.09840 0.005319
## Detection Rate        0.07979     0.4840  0.00000 0.000000
## Detection Prevalence  0.15957     0.8378  0.00266 0.000000
## Balanced Accuracy     0.55670     0.5372  0.49853 0.500000
```

```
knn_cmatrix18 <- confusionMatrix(knn_res18)
knn_cmatrix18
```

```
## Confusion Matrix and Statistics
##
##           test_sample_labels
## knn_pred18   0 0.5   1   2
##        0    18  12   3   0
##        0.5 110 197  34   2
##        1     0   0   0   0
##        2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5718
##                  95% CI : (0.5201, 0.6224)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.2845
##
```

```
##                   Kappa : 0.0753
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.14062     0.9426   0.0000 0.000000
## Specificity           0.93952     0.1257   1.0000 1.000000
## Pos Pred Value        0.54545     0.5743      NaN      NaN
## Neg Pred Value        0.67930     0.6364   0.9016 0.994681
## Prevalence            0.34043     0.5559   0.0984 0.005319
## Detection Rate        0.04787     0.5239   0.0000 0.000000
## Detection Prevalence  0.08777     0.9122   0.0000 0.000000
## Balanced Accuracy     0.54007     0.5342   0.5000 0.500000
```

```
# Representación de la precisión:

k<-factor(c("1","3","5","7","10","18"),levels=c("1","3","5","7","10","18"))
precision<-c(0.4654,0.5053,0.5505,0.5346,0.5638,0.5718)
plot(k,precision,type="p",xlab="Valor de k",ylab="Precisión",main="Dataset 2",
color="blue")
```

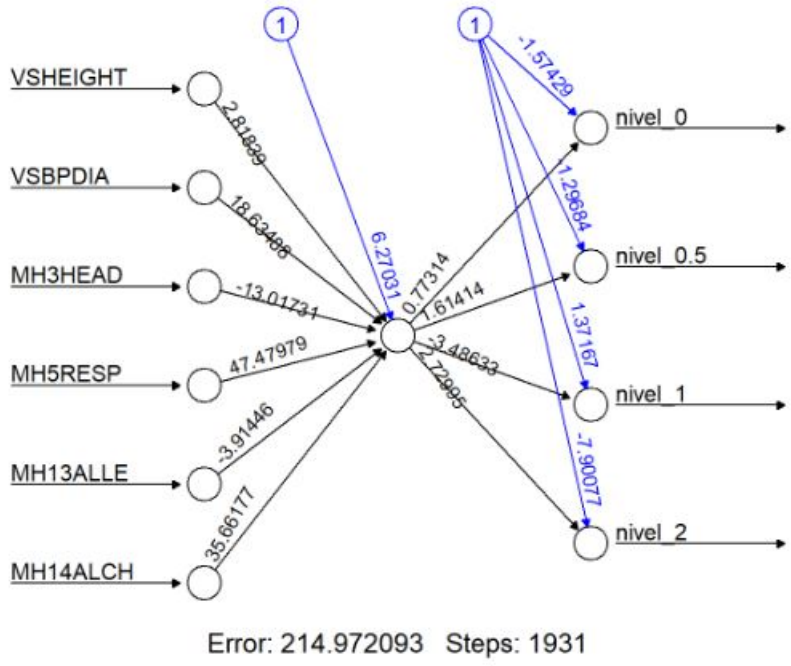



```
#++++++++++++++++++++++++++++++++++#
# Algoritmo Red Neuronal Artificial #
#++++++++++++++++++++++++++++++++++#

# Preparación de los datos #

# Dividimos la variable "cdrglob" en 5 variables según sus posibles resultados:
cdr_vitals_medhist_norm$nivel_0<-cdr_vitals_medhist$CDGLOBAL==0
```

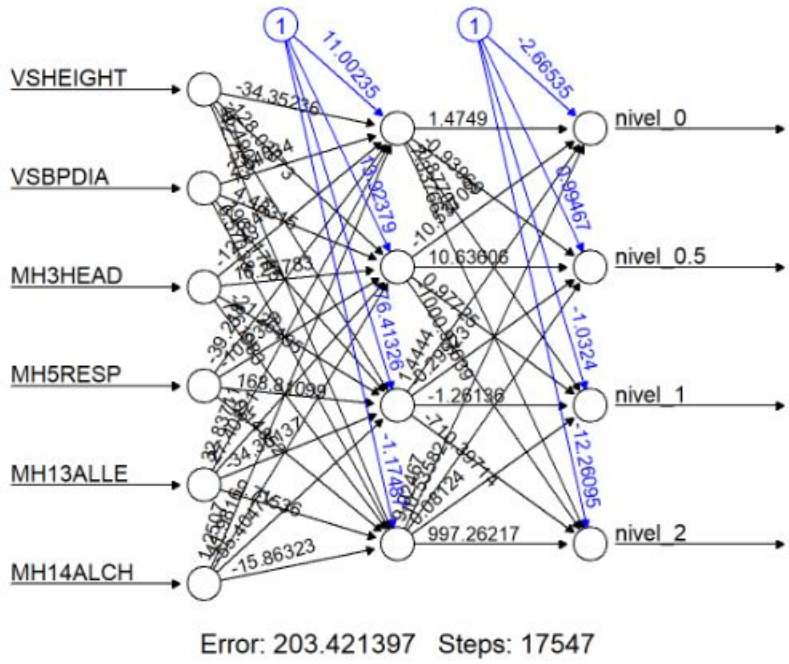
```
cdr_vitals_medhist_norm$nivel_0.5<-cdr_vitals_medhist$CDGLOBAL==0.5
cdr_vitals_medhist_norm$nivel_1<-cdr_vitals_medhist$CDGLOBAL==1
cdr_vitals_medhist_norm$nivel_2<-cdr_vitals_medhist$CDGLOBAL==2
str(cdr_vitals_medhist_norm)
```

```
## 'data.frame':    1138 obs. of  10 variables:
##  $ VSHEIGHT : num  0.385 0.374 0.374 0.385 0.349 ...
##  $ VSBPDIA  : num  0.73 0.64 0.64 0.73 0.685 ...
##  $ MH3HEAD  : num  0 1 1 0 1 1 0 0 1 1 ...
##  $ MH5RESP  : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ MH13ALLE : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ MH14ALCH : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ nivel_0  : logi  TRUE FALSE FALSE TRUE FALSE TRUE ...
##  $ nivel_0.5: logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ nivel_1  : logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
##  $ nivel_2  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
# Separación de los datos normalizados:
train_sample_norm<-cdr_vitals_medhist_norm[train_ind, ] # Datos de entrenamiento
test_sample_norm<-cdr_vitals_medhist_norm[-train_ind, ] # Datos de prueba

# Entrenamiento del modelo #

# Cargamos el paquete "neuralnet":
library(neuralnet)
# Creamos el modelo de 1 nodo:
red_pred1<-neuralnet(nivel_0 + nivel_0.5 + nivel_1 + nivel_2 ~ VSHEIGHT +
VSBPDIA + MH3HEAD + MH5RESP + MH13ALLE + MH14ALCH, data = train_sample_norm,
                           hidden=1,linear.output=FALSE)
# Representamos el modelo de 1 nodo:
plot(red_pred1,rep="best")
```




```
# Creamos el modelo de 5 nodos:
red_pred2<-neuralnet(nivel_0 + nivel_0.5 + nivel_1 + nivel_2 ~ VSHEIGHT +
VSBPDIA + MH3HEAD + MH5RESP + MH13ALLE + MH14ALCH, data = train_sample_norm,
                          hidden=4,linear.output=FALSE)
# Representamos el modelo de 5 nodos:
plot(red_pred2,rep="best")
```




```
# Predicción y evaluación del modelo #

# Realizamos la predicción con el modelo de 1 nodo:
red_pred1_results <- neuralnet::compute(red_pred1, test_sample_norm[,1:7])$net.result
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx_1 <- apply(red_pred1_results, 1, maxidx)
prediction_1<- c("0","0.5","1","2")[idx_1]
res_1 <- table(prediction_1, cdr_vitals_medhist$CDGLOBAL[-train_ind])
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-rbind(res_1,c(0,0,0,0,0))
res_1 <-res_1[c(2,1,3,4),]
rownames(res_1) <-c("0","0.5","1","2")
res_1
# Realizamos la evaluación del rendimiento del modelo:
library(caret)
require(caret)
cmatrix1 <- confusionMatrix(res_1)
cmatrix1
```

```
## Confusion Matrix and Statistics
##
##       0 0.5   1   2
## 0     1   3   0   0
## 0.5 127 206  37   2
## 1     0   0   0   0
## 2     0   0   0   0
##
## Overall Statistics
##
##               Accuracy : 0.5505
##                 95% CI : (0.4987, 0.6016)
##    No Information Rate : 0.5559
##    P-Value [Acc > NIR] : 0.603
##
##                  Kappa : -0.0068
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity          0.007812   0.985646   0.0000 0.000000
## Specificity          0.987903   0.005988   1.0000 1.000000
## Pos Pred Value       0.250000   0.553763      NaN      NaN
## Neg Pred Value       0.658602   0.250000   0.9016 0.994681
## Prevalence           0.340426   0.555851   0.0984 0.005319
## Detection Rate       0.002660   0.547872   0.0000 0.000000
## Detection Prevalence 0.010638   0.989362   0.0000 0.000000
## Balanced Accuracy    0.497858   0.495817   0.5000 0.500000
```

```
# Realizamos la predicción con el modelo de 5 nodos:
red_pred2_results <- neuralnet::compute(red_pred2, test_sample_norm[,1:7])$net.result
maxidx2 <- function(arr) {
  return(which(arr == max(arr)))
}
idx_2 <- apply(red_pred2_results, 1, maxidx2)
prediction_2<- c("0","0.5","1","2")[idx_2]
res_2 <- table(prediction_2, cdr_vitals_medhist$CDGLOBAL[-train_ind])
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-rbind(res_2,c(0,0,0,0,0))
res_2 <-res_2[c(2,1,3,4),]
rownames(res_2) <-c("0","0.5","1","2")
res_2

# Realizamos la evaluación del rendimiento del modelo:
library(caret)
require(caret)
cmatrix2 <- confusionMatrix(res_2)
cmatrix2
```

```
## Confusion Matrix and Statistics
##
##       0 0.5   1   2
## 0   119 196  35   2
## 0.5   8  13   2   0
## 1     1   0   0   0
## 2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.3511
##                  95% CI : (0.3028, 0.4017)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0029
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.92969    0.06220  0.00000 0.000000
## Specificity           0.06048    0.94012  0.99705 1.000000
## Pos Pred Value        0.33807    0.56522  0.00000      NaN
## Neg Pred Value        0.62500    0.44476  0.90133 0.994681
## Prevalence            0.34043    0.55585  0.09840 0.005319
## Detection Rate        0.31649    0.03457  0.00000 0.000000
## Detection Prevalence  0.93617    0.06117  0.00266 0.000000
## Balanced Accuracy     0.49509    0.50116  0.49853 0.500000
```

```
#++++++++++++++++++++++++#
# Algoritmo Naive Bayes #
#++++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos el paquete "e1071":
library(e1071)
# Creamos el modelo:
naive_bayes<-naiveBayes(train_sample,train_sample_labels,laplace=0)
# Visualizamos tablas formadas en nuestro modelo:
naive_bayes$tables

## $VSHEIGHT
##                       VSHEIGHT
## train_sample_labels     [,1]     [,2]
##                   0   49.68712 50.96961
##                   0.5 49.33649 52.97730
##                   1   37.48478 52.17475
##                   2   13.75000 35.50000
##
```

```
## $VSBPDIA
##                      VSBPDIA
## train_sample_labels     [,1]      [,2]
##                 0   75.20172  9.400917
##                 0.5 74.19169 11.423973
##                 1   72.80435 10.009050
##                 2   73.50000  4.123106
##
## $MH3HEAD
##                      MH3HEAD
## train_sample_labels      [,1]      [,2]
##                 0   0.5407725 0.4994077
##                 0.5 0.6073903 0.4888960
##                 1   0.7391304 0.4415150
##                 2   0.7500000 0.5000000
##
## $MH5RESP
##                      MH5RESP
## train_sample_labels      [,1]      [,2]
##                 0   0.1587983 0.3662747
##                 0.5 0.2355658 0.4248428
##                 1   0.2826087 0.4527350
##                 2   0.0000000 0.0000000
##
## $MH13ALLE
##                      MH13ALLE
## train_sample_labels      [,1]      [,2]
##                 0   0.4034335 0.4916424
##                 0.5 0.3579677 0.4799571
##                 1   0.4456522 0.4997611
##                 2   0.0000000 0.0000000
##
## $MH14ALCH
##                      MH14ALCH
## train_sample_labels       [,1]      [,2]
##                 0   0.01716738 0.1301745
##                 0.5 0.03464203 0.1830830
##                 1   0.03260870 0.1785834
##                 2   0.50000000 0.5773503

# Predicción y evaluación del modelo #

# Realizamos la predicción de los datos de prueba utilizando el modelo:
```

```
naive_bayes_pred<-predict(naive_bayes,test_sample)
# Evaluamos su eficiencia mediante una matriz de confusión:
naive_bayes_res <- table(naive_bayes_pred, cdr_vitals_medhist$CDGLOBAL[-train_ind])
library(caret)
naive_bayes_cmatrix <- confusionMatrix(naive_bayes_res)
# Mostramos el resultado:
naive_bayes_cmatrix

## Confusion Matrix and Statistics
##
##
## naive_bayes_pred  0 0.5  1  2
##              0   31  35  9  0
##              0.5 38  75 14  2
##              1    0   0  0  0
##              2   59  99 14  0
##
## Overall Statistics
##
##                Accuracy : 0.2819
##                  95% CI : (0.237, 0.3303)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0282
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.24219     0.3589   0.0000 0.000000
## Specificity           0.82258     0.6766   1.0000 0.540107
## Pos Pred Value        0.41333     0.5814      NaN 0.000000
## Neg Pred Value        0.67774     0.4575   0.9016 0.990196
## Prevalence            0.34043     0.5559   0.0984 0.005319
## Detection Rate        0.08245     0.1995   0.0000 0.000000
## Detection Prevalence  0.19947     0.3431   0.0000 0.457447
## Balanced Accuracy     0.53238     0.5177   0.5000 0.270053

# Probaremos un nuevo valor de la opción "laplace" #

# Creamos el nuevo modelo:
```

```
naive_bayes2<-naiveBayes(train_sample,train_sample_labels,laplace=1)
# Mostramos las tablas formadas en el modelo:
naive_bayes2$tables
```

```
## $VSHEIGHT
##                       VSHEIGHT
## train_sample_labels     [,1]     [,2]
##                 0   49.68712 50.96961
##                 0.5 49.33649 52.97730
##                 1   37.48478 52.17475
##                 2   13.75000 35.50000
##
## $VSBPDIA
##                       VSBPDIA
## train_sample_labels     [,1]      [,2]
##                 0   75.20172  9.400917
##                 0.5 74.19169 11.423973
##                 1   72.80435 10.009050
##                 2   73.50000  4.123106
##
## $MH3HEAD
##                       MH3HEAD
## train_sample_labels      [,1]      [,2]
##                 0   0.5407725 0.4994077
##                 0.5 0.6073903 0.4888960
##                 1   0.7391304 0.4415150
##                 2   0.7500000 0.5000000
##
## $MH5RESP
##                       MH5RESP
## train_sample_labels      [,1]      [,2]
##                 0   0.1587983 0.3662747
##                 0.5 0.2355658 0.4248428
##                 1   0.2826087 0.4527350
##                 2   0.0000000 0.0000000
##
## $MH13ALLE
##                       MH13ALLE
## train_sample_labels      [,1]      [,2]
##                 0   0.4034335 0.4916424
##                 0.5 0.3579677 0.4799571
##                 1   0.4456522 0.4997611
##                 2   0.0000000 0.0000000
```

```
##
## $MH14ALCH
##                     MH14ALCH
## train_sample_labels       [,1]      [,2]
##                 0   0.01716738 0.1301745
##                 0.5 0.03464203 0.1830830
##                 1   0.03260870 0.1785834
##                 2   0.50000000 0.5773503
```

```
# Realizamos la predicción:
naive_bayes_pred2<-predict(naive_bayes2,test_sample)
# Analizamos el rendimiento del mismo mediante una matriz de confusión:
naive_bayes_res2 <- table(naive_bayes_pred2, cdr_vitals_medhist$CDGLOBAL[-train_ind])
library(caret)
naive_bayes_cmatrix2 <- confusionMatrix(naive_bayes_res2)
# Mostramos el resultado:
naive_bayes_cmatrix2
```

```
## Confusion Matrix and Statistics
##
##
## naive_bayes_pred2  0 0.5  1  2
##               0   31  35  9  0
##               0.5 38  75 14  2
##               1    0   0  0  0
##               2   59  99 14  0
##
## Overall Statistics
##
##                Accuracy : 0.2819
##                  95% CI : (0.237, 0.3303)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0282
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity           0.24219     0.3589   0.0000 0.000000
## Specificity           0.82258     0.6766   1.0000 0.540107
```

```
## Pos Pred Value         0.41333     0.5814      NaN 0.000000
## Neg Pred Value         0.67774     0.4575   0.9016 0.990196
## Prevalence             0.34043     0.5559   0.0984 0.005319
## Detection Rate         0.08245     0.1995   0.0000 0.000000
## Detection Prevalence   0.19947     0.3431   0.0000 0.457447
## Balanced Accuracy      0.53238     0.5177   0.5000 0.270053

#++++++++++++++++++++++++++++++++++#
# Algoritmo Support Vector Machine #
#++++++++++++++++++++++++++++++++++#

# Preparación de datos #
set.seed(123)
cdr_vitals_medhist$CDGLOBAL<-as.factor(cdr_vitals_medhist$CDGLOBAL)
train_ind2<-sample(seq_len(nrow(cdr_vitals_medhist)),size = smp_size)
train_sample2<-cdr_vitals_medhist[train_ind2, ] # Datos de entrenamiento 2
test_sample2<-cdr_vitals_medhist[-train_ind2, ] # Datos de prueba 2

# Entrenamiento del modelo #

# Cargamos el paquete "kernlab":
library(kernlab)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos un modelo con la función lineal:
svm <- ksvm(CDGLOBAL~.,data=train_sample2, kernel='vanilladot')

##  Setting default kernel parameters

# Mostramos la información básica del modelo:
svm

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 635
##
## Objective Function Value : -490 -166 -12 -166 -12 -12
## Training error : 0.43832
```

```r
# Predicción y evaluación del modelo #

# LLevamos a cabo la predicción de nuestros datos de prueba utilizando el modelo SVM
# de la función lineal:
svm_pred<-predict(svm,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
library(caret)
svm_res<-table(svm_pred,test_sample2$CDGLOBAL)
svm_cmatrix<-confusionMatrix(svm_res)
# Mostramos el resultado:
svm_cmatrix
```

```
## Confusion Matrix and Statistics
##
##
## svm_pred   0 0.5   1   2
##      0     0   0   0   0
##      0.5 116 214  46   0
##      1     0   0   0   0
##      2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5691
##                  95% CI : (0.5174, 0.6198)
##     No Information Rate : 0.5691
##     P-Value [Acc > NIR] : 0.5217
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.0000     1.0000   0.0000       NA
## Specificity            1.0000     0.0000   1.0000        1
## Pos Pred Value            NaN     0.5691      NaN       NA
## Neg Pred Value         0.6915        NaN   0.8777       NA
## Prevalence             0.3085     0.5691   0.1223        0
## Detection Rate         0.0000     0.5691   0.0000        0
## Detection Prevalence   0.0000     1.0000   0.0000        0
## Balanced Accuracy      0.5000     0.5000   0.5000       NA
```

```r
# Mejora del modelo #

# Cargamos el paquete "kernlab":
library(kernlab)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos un modelo con la función lineal:
svm_rbf <- ksvm(CDGLOBAL~.,data=train_sample2, kernel='rbfdot')
# Mostramos la información básica del modelo:
svm_rbf

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.248148832525565
##
## Number of Support Vectors : 669
##
## Objective Function Value : -480.2763 -158.2611 -11.2365 -164.7989 -11.3681 -11.4984
## Training error : 0.423885

# LLevamos a cabo la predicción de nuestros datos de prueba utilizando el modelo SVM
# de la función lineal:
svm_rbf_pred<-predict(svm_rbf,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
library(caret)
svm_rbf_res<-table(svm_rbf_pred,test_sample2$CDGLOBAL)
svm_rbf_cmatrix<-confusionMatrix(svm_rbf_res)
svm_rbf_cmatrix

## Confusion Matrix and Statistics
##
##
## svm_rbf_pred   0 0.5   1   2
##          0     1   3   0   0
##          0.5 115 211  46   0
##          1     0   0   0   0
##          2     0   0   0   0
##
## Overall Statistics
```

```
##
##                Accuracy : 0.5638
##                  95% CI : (0.512, 0.6146)
##     No Information Rate : 0.5691
##     P-Value [Acc > NIR] : 0.6035
##
##                   Kappa : -0.0059
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity          0.008621   0.985981   0.0000       NA
## Specificity          0.988462   0.006173   1.0000        1
## Pos Pred Value       0.250000   0.567204      NaN       NA
## Neg Pred Value       0.690860   0.250000   0.8777       NA
## Prevalence           0.308511   0.569149   0.1223        0
## Detection Rate       0.002660   0.561170   0.0000        0
## Detection Prevalence 0.010638   0.989362   0.0000        0
## Balanced Accuracy    0.498541   0.496077   0.5000       NA
```

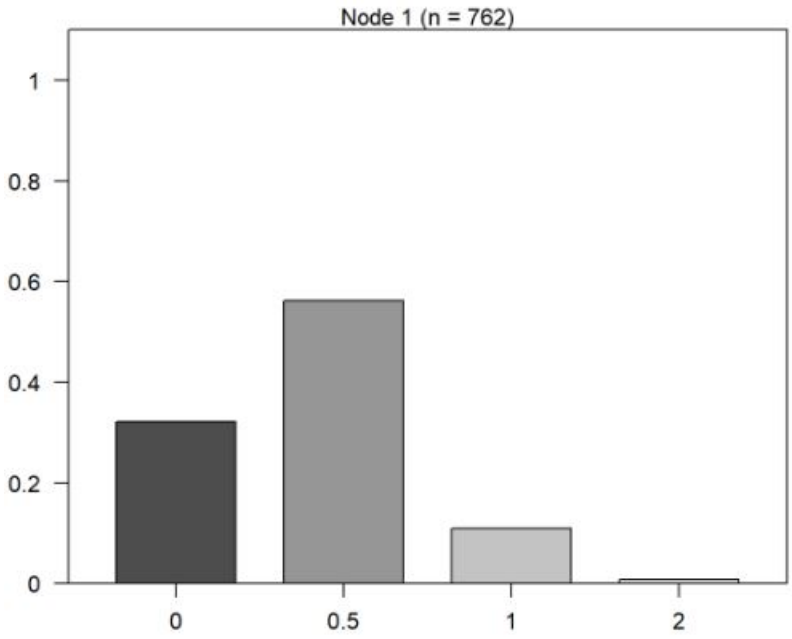
```
#++++++++++++++++++++++++++++++++#
# Algoritmo Árbol de Decisión  #
#++++++++++++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos las librerías necesarias para llevar a cabo este algoritmo:
library(C50)
library(caret)
library(tidyverse)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos el modelo:
train_sample2$CDGLOBAL<-as.factor(train_sample2$CDGLOBAL)
arbol<-C5.0(train_sample2[-1],train_sample2$CDGLOBAL)
# Mostramos la información del modelo:
arbol
```

```
##
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$CDGLOBAL)
##
## Classification Tree
## Number of samples: 762
## Number of predictors: 6
##
## Tree size: 1
##
## Non-standard options: attempt to group attributes
```

```
# Hacemos un resumen estadístico del modelo:
summary(arbol)
```

```
##
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$CDGLOBAL)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Dec 23 21:03:49 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 762 cases (7 attributes) from undefined.data
##
## Decision tree:
##  0.5 (762/334)
##
##
## Evaluation on training data (762 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       1  334(43.8%)   <<
##
##
##     (a)   (b)   (c)   (d)    <-classified as
##    ----  ----  ----  ----
##           245                (a): class 0
##           428                (b): class 0.5
##            83                (c): class 1
##             6                (d): class 2
##
##
## Time: 0.0 secs
```

```
# Realizamos una representación gráfica del modelo:
plot(arbol)
```




```
# Predicción y evaluación del modelo #

# Realizamos la prediccion de los datos de prueba utilizando el modelo creado:
arbol_pred<-predict(arbol,test_sample2)
# Calculamos la matriz de confusión que nos mostrará su eficiencia:
arbol_res<-table(arbol_pred,test_sample_labels)
arbol_cmatrix<-confusionMatrix(arbol_res)
# Mostramos la matriz de confusión:
arbol_cmatrix
```

```
## Confusion Matrix and Statistics
##
##           test_sample_labels
## arbol_pred   0 0.5   1   2
##        0     0   0   0   0
##        0.5 128 209  37   2
##        1     0   0   0   0
##        2     0   0   0   0
##
## Overall Statistics
##
##               Accuracy : 0.5559
##                 95% CI : (0.504, 0.6068)
##    No Information Rate : 0.5559
##    P-Value [Acc > NIR] : 0.5215
##
##                  Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.0000     1.0000   0.0000 0.000000
## Specificity            1.0000     0.0000   1.0000 1.000000
## Pos Pred Value            NaN     0.5559      NaN      NaN
## Neg Pred Value         0.6596        NaN   0.9016 0.994681
## Prevalence             0.3404     0.5559   0.0984 0.005319
## Detection Rate         0.0000     0.5559   0.0000 0.000000
## Detection Prevalence   0.0000     1.0000   0.0000 0.000000
## Balanced Accuracy      0.5000     0.5000   0.5000 0.500000
```

```
# Mejora del modelo #

# Creamos el nuevo modelo:
boost10<-C5.0(train_sample2[-1],train_sample2$CDGLOBAL,trials=10)
# Mostramos la información básica del modelo:
boost10
```

```
##
## Call:
## C5.0.default(x = train_sample2[-1], y = train_sample2$CDGLOBAL, trials = 10)
##
## Classification Tree
## Number of samples: 762
## Number of predictors: 6
##
## Number of boosting iterations: 10 requested;  1 used due to early stopping
##
## Non-standard options: attempt to group attributes
```

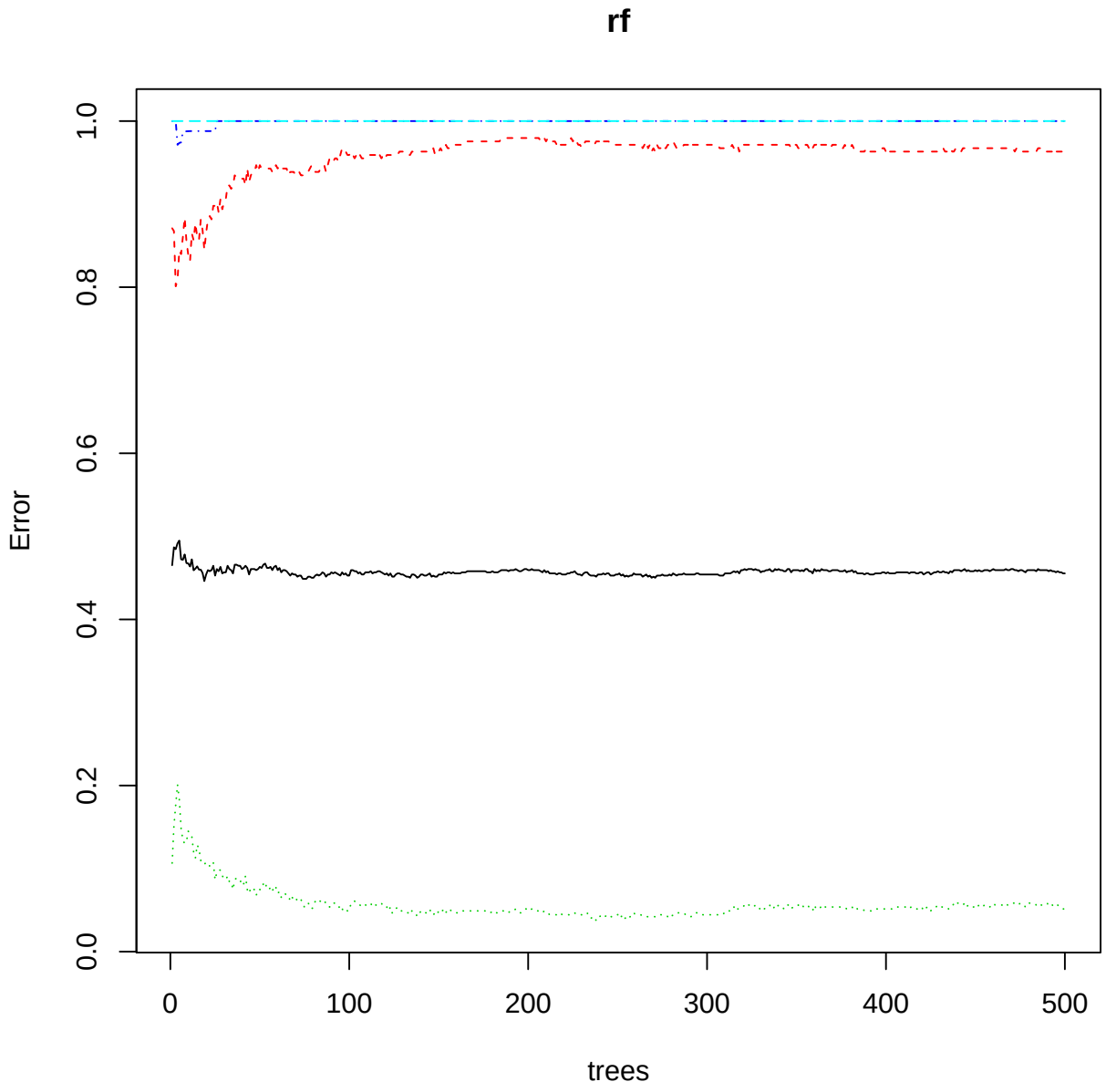
```
# Realizamos la predicción utilizando el nuevo modelo:
boost10_pred<-predict(boost10,test_sample2)
# Realizamos la comparativa de los datos predichos con los reales utilizando
# una matriz de confusión:
boost10_res<-table(boost10_pred,test_sample_labels)
```

```
boost10_cmatrix<-confusionMatrix(boost10_res)
# Mostramos el resultado:
boost10_cmatrix
```

```
## Confusion Matrix and Statistics
##
##              test_sample_labels
## boost10_pred   0 0.5   1   2
##          0     0   0   0   0
##          0.5 128 209  37   2
##          1     0   0   0   0
##          2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5559
##                  95% CI : (0.504, 0.6068)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.5215
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity            0.0000     1.0000   0.0000 0.000000
## Specificity            1.0000     0.0000   1.0000 1.000000
## Pos Pred Value            NaN     0.5559      NaN      NaN
## Neg Pred Value         0.6596        NaN   0.9016 0.994681
## Prevalence             0.3404     0.5559   0.0984 0.005319
## Detection Rate         0.0000     0.5559   0.0000 0.000000
## Detection Prevalence   0.0000     1.0000   0.0000 0.000000
## Balanced Accuracy      0.5000     0.5000   0.5000 0.500000
```

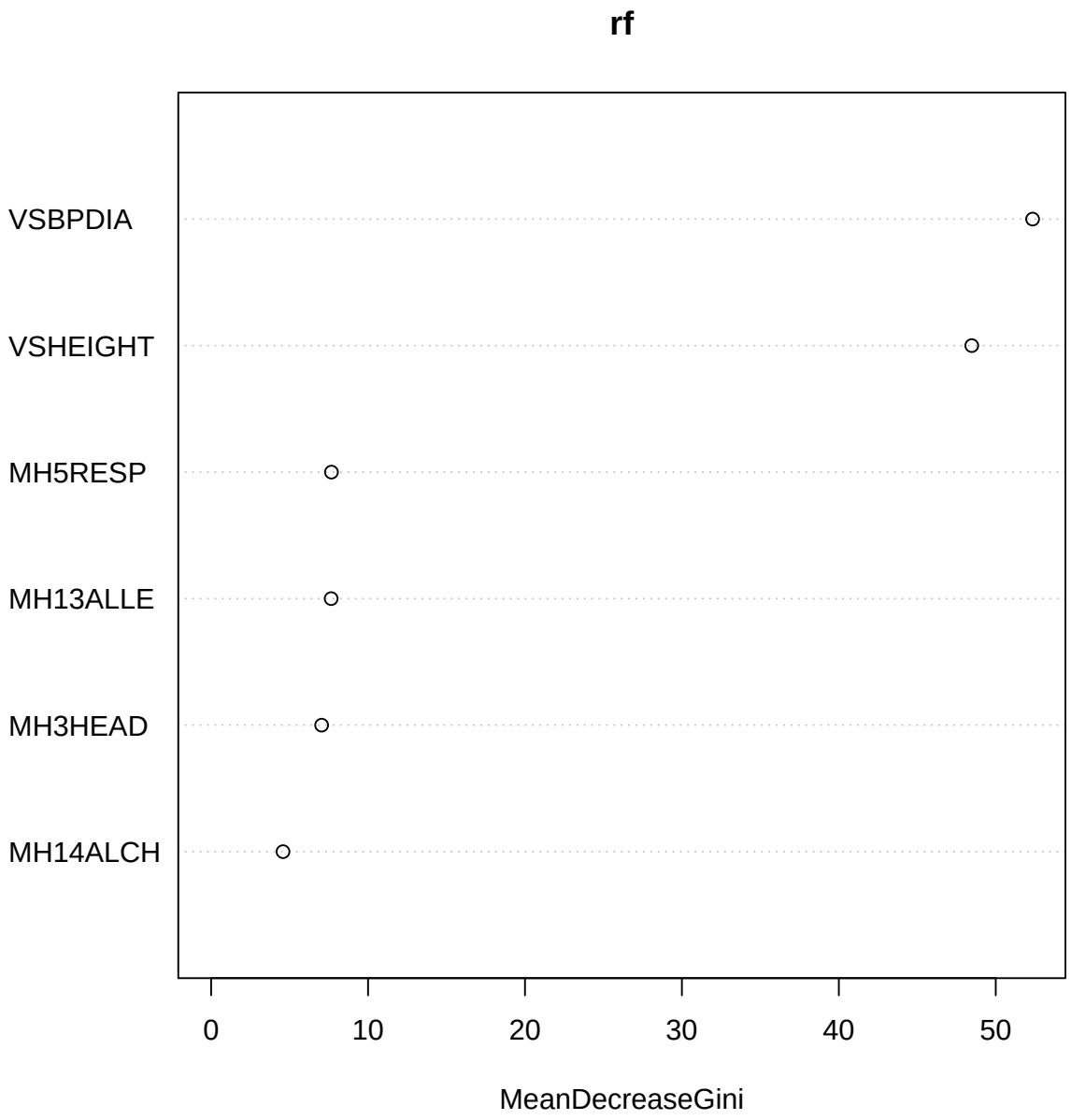
```
#+++++++++++++++++++++++++#
# Algoritmo Random Forest #
#+++++++++++++++++++++++++#

# Entrenamiento del modelo #

# Importamos el paquete "randomForest":
library(randomForest)
# Fijamos la semilla aleatoria:
set.seed(123)
# Creamos el modelo:
rf <- randomForest(CDGLOBAL ~ ., data = train_sample2)
# Mostramos el modelo:
print(rf)

##
```

```
## Call:
##  randomForest(formula = CDGLOBAL ~ ., data = train_sample2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 45.54%
## Confusion matrix:
##      0 0.5 1 2 class.error
## 0    9 236 0 0  0.96326531
## 0.5 22 406 0 0  0.05140187
## 1    2  81 0 0  1.00000000
## 2    0   6 0 0  1.00000000

# Representamos gráficamente el modelo:
plot(rf)
```




```
# Mostramos la importancia de las variables:
importance(rf)

##          MeanDecreaseGini
## VSHEIGHT        48.470422
## VSBPDIA         52.347471
## MH3HEAD          7.042503
## MH5RESP          7.665516
## MH13ALLE         7.648112
```

```
## MH14ALCH         4.580542

# Representamos gráficamente la importancia de las variables:
varImpPlot(rf)
```




```
# Predicción y evaluación del modelo #

# Realizamos la predicción:
```

```
rf_pred<-predict(rf,test_sample2[-9])
# Calculamos la matriz de confusión:
rf_res<-table(rf_pred,test_sample_labels)
rf_cmatrix<-confusionMatrix(rf_res)
# Mostramos el resultado:
rf_cmatrix
```

```
## Confusion Matrix and Statistics
##
##        test_sample_labels
## rf_pred   0 0.5   1   2
##     0     1   2   0   0
##     0.5 127 207  37   2
##     1     0   0   0   0
##     2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5532
##                  95% CI : (0.5014, 0.6042)
##     No Information Rate : 0.5559
##     P-Value [Acc > NIR] : 0.5626
##
##                   Kappa : -0.0021
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity          0.007812   0.990431   0.0000 0.000000
## Specificity          0.991935   0.005988   1.0000 1.000000
## Pos Pred Value       0.333333   0.554960      NaN      NaN
## Neg Pred Value       0.659517   0.333333   0.9016 0.994681
## Prevalence           0.340426   0.555851   0.0984 0.005319
## Detection Rate       0.002660   0.550532   0.0000 0.000000
## Detection Prevalence 0.007979   0.992021   0.0000 0.000000
## Balanced Accuracy    0.499874   0.498209   0.5000 0.500000
```

```
# Mejora del modelo #

# Fijamos la semilla aleatoria:
set.seed(123)
```

```
# Creamos el modelo:
rf.1000<-randomForest(CDGLOBAL ~ ., data = train_sample2,ntree=1000)
# Mostramos la información básica del modelo:
rf.1000

##
## Call:
##  randomForest(formula = CDGLOBAL ~ ., data = train_sample2, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 45.67%
## Confusion matrix:
##      0 0.5 1 2 class.error
## 0    7 238 0 0  0.97142857
## 0.5 21 407 0 0  0.04906542
## 1    3  80 0 0  1.00000000
## 2    0   6 0 0  1.00000000

# Realizamos la predicción:
rf.1000_pred<-predict(rf.1000,test_sample2[-9])
# Contrastamos el rendimiento del mismo con resultados verdaderos de los datos de prueb
# utilizando una matriz de confusión:
rf.1000_res<-table(rf.1000_pred,test_sample_labels)
rf.1000_cmatrix<-confusionMatrix(rf.1000_res)
# Mostramos los resultados:
rf.1000_cmatrix

## Confusion Matrix and Statistics
##
##             test_sample_labels
## rf.1000_pred   0 0.5   1   2
##          0     0   2   1   0
##          0.5 128 207  36   2
##          1     0   0   0   0
##          2     0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5505
##                  95% CI : (0.4987, 0.6016)
##     No Information Rate : 0.5559
```

```
##     P-Value [Acc > NIR] : 0.603
##
##                   Kappa : -0.0081
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1 Class: 2
## Sensitivity          0.000000   0.990431   0.0000 0.000000
## Specificity          0.987903   0.005988   1.0000 1.000000
## Pos Pred Value       0.000000   0.554960      NaN      NaN
## Neg Pred Value       0.656836   0.333333   0.9016 0.994681
## Prevalence           0.340426   0.555851   0.0984 0.005319
## Detection Rate       0.000000   0.550532   0.0000 0.000000
## Detection Prevalence 0.007979   0.992021   0.0000 0.000000
## Balanced Accuracy    0.493952   0.498209   0.5000 0.500000
```

Código Python (Tensorflow):

```
# Importamos todas las librerías que utilizaremos
from scipy.sparse.construct import random
import numpy as np
import cv2
import os
import PIL
from PIL import Image
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from keras.utils.np_utils import normalize
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation,Dropout,Flatten,Dense
# Creamos las variables que utilizaremos posteriormente:
direccion_imagen="Datasets/"
CN=os.listdir(direccion_imagen+"CN/")
MCI=os.listdir(direccion_imagen+"MCI/")
AD=os.listdir(direccion_imagen+"AD/")
dataset=[]
```

```
label=[]
input_size=256
from numpy.random import seed
seed(123)
tf.random.set_seed(123)
import random
random.seed(123)
# Importamos los datos de las tres categorías:
for i, nombre_imagen in enumerate(CN):
  if(nombre_imagen.split(".")[1]=="jpg"):
    imagen=cv2.imread(direccion_imagen+"CN/"+nombre_imagen)
    imagen=Image.fromarray(imagen,"RGB")
    imagen.resize((input_size,input_size))
    dataset.append(np.array(imagen))
    label.append(0)
for i, nombre_imagen in enumerate(MCI):
  if(nombre_imagen.split(".")[1]=="jpg"):
    imagen=cv2.imread(direccion_imagen+"MCI/"+nombre_imagen)
    imagen=Image.fromarray(imagen,"RGB")
    imagen.resize((input_size,input_size))
    dataset.append(np.array(imagen))
    label.append(1)
for i, nombre_imagen in enumerate(AD):
  if(nombre_imagen.split(".")[1]=="jpg"):
    imagen=cv2.imread(direccion_imagen+"AD/"+nombre_imagen)
    imagen=Image.fromarray(imagen,"RGB")
    imagen.resize((input_size,input_size))
    dataset.append(np.array(imagen))
    label.append(2)
# Creamos los conjuntos de datos de entrenamiento (train) y de prueba (test)
# junto con sus etiquetas, los cuales normalizaremos:
dataset=np.array(dataset)
label=np.array(label)
x_train, x_test, y_train, y_test=train_test_split(dataset,label,test_size=0.3,
                                                  random_state=0)
x_train=normalize(x_train,axis=1)
x_test=normalize(x_test,axis=1)
y_train=to_categorical(y_train,num_classes=3)
y_test=to_categorical(y_test,num_classes=3)

# Constuimos el modelo:
modelo=Sequential()
modelo.add(Conv2D(32,(3,3),input_shape=(input_size,input_size,3)))
```

```
modelo.add(Activation("relu"))
modelo.add(MaxPooling2D(pool_size=(2,2)))

modelo.add(Conv2D(32,(3,3),kernel_initializer="he_uniform"))
modelo.add(Activation("relu"))
modelo.add(MaxPooling2D(pool_size=(2,2)))

modelo.add(Conv2D(32,(3,3),kernel_initializer="he_uniform"))
modelo.add(Activation("relu"))
modelo.add(MaxPooling2D(pool_size=(2,2)))

modelo.add(Conv2D(32,(3,3),kernel_initializer="he_uniform"))
modelo.add(Activation("relu"))
modelo.add(MaxPooling2D(pool_size=(2,2)))

modelo.add(Flatten())
modelo.add(Dense(64))
modelo.add(Activation("relu"))
modelo.add(Dropout(0.5))
modelo.add(Dense(3))
modelo.add(Activation("softmax"))

# Mostramos un resumen del modelo:
modelo.summary()
```

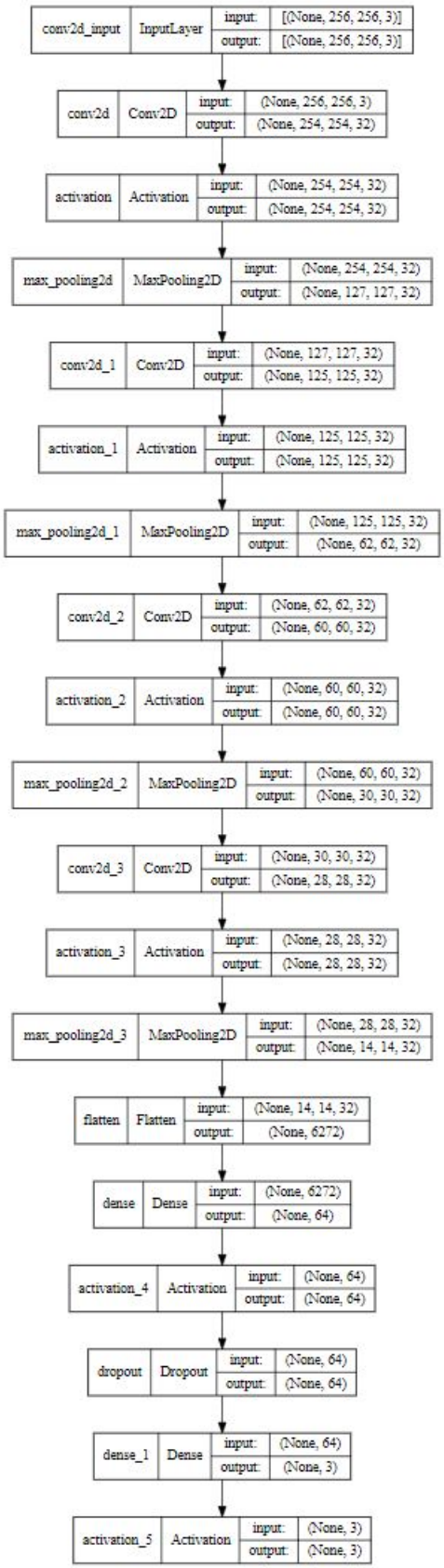
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 254, 254, 32)      896

 activation_15 (Activation)  (None, 254, 254, 32)      0

 max_pooling2d_9 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_10 (Conv2D)          (None, 125, 125, 32)      9248

 activation_16 (Activation)  (None, 125, 125, 32)      0

 max_pooling2d_10 (MaxPoolin  (None, 62, 62, 32)       0
 g2D)

 conv2d_11 (Conv2D)          (None, 60, 60, 32)        9248

 activation_17 (Activation)  (None, 60, 60, 32)        0

 max_pooling2d_11 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_12 (Conv2D)          (None, 28, 28, 32)        9248

 activation_18 (Activation)  (None, 28, 28, 32)        0

 max_pooling2d_12 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 flatten_3 (Flatten)         (None, 6272)              0

 dense_6 (Dense)             (None, 64)                401472

 activation_19 (Activation)  (None, 64)                0

 dropout_3 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 3)                 195

 activation_20 (Activation)  (None, 3)                 0

=================================================================
Total params: 430,307
Trainable params: 430,307
Non-trainable params: 0
_________________________________________________________________
```

```
tf.random.set_seed(123)
import random
random.seed(123)
# Entrenamos el modelo y realizamos su validación:
modelo.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
from keras import callbacks
earlystopping=callbacks.EarlyStopping(monitor="val_loss",mode="min",patience=5,restore_be
modelo.fit(x_train,y_train,batch_size=128,verbose=1,epochs=25,
```

```
                validation_data=(x_test,y_test),callbacks=earlystopping)
```

```
Epoch 1/25
2/2 [==============================] - 16s 7s/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 4.9418 - val_accuracy: 0.6111
Epoch 2/25
2/2 [==============================] - 16s 7s/step - loss: 0.0120 - accuracy: 0.9952 - val_loss: 4.9258 - val_accuracy: 0.6111
Epoch 3/25
2/2 [==============================] - 15s 7s/step - loss: 7.8073e-04 - accuracy: 1.0000 - val_loss: 4.5549 - val_accuracy: 0.6111
Epoch 4/25
2/2 [==============================] - 15s 7s/step - loss: 0.0248 - accuracy: 0.9952 - val_loss: 4.2752 - val_accuracy: 0.6111
Epoch 5/25
2/2 [==============================] - 15s 7s/step - loss: 0.0328 - accuracy: 0.9903 - val_loss: 4.4306 - val_accuracy: 0.5778
Epoch 6/25
2/2 [==============================] - 15s 7s/step - loss: 0.0248 - accuracy: 0.9903 - val_loss: 4.1030 - val_accuracy: 0.5667
Epoch 7/25
2/2 [==============================] - 15s 7s/step - loss: 0.0353 - accuracy: 0.9758 - val_loss: 3.3243 - val_accuracy: 0.6333
Epoch 8/25
2/2 [==============================] - 15s 7s/step - loss: 0.0807 - accuracy: 0.9662 - val_loss: 3.0211 - val_accuracy: 0.6222
Epoch 9/25
2/2 [==============================] - 15s 7s/step - loss: 0.0145 - accuracy: 1.0000 - val_loss: 3.0252 - val_accuracy: 0.5889
Epoch 10/25
2/2 [==============================] - 15s 7s/step - loss: 0.0407 - accuracy: 0.9807 - val_loss: 3.0368 - val_accuracy: 0.6000
Epoch 11/25
2/2 [==============================] - 15s 7s/step - loss: 0.0656 - accuracy: 0.9758 - val_loss: 2.9130 - val_accuracy: 0.6444
Epoch 12/25
2/2 [==============================] - 15s 7s/step - loss: 0.0681 - accuracy: 0.9614 - val_loss: 2.9582 - val_accuracy: 0.6333
Epoch 13/25
2/2 [==============================] - 15s 7s/step - loss: 0.0478 - accuracy: 0.9855 - val_loss: 3.0262 - val_accuracy: 0.6222
Epoch 14/25
2/2 [==============================] - 15s 7s/step - loss: 0.0534 - accuracy: 0.9807 - val_loss: 3.1554 - val_accuracy: 0.5667
Epoch 15/25
2/2 [==============================] - 15s 7s/step - loss: 0.1032 - accuracy: 0.9614 - val_loss: 2.6864 - val_accuracy: 0.5889
Epoch 16/25
2/2 [==============================] - 15s 7s/step - loss: 0.0586 - accuracy: 0.9758 - val_loss: 2.3039 - val_accuracy: 0.6222
Epoch 17/25
2/2 [==============================] - 15s 7s/step - loss: 0.0621 - accuracy: 0.9758 - val_loss: 2.2890 - val_accuracy: 0.6444
Epoch 18/25
2/2 [==============================] - 15s 7s/step - loss: 0.0741 - accuracy: 0.9565 - val_loss: 2.5800 - val_accuracy: 0.6222
Epoch 19/25
2/2 [==============================] - 15s 7s/step - loss: 0.0251 - accuracy: 0.9903 - val_loss: 3.0853 - val_accuracy: 0.6000
Epoch 20/25
2/2 [==============================] - 15s 7s/step - loss: 0.0541 - accuracy: 0.9855 - val_loss: 3.5234 - val_accuracy: 0.5889
Epoch 21/25
2/2 [==============================] - 15s 7s/step - loss: 0.0399 - accuracy: 0.9855 - val_loss: 3.7491 - val_accuracy: 0.5778
Epoch 22/25
2/2 [==============================] - 15s 7s/step - loss: 0.0246 - accuracy: 0.9952 - val_loss: 3.7901 - val_accuracy: 0.6111
<keras.callbacks.History at 0x7f315870df50>
```

```
# Mostramos un esquema de nuestra red neuronal:
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
SVG(model_to_dot(modelo,show_shapes = True).create(prog='dot', format='svg'))
```


conv2d_input | InputLayer | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
conv2d | Conv2D | input: (None, 256, 256, 3) | output: (None, 254, 254, 32)
activation | Activation | input: (None, 254, 254, 32) | output: (None, 254, 254, 32)
max_pooling2d | MaxPooling2D | input: (None, 254, 254, 32) | output: (None, 127, 127, 32)
conv2d_1 | Conv2D | input: (None, 127, 127, 32) | output: (None, 125, 125, 32)
activation_1 | Activation | input: (None, 125, 125, 32) | output: (None, 125, 125, 32)
max_pooling2d_1 | MaxPooling2D | input: (None, 125, 125, 32) | output: (None, 62, 62, 32)
conv2d_2 | Conv2D | input: (None, 62, 62, 32) | output: (None, 60, 60, 32)
activation_2 | Activation | input: (None, 60, 60, 32) | output: (None, 60, 60, 32)
max_pooling2d_2 | MaxPooling2D | input: (None, 60, 60, 32) | output: (None, 30, 30, 32)
conv2d_3 | Conv2D | input: (None, 30, 30, 32) | output: (None, 28, 28, 32)
activation_3 | Activation | input: (None, 28, 28, 32) | output: (None, 28, 28, 32)
max_pooling2d_3 | MaxPooling2D | input: (None, 28, 28, 32) | output: (None, 14, 14, 32)
flatten | Flatten | input: (None, 14, 14, 32) | output: (None, 6272)
dense | Dense | input: (None, 6272) | output: (None, 64)
activation_4 | Activation | input: (None, 64) | output: (None, 64)
dropout | Dropout | input: (None, 64) | output: (None, 64)
dense_1 | Dense | input: (None, 64) | output: (None, 3)
activation_5 | Activation | input: (None, 3) | output: (None, 3)

```
# Evaluamos el modelo con los datos de prueba:
score = modelo.evaluate(x_test, y_test, verbose=0)

# Mostramos la precisión del modelo:
print('\n', 'Test accuracy:', score[1])
```

```
Test accuracy: 0.6222222447395325
```