UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

TRABAJO DE FIN DE GRADO

USO DE LA AGREGACIÓN BOOTSTRAP Y LOS BOSQUES ALEATORIOS EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Tutor: Miguel Rodríguez Rosa

Autora: Carla Sánchez Jiménez

Grado en Estadística

Curso académico 2023-24

UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

TRABAJO DE FIN DE GRADO

USO DE LA AGREGACIÓN BOOTSTRAP Y LOS BOSQUES ALEATORIOS EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Firmado por:

SANCHEZ JIMENEZ CARLA - 09209305J

Firmado digitalmente por SANCHEZ JIMENEZ CARLA - 09209305J
Fecha: 2024.07.03 10:53:37 +02'00'

RODRIGUEZ ROSA MIGUEL - 70821913Z

Firmado digitalmente por RODRIGUEZ ROSA MIGUEL - 70821913Z
Fecha: 2024.07.03 10:55:54 +02'00'

Autora: Carla Sánchez Jiménez

Tutor: Miguel Rodríguez Rosa

Grado en Estadística

Curso académico 2023-24

Certificado del tutor TFG Grado en Estadística

D. Miguel Rodríguez Rosa, profesor del Departamento de Estadística de la Universidad de Salamanca,

HACE CONSTAR:

Que el trabajo titulado "*Uso de la agregación Bootstrap y los bosques aleatorios en la minería de datos. Creación de una aplicación web con R para clasificar o predecir datos reales*", que se presenta, ha sido realizado por D.ª Carla Sánchez Jiménez, con DNI 09209305J y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Estadística en esta Universidad.

Salamanca, a 3 de julio de 2024.

RODRIGUEZ ROSA MIGUEL - 70821913Z

Firmado digitalmente por RODRIGUEZ ROSA MIGUEL - 70821913Z
Fecha: 2024.07.03 10:56:47 +02'00'

Fdo.: Miguel Rodríguez Rosa

Índice general

Resumen

El trabajo se centra en la implementación y aplicación de los algoritmos bagging y random forest para predecir enfermedades cardiovasculares. El objetivo principal fue desarrollar una aplicación web utilizando RStudio, integrando técnicas estadísticas para ayudar en la toma de decisiones clínicas. El proyecto detalla los pasos de preprocesamiento de datos y elabora el proceso de desarrollo de software utilizando R y Shiny.

Inicialmente, se eligió una base de datos de enfermedades cardiovasculares por su relevancia clínica y calidad de datos. El conjunto de datos se dividió en subconjuntos de entrenamiento, testeo y predicción para facilitar el desarrollo y validación del modelo.

El núcleo del trabajo se centra en la aplicación de los algoritmos bagging y random forest. Bagging, o bootstrap aggregating, implica generar múltiples versiones de un predictor y usar estos para obtener un resultado agregado. Random Forest, una extensión de bagging, construye una multitud de árboles de decisión y combina sus resultados para mejorar la precisión predictiva. Estos métodos se implementaron en R, con su rendimiento evaluado en los datos de enfermedades cardiovasculares.

Los resultados mostraron el potencial de estas técnicas en entornos clínicos. El trabajo también aborda la creación de una aplicación web interactiva utilizando Shiny, que permite a los usuarios cargar datos, especificar divisiones de entrenamiento y testeo, y visualizar los resultados de las predicciones. Esta aplicación sirve como una herramienta práctica para los sanitarios, mejorando su capacidad para diagnosticar condiciones cardiovasculares con precisión.

En conclusión, el trabajo cumple sus objetivos al proporcionar una aplicación web funcional que aprovecha métodos estadísticos avanzados para ayudar en el diagnóstico de enfermedades cardiovasculares. La integración de los algoritmos bagging y random forest en una interfaz fácil de usar ejemplifica la aplicación práctica de la ciencia de datos en la atención médica, ofreciendo un recurso valioso para los profesionales médicos.

Summary

The work focuses on the implementation and application of bagging and random forest algorithms to predict cardiovascular diseases. The main objective was to develop a web app using RStudio, integrating statistical techniques to aid in clinical decision-making. The project details the steps of data preprocessing and elaborates on the software development process using R and Shiny.

Initially, a cardiovascular disease database was chosen for its clinical relevance and data quality. The dataset was divided into training, testing, and prediction subsets to facilitate the model development and validation.

The core of the work centers on the application of the bagging and random forest algorithms. Bagging, or bootstrap aggregating, involves generating multiple versions of a predictor and using these to obtain an aggregated result. Random Forest, an extension of bagging, builds a multitude

of decision trees and combines their results to improve predictive accuracy. These methods were implemented in R, with their performance evaluated on cardiovascular disease data.

The results demonstrated the potential of these techniques in clinical settings. The work also addresses the creation of an interactive web app using Shiny, allowing users to upload data, specify training and testing splits, and visualize prediction results. This app serves as a practical tool for healthcare providers, enhancing their ability to accurately diagnose cardiovascular conditions.

In conclusion, the work meets its objectives by providing a functional web app that leverages advanced statistical methods to aid in the diagnosis of cardiovascular diseases. The integration of bagging and random forest algorithms into an easy-to-use interface exemplifies the practical application of data science in healthcare, offering a valuable resource for medical professionals.

Introducción

Historia de la minería de datos

Introducción a la minería de datos

Los datos son representaciones de variables, los cuales pueden ser cuantitativos o cualitativos, e indican un valor (Cutro, 2010). Estos se pueden representar como números, símbolos o letras. Tienen una gran importancia porque pueden convertirse en información, ya que tienen la posibilidad de asociarse con un contexto. La información permite tomar decisiones más precisas porque disminuye la incertidumbre.

Es necesario que los datos estén en una memoria, pero existen algunos problemas. Para solucionarlos se guardan en un conjunto de archivos, a lo que llamamos base de datos. El descubrimiento de las bases de datos es fundamental para procesar datos y así poder encontrar información valiosa y útil, pero a veces tanta cantidad de información implica imposibilidad de sacar la relevante. Debido al volumen significativo de datos que gestionan ciertas organizaciones, es imposible analizar estos mediante análisis simples, como la regresión, y se necesitan de algunos más avanzados. A finales de los años 80, la estadística amplió sus técnicas, como el desarrollo de las redes neuronales, entre otras.

Para solventar el problema anterior, nació el *Knowledge Discovery from Databases* (KDD) en 1989: este es el proceso del descubrimiento de patrones valiosos, útiles, novedosos y comprensibles llevado a cabo mediante algoritmos. Es un tipo de conocimiento no supervisado. Últimamente, hay gran interés en que la información obtenida sea representada mediante un gráfico y sea visualmente clara sin ningún ruido. Los nueve pasos que sigue el KDD son los siguientes: Conocimiento relevante y entendimiento del objetivo final del usuario, creación de los datos objetivos, preprocesado y limpieza de dichos datos, transformación y reducción de esos datos teniendo en cuenta nuestro objetivo final, elección del tipo de sistema para minería de datos (clasificación, regresión, *clustering*, ...), elección del algoritmo, aplicación de dicho método o técnica, interpretación de los resultados y, por último, documentación de dichas conclusiones a la parte interesada, para buscar una estrategia.

Para poder analizar los datos, es importante que sean coherentes entre ellos. Hay veces que la información que se quiere investigar se encuentra en distintas bases de datos, tanto internas como externas, en ese momento se crea una nueva necesidad: la necesidad de tener que combinar bases de datos y ficheros, de manera que podamos analizarlos y obtener conclusiones. También surgen otros problemas como el lenguaje de cada organización para nombrar un mismo concepto. En este contexto surge el concepto de *Data-Warehousing* (DW): **Almacenes o Bodegas de Datos**. Es un sistema que agrega y combina información de distintas bases de datos. Para que los datos estén centralizados, es importante tenerlos en un único almacén del que puedan hacer uso las empresas y poder mejorar, así, la toma de decisiones, permitiendo análisis como la realización de consultas complejas.

Otro concepto importante es el de **inteligencia de negocios** (Beltrán-Martínez, 2001). Este es un conjunto de tecnologías y procesos que permiten reunir, depurar y transformar datos mediante análisis, generación de informes y visualización de datos, para proporcionar a las organizaciones una visión clara y ayudar en el planteamiento de estrategias mediante la toma de decisiones. Algunos de los beneficios de este sistema incluyen la capacidad de análisis y la reducción de costos, entre otros. Las herramientas empleadas para lograr esto son técnicas de minería de datos. La minería se especializa en descubrir patrones, tendencias y relaciones ocultas en grandes conjuntos de datos.

La minería de datos es la etapa de descubrimiento en el proceso de KDD. Es una disciplina que consiste en el uso de algoritmos para buscar información y generar patrones a partir de datos preprocesados, para así, posteriormente, transformar esa información en una estrategia para cualquier organización y poder competir a nivel mundial con otras organizaciones. Estos algoritmos utilizan la predicción y la descripción. La minería une técnicas semiautomáticas de inteligencia artificial (IA), así como análisis estadístico para lograr obtener información oculta en los datos. Es uno de los procesos tecnológicos de análisis de datos más relevantes a nivel mundial. Está asociado al DW, que proporciona la información necesaria, mientras que los algoritmos de minería extraen dicha información. El explorador puede decidir qué tipos de patrones desea descubrir. La búsqueda de conocimiento se puede dividir en *directed data mining*, donde se sabe lo que se busca, generalmente patrones de datos, y *undirected data mining*, donde no se conoce lo que se busca y se trabaja con los datos, en general sin un objetivo específico.

El proceso de la minería es el siguiente: Se inicia identificando los datos, luego se preparan esos datos filtrándolos según nuestro interés. A continuación, se seleccionan las variables que serán útiles para el posterior análisis y, tras ello, se aplica el algoritmo elegido para obtener un modelo de conocimiento, mediante un criterio previo. Se pueden usar uno o más, pero hay que tener en cuenta el posible cambio de preprocesado. Después, se analizan los resultados y patrones y se comunica lo obtenido, comprobando previamente la validez de esa información. Si se han realizado varios algoritmos se tendrán que comprobar y quedarnos con el que se ajuste mejor a nuestro problema de estudio.

Un aspecto crucial es evitar confundir la minería de datos con el monitoreo de datos. El monitoreo de datos se enfoca en datos que están en constante cambio, mientras que la minería de datos trabaja con datos estáticos.

En cuanto a la evolución de la minería de datos, es esencial destacar que no es un concepto nuevo, sino que tiene sus raíces en los años 50. En esa época, los departamentos de informática realizaban resúmenes de la información almacenada en archivos de computadoras centrales para facilitar la gestión directiva. Este proceso dio lugar a los sistemas de información, aunque inicialmente eran voluminosos y difíciles de manejar. El análisis de datos ha ganado relevancia gracias a tres factores principales: el incremento en la capacidad de procesamiento de los ordenadores, el surgimiento de sistemas de gestión de bases de datos en los años 60 y 70 (aunque eran poco flexibles y se limitaban a cálculos simples), el aumento en la velocidad de adquisición de datos, y el desarrollo de nuevos métodos, incluyendo técnicas de minería de datos para descubrir correlaciones sin necesidad de hipótesis previas en datos con mucho ruido.

En los años 80, figuras como Rakesh Agrawal, Gio Wiederhold, Robert Blum y Gregory Piatetsky-Shapiro, entre otros, comenzaron a consolidar el concepto de Minería de Datos y KDD. La evolución de sus herramientas a lo largo del tiempo se puede dividir en cuatro etapas principales: Recolección de Datos (década de 1960), Acceso a los Datos (década de 1980), Almacén de Datos y Soporte a Decisiones (principios de la década de 1990), y Minería de Datos Inteligente (finales de la década de 1990).

La minería de datos es un proceso que involucra aplicaciones de software. Las etapas de este pro-

ceso son las siguientes: selección y transformación de los datos de entrada, aplicación de técnicas de minería de datos, y finalmente, la interpretación de los resultados obtenidos en la etapa anterior.

¿Qué no es la minería de datos? El *data mining* no se limita a la estadística. Aunque ambos buscan construir modelos, existen diferencias fundamentales entre ellos. La minería de datos combina técnicas estadísticas, ajuste de modelos y aplicaciones de IA para descubrir patrones. Mientras que las técnicas estadísticas son predominantemente confirmatorias, las del *data mining* son exploratorias y resultan especialmente útiles en el análisis de grandes volúmenes de datos.

La minería de datos tiene numerosas aplicaciones en diversos campos, como buscadores inteligentes en internet, marketing empresarial y ciencias de la salud.

Para este estudio, se empleará la minería de datos para predecir enfermedades cardiovasculares.

Las enfermedades cardiovasculares son una de las principales causas de muerte a nivel mundial, junto con las enfermedades cerebrovasculares. Según la *World Heart Federation* (WHF), estas enfermedades provocan aproximadamente 17.9 millones de muertes al año y son una causa frecuente de mortalidad en Estados Unidos.

Estas enfermedades afectan al corazón y a los vasos sanguíneos, incluyendo la obstrucción de estos últimos. La ateroesclerosis ocurre cuando se acumula grasa y colesterol en las paredes arteriales, formando placas que pueden causar eventos graves como ataques cardíacos o accidentes cerebrovasculares.

Reducir la obesidad, controlar la presión arterial, los niveles de colesterol y el hábito de fumar son medidas cruciales para mitigar estos riesgos y potencialmente reducir a la mitad las tasas de enfermedades cardiovasculares.

Los diferentes tipos de enfermedades cardiovasculares incluyen (MedlinePlus. Información de salud para usted, 2024):

- Cardiopatía coronaria (CHD)

  Es la forma más común de enfermedad cardíaca, causada por la acumulación de placa en las arterias, lo que reduce el flujo de sangre y oxígeno al corazón, potencialmente conduciendo a insuficiencia cardíaca o arritmias.

- Insuficiencia cardíaca

  Ocurre cuando el músculo cardíaco se debilita o se vuelve rígido, dificultando que el corazón bombee suficiente sangre al cuerpo. Esto puede afectar a ambos lados del corazón y está asociado con hipertensión y cardiopatía coronaria.

- Arritmias

  Son alteraciones en el ritmo cardíaco, causadas por problemas en el sistema eléctrico del corazón, que pueden manifestarse como latidos irregulares, demasiado rápidos o lentos.

- Enfermedades de las válvulas cardíacas

  Ocurre cuando una de las válvulas del corazón no funciona correctamente, provocando flujo sanguíneo defectuoso o restricción en el flujo sanguíneo normal. Los síntomas pueden incluir soplos cardíacos y sonidos anormales en los latidos del corazón.

- Arteriopatía periférica

  Esta condición se produce cuando las arterias de las piernas y los pies se estrechan o se bloquean debido a la acumulación de placas, lo que impide que la sangre y el oxígeno

lleguen adecuadamente a los tejidos y nervios de estas extremidades.

- Presión arterial alta (hipertensión arterial)

  Es una enfermedad que puede llevar a ataques cardíacos, insuficiencia cardíaca y accidentes cerebrovasculares debido a la presión elevada en las arterias.

- Accidentes cerebrovasculares

  Ocurre cuando hay interrupción del flujo sanguíneo en el cerebro, ya sea por un coágulo en los vasos sanguíneos cerebrales o por un sangrado. Comparte muchas características con los problemas de la enfermedad coronaria.

- Cardiopatía congénita

  Las personas afectadas por esta condición nacen con malformaciones en el corazón, las cuales pueden provocar diversos problemas cardíacos desde el nacimiento, siendo la anomalía congénita la más común.

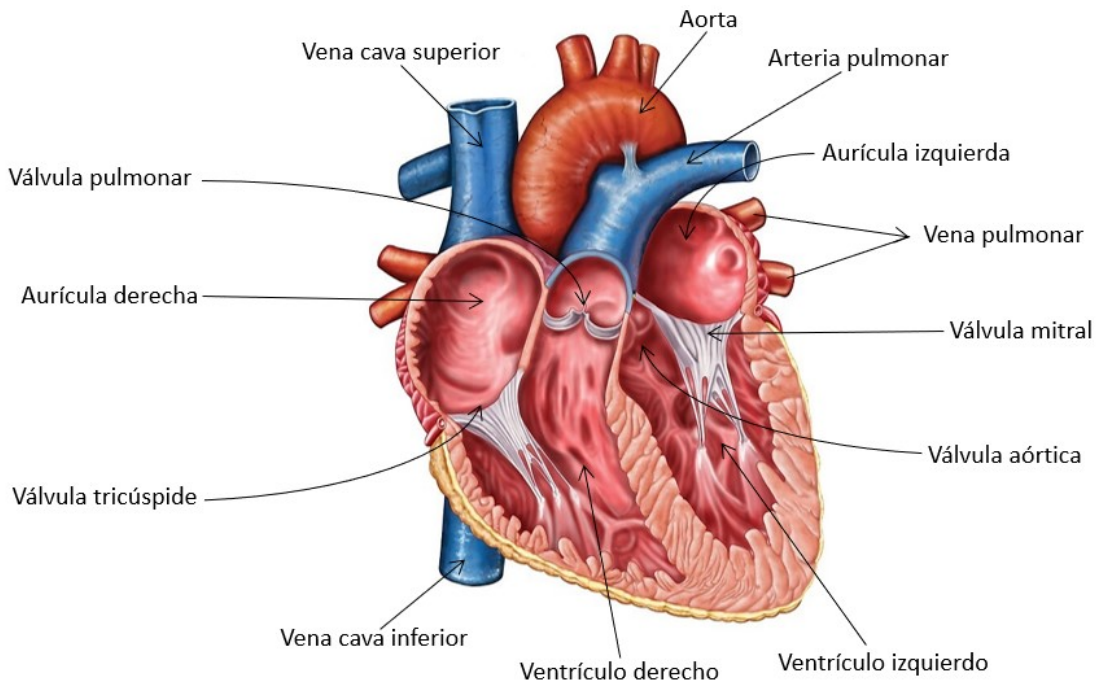

Figura 1: Diagrama de un corazón con sus partes.

Para estos tipos de enfermedades existen distintos tratamientos, a continuación se mencionan y explican brevemente los más importantes y en qué consiste cada uno de ellos (Fundación Española del Corazón (FEC), 2024):

- Sacubitril/Valsartan (SAC/VAL)

  Son dos fármacos, pero con un compuesto único. Están indicados para pacientes con insuficiencia cardíaca sintomática con la cantidad de sangre que sale del corazón (eyección) reducida. Este tratamiento reduce la muerte cardiovascular, así como la hospitalización. También mejora la calidad de vida.

- Rehabilitación cardíaca

  Conjunto de actividades orientadas a los pacientes enfermos del corazón para asegurar una

condición física, mental y social mejorada. Para conseguirlo se les ofrece ejercicio físico y un programa psicológico, entre otras cosas. Hay varias fases, la 1ª se realiza en el hospital; la 2ª, después del alta, también en el hospital, donde están involucrados cardiólogos, enfermeros y otros especialistas; y en la 3ª y última fase, se lleva a cabo una serie de programas de ejercicio para mantener y mejorar lo adquirido en la 2ª fase.

La rehabilitación cardíaca es una de las mejores intervenciones para la reducción de la mortalidad cardíaca.

- Inhibidores PCSK9

  Es un tratamiento para reducir los niveles de colesterol de baja densidad y el riesgo de enfermedades cardiovasculares. La PCSK9 reduce el número de receptores, que son a los que se fijan el colesterol en el hígado, entonces estos fármacos eliminan la PCSK9 para así reducir los niveles de colesterol. Se administra en pacientes con niveles de cLDL >100 mg/dl y en pacientes con cardiopatía isquémica.

- Ezetimibe

  Reduce los niveles de colesterol total y LDL en sangre. Se usa junto a cambios en el estilo de vida. Es mejor la administración junto con estatina. Se administra en pacientes con riesgo cardiovascular que no han conseguido reducir los niveles de colesterol LDL con los cambios de vida, así como en pacientes que presenten insuficiencia renal, diabetes, o que hayan sufrido un infarto o una angina.

- Antianginosos
  - Nitratos: Se utilizan para el dolor de pecho por angina de pecho y para aliviar los síntomas de la insuficiencia cardíaca. Son vasodilatadores que hacen que mejore el flujo de la sangre.
  - Calcios antagonistas: Dilatan las arterias coronarias, periféricas y arteriolas, aparte, también reduce la frecuencia cardíaca. Debe evitarse en pacientes con insuficiencia cardíaca.
  - Ivabradina: Este fármaco reduce la frecuencia cardíaca, para ello inhibe la corriente If (corriente lenta de sodio) para así reducir las demandas de oxígeno, aumentando el flujo sanguíneo coronario. Está indicado, principalmente, para pacientes con tratamiento de la angina de pecho.
  - Ranolazina: Se usa sola o junto con otros medicamentos para tratar la angina crónica. Reduce la acumulación intracelular de sodio. Se administra en pacientes que no respondieron a los betabloqueantes o a antagonistas de calcio.

- Asistencias ventriculares/Oxigenador de membrana extracorpórea (ECMO)

  Es un sistema de asistencia mecánica que facilita soporte circulatorio durante un periodo corto de tiempo. Puede emplearse como puente a la recuperación, al trasplante o a la implantación de asistencia ventricular. Se da en pacientes con insuficiencia cardíaca o respiratoria.

- Cardioversión eléctrica

  Consiste en descargas eléctricas al corazón mediante parches. Se realiza para actuar ante una taquicardia o fibrilación. Se hace mediante sedación y tiene un éxito de 90 %, pero después se necesitará un tratamiento.

- Stent coronario

  Son dispositivos con forma de muelle que se instauran en el vaso sanguíneo concreto para conseguir el paso de sangre y así corregir el ensanchamiento de las arterias. Los tipos de stent son: stent convencional, stent recubierto de fármacos y stent farmacoactivos bioabsorbibles.

- Marcapasos

  Es un dispositivo electrónico cuya función es enviar al corazón impulsos para estimularlo. El marcapasos se instala en pacientes con frecuencia anormal del ritmo cardíaco. Existen dos tipos: marcapasos temporales y permanentes. Una vez instalado el marcapasos en un cuerpo, se debe seguir unas recomendaciones para que el dispositivo haga adecuadamente su función.

- Desfibrilador

  Los desfibriladores emiten corrientes eléctricas para llevar al corazón a su ritmo normal. Los más comunes son los externos. Se hace uso de él, principalmente, cuando la persona entra en parada cardíaca.

  El desfibrilador automático implantable (DAI) son aparatos que se instalan en el cuerpo para detectar de manera rápida una arritmia y así emitir una corriente eléctrica para poder evitar la muerte. Como estos son caros, los más utilizados son los desfibriladores semiautomáticos.

- Prótesis valvulares

  Estas prótesis son de tipo biológicas y mecánicas, y la intervención de corazón consiste en reemplazar la válvula dañada por otra. Estas válvulas se dañan debido a la estenosis y la insuficiencia o cierre valvular. La válvula se sustituye en caso de sufrir una enfermedad desde el nacimiento, lesiones degenerativas, ...

- Balón intraaórtico

  Es un dispositivo que se implanta en caso de haber sufrido un infarto agudo de miocardio, pacientes con insuficiencia mitral aguda secundaria y en pacientes que se van a someter a cirugía cardíaca. Este se sitúa en la arteria aorta torácica.

- Investigación genética

  El objetivo es buscar genes para poder identificar enfermedades hereditarias, así como defectos genéticos. En 1998 se identificaron 5 genes y 70 mutaciones que estaban relacionadas con la miocardiopatía hipertrófica.

- Ablación por radiofrecuencia

  Es una técnica que se realiza para controlar el ritmo cardíaco. Está indicada para pacientes con taquiarritmias y para pacientes con algunas arritmias como el síndrome de Wolf-Parkinson-White.

- Ventilación mecánica

  Es una técnica que consiste en darle al paciente una asistencia para poder llevar a cabo la respiración. Hay dos tipos de ventilación, la mecánica invasiva y la no invasiva.

- Hemofiltración

  Las TRR son un conjunto de procedimientos, que se lleva a cabo en pacientes con la función renal afectada, muchos de ellos tienen ese trastorno una vez que se les diagnostica de insuficiencia cardíaca. La técnica consiste en el reemplazo de dicha función renal. Después de esto, se debe administrar un tratamiento. Hay dos tipos de TRR, las continuas y las intermitentes.

- Anticoagulación

  Los coágulos en general son un mecanismo muy importante para el sangrado, pero hay veces que la formación de coágulos en la sangre puede ser un problema. Para ello, se deben administrar anticoagulantes para solventar ese problema. Los más utilizados son los siguientes: heparinas, los anticoagulantes orales (sintrom) y anticoagulantes de acción directa. Se

emplea en pacientes con valvulopatías o con síndrome coronario agudo.

- Antiagregantes

  Las plaquetas evitan hemorragias, pero esta acción puede tener consecuencias hasta llegar a un infarto. Los antiagregantes evitan que las plaquetas lleven a cabo el procedimiento de formación de coágulos. Los fármacos más utilizados son las aspirinas.

- Cirugía de revascularización coronaria

  Esta cirugía consiste en suministrar sangre a las zonas con falta de riego. Está indicada para pacientes con enfermedad coronaria.

- Estatinas

  Es un fármaco que se administra a pacientes con niveles de colesterol altos, ya que estas cifras pueden llevar a varias enfermedades cardiovasculares. También se administra en pacientes con patología cardiovascular.

- Trasplante

  El trasplante consiste en el reemplazo de un corazón enfermo por uno de una persona fallecida. Esto puede tener alguna consecuencia, aunque la esperanza de supervivencia es de un 70 % a los 5 años del trasplante. Se realiza en pacientes con insuficiencia cardíaca terminal.

- Betabloqueantes

  Son fármacos utilizados para diversas patologías. Los betabloqueantes, impiden la unión de los beta-adrenérgicos con la catecolamina, porque esta unión provoca una elevada frecuencia cardíaca, presión arterial y contractilidad cardíaca.

  Estos se utilizan en caso de una angina de pecho, arritmias, hipertensión arterial, entre otras, y en enfermedades no cardíacas como, por ejemplo, en ansiedad y migraña.

- IECA y ARA II

  Son medicamentos empleados para tratar la hipertensión arterial, ya que ayudan a regular la presión sanguínea y reducen el riesgo de eventos cardiovasculares. Se prescriben a pacientes con hipertensión arterial, así como con condiciones adicionales como insuficiencia cardíaca, enfermedad coronaria y diabetes mellitus.

Después de una explicación detallada sobre las enfermedades cardiovasculares y su importancia, se justificará la elección de la base de datos.

El estudio y la predicción de enfermedades cardiovasculares son importantes en la investigación, así como, fundamentales en la atención médica.

Las enfermedades cardíacas a veces pueden progresar de manera silenciosa, sin síntomas evidentes en las etapas iniciales. La predicción ayuda a detectar factores de riesgo y signos tempranos de esta enfermedad antes de que se manifiesten. Esto ayuda a la toma de medidas preventivas para reducir los riesgos y las complicaciones más graves.

Actuar a tiempo hace que mejore la calidad de vida del paciente con la ayuda del cambio de estilo de vida, de intervenciones quirúrgicas, junto con tratamientos, y prevé discapacidades relacionadas con el corazón. También ayuda a reducir los costos relacionados con los tratamientos de enfermedades cardiovasculares en etapas avanzadas.

Aparte de todo esto, es importante el tema de la medicina personalizada a la que se pretende llegar hoy en día, identificando el caso de cada paciente antes de que ocurra un episodio, para así poder actuar con más precisión y de manera particular.

En resumen, el estudio de la predicción de las enfermedades cardiovasculares es fundamental para una atención médica preventiva y eficaz. El descubrimiento temprano y la actuación pueden marcar la diferencia en la salud y la vida de las personas, al igual que en los costos asociados con el tratamiento de enfermedades cardíacas.

Ahora una vez definidos los términos principales de este trabajo se verá su estructura:

Tras un primer apartado con los objetivos del trabajo, el capítulo inicial tratará del desarrollo estadístico de los árboles de decisión, el bagging y el random forest.

En el segundo capítulo se hablará de la base de datos elegida para el trabajo.

El siguiente capítulo se centrará en el software utilizado para el trabajo, así como el código de la aplicación web.

El capítulo final tratará de la utilización de la aplicación web creada a la base de datos.

Por último, se ofrecerá la bibliografía utilizada para la realización de este trabajo, en esta se encontrarán diversos libros y páginas web.

Objetivos

El objetivo principal de este trabajo es crear una aplicación web con R para el uso de las técnicas de bagging y random forest para la predicción de una variable respuesta.

Por otra parte, los objetivos secundarios son:

- Definir y demostrar matemáticamente la técnica de bootstrap (bagging).
- Definir y demostrar matemáticamente la técnica de random forest.
- Realizar una búsqueda para elegir una base de datos adecuada.
- Aplicar los algoritmos de R a la base de datos encontrada de enfermedades cardiovasculares para así poder ayudar a los médicos en el diagnóstico.

Capítulo 1

Desarrollo estadístico

1.1. Inducción del árbol de decisión

Los árboles de decisión son algoritmos de aprendizaje supervisados. La inducción de estos se da a partir de tuplas de entrenamiento clasificadas con clases (IBM, 2024b). Estos algortimos son utilizados tanto para **clasificación** (Árboles de clasificación) como para **regresión** (CART), cuya estructura es jerárquica (Ye, 2013). Los nodos del árbol realizan en cada iteracción una evaluación para formar así los subconjuntos homogéneos, los cuales suelen representarse con rectángulos. Las ramas describen los resultados de cada prueba. Cada nodo hoja (nodo terminal) contiene una etiqueta de clase, y se representan con óvalos. El nodo raíz es el nodo superior del árbol. Algunos algoritmos generan árboles binarios mientras que otros no.

Para clasificar una tupla usando un árbol de decisión, se comparan los valores de sus atributos con las condiciones en el árbol (Jiawei et al., 2006). Siguiendo las decisiones en el árbol, se traza un camino desde la raíz hasta llegar a una hoja, que contiene la predicción de clase para esa tupla. Los árboles de decisión también pueden ser transformados de manera sencilla en reglas de clasificación, lo que facilita su interpretación y aplicación en diferentes contextos.

El uso de árboles de decisiones es útil para análisis exploratorios, ya que son fáciles de construir así como de interpretar. El uso de estos suele tener buena precisión pero depende un poco de los datos con los que se trabaje y de los disponibles. Estos se utilizan bastante en áreas como la medicina, la astronomía y la biología molecular.

Para la construcción de los árboles se usan medidas de selección para elegir el atributo que mejor divida a las tuplas en cada división. Si el árbol es muy pequeño se puede producir un subajuste, pero, sin embargo, si el árbol es muy grande se puede producir un sobreajuste. Para evitar este problema una de las soluciones es usar la poda, que consiste en eliminar las ramas con menos importancia. La poda también es usada para los ruidos y los valores atípicos que pueden mostrarse en las ramas del árbol

1.1.1. Historia de los algoritmos

Los primeros algoritmos para la creación de árboles de decisión fueron el ID3 (*Iterativo Dichotomiser*) y CART. ID3 fue creado por el investigador en aprendizaje automático J. Ross Quinlan, a finales de los 70 principios de los 80. CART fue inventado por separado pero al mismo tiempo. Estos dos siguen un planteamiento muy parecido basado en el uso de tuplas de entrenamiento. El sucesor de ID3 fue C4.5, y se convirtió en un punto de referencia con el que muchas veces se com-

paran los algortimos de aprendizaje más nuevos. El grupo de estadísticos Breiman et al. (1984), publicaron un libro llamado "Classification and Regression Trees (CART)" en el que se explicaba la formación de árboles de decisión binarios.

Los 3 algoritmos mencionados anteriormente (ID3, C4.5 y CART) adoptan un enfoque codicioso, los árboles se contruyen de arriba a abajo empezando por un conjunto de entrenamiento que se va dividiendo en subconjuntos más pequeños a medida que se va avanzando en la contrucción del árbol:

(1) Crear un nodo N.

(2) Si todas las tuplas en el conjunto de datos D pertenecen a la misma clase C, entonces etiquetar N como un nodo hoja con la clase C.

(3) Si la lista de atributos está vacía, etiquetar N como un nodo hoja con la clase mayoritaria en D (votación por mayoría).

(4) Aplicar el método de entropía para seleccionar el "mejor" atributo de división. Véase la ecuación 1.1.

(5) Etiquetar el nodo N con el atributo seleccionado para la división.

(6) Si el atributo de división es discreto, permitir divisiones multidireccionales y eliminar dicho atributo de la lista de atributos.

lista de atributos ← lista de atributos - atributo de división

(7) Para cada posible resultado j del criterio de división:

- Crear un conjunto D_j de tuplas de datos en D que satisfagan el resultado j.
- Si D_j está vacío, adjuntar un nodo hoja etiquetado con la clase mayoritaria en D al nodo N.
- De lo contrario, adjuntar el árbol de decisión generado recursivamente para D_j y la lista de atributos restantes al nodo N.

(8) Fin

(9) Devolver N.

Ahora se pasará a explicar más detalladamente cada uno de los pasos del algoritmo.

- El algoritmo comienza con 3 parámetros: D que representan una partición de los datos junto con sus etiquetas, una lista de atributos de parámetros que describen las características de los datos y el método de selección para elegir el "mejor" atributo que discrimina las tuplas, que en el estudio se utilizará el método de selección por entropía. El método de entropía permite divisiones de múltiples vías, lo que significa que un nodo puede tener dos o más ramas.
- El algoritmo empieza con un nodo N que representa todas las tuplas de entrenamiento en D (paso 1).
- Si todas las tuplas en D pertenecen a la misma clase, entonces el nodo N será un nodo hoja y por lo tanto se le asigna con esa clase (paso 2). Cabe destacar que en el paso 3 se muestra la condición final del algoritmo que se detalla al final del procedimiento.
- De lo contrario, el algortimo llama al método de selección de atributos para así establecer el método de división. Este criterio nos indica qué atributo evaluar en el nodo N para determinar la forma óptima de separar las tuplas en D en clases individuales (paso 4). El criterio de división también informa sobre qué ramificaciones crecen desde el nodo N teniendo en

cuenta los resultados de la función de entropía. Específicamente, el criterio de división identifica el atributo de división y, en ocasiones, puede especificar un punto o un subconjunto para la división. También se intenta que las ramas sean lo más "puras" posibles, esto quiere decir que todas las tuplas que contiene pertenecen a la misma clase. En otras palabras, al dividir las tuplas en D según los resultados del criterio de división, se espera obtener particiones lo más puras posible.

- Se etiqueta el nodo N con el criterio de división que actúa como una condición en el nodo (paso 5). Se crea una rama desde el nodo N para cada resultado del criterio de división. Las tuplas en D se dividen en consecuencia (paso 7). Hay tres escenarios posibles. Sea A el atributo de divisón, este tiene v valores distintos $\{a_1, a_2, \ldots, a_v\}$, según los datos de entrenamiento.

  1. A tiene un valor discreto: Los resultados del nodo N son valores conocidos de A, por lo que se crea una rama para cada unos de estos valores, a_j, y se pone la etiqueta de ese valor. La partición D_j es el subconjunto de tuplas etiquetadas con clases en D que tienen el valor a_j de A. No es necesario considerar A en ninguna partición futura debido a que todas las tuplas en la partición tienen el mismo valor para A.
  2. A tiene un valor continuo: En este caso, el nodo N tiene dos posibles resultados, $A \leq$ punto de división y $A >$ punto de división, donde ese punto de división se determina por el método de selección de atributos como parte del criterio de división. Este valor no siempre es preexistente. Del nodo N salen dos ramas y se etiquetan de acuerdo a los resultados anteriores. Las tuplas se dividen en dos, D_1 contiene al subconjunto de tuplas que cumplen el criterio $A \leq$ punto de división y D_2 las que cumplen el otro criterio, $A >$ punto de división.
  3. A tiene un valor discreto y se debe producir un árbol binario: En este caso, la prueba en el nodo D tiene la forma "¿$A \in S_A$?", donde S_A es el subconjunto de división para A que devuelve el método de selección de atributos como parte del criterio de división.

     De este nodo N salen dos posibles ramas de la derecha etiquetada como Sí y la de la izquierda etiquetada como No. La primera rama, la de la derecha, corresponde al subconjunto de tuplas etiquetadas D_1, que satisfacen la prueba, y en la otra rama el subconjunto de tuplas etiquetadas en D_2, que no satisfacen la prueba.

El algoritmo utiliza el mismo proceso de forma recursiva para formar el árbol. Las particiones recursivas sólo se detienen si cumplen alguna de estas 3 condiciones de terminación;

1. Si todas las tuplas de partición en D pertenecen a la misma clase
2. Cuando se llega al punto en un árbol de decisión en el que ya no quedan más atributos para dividir las tuplas, se usa un método llamado votación por mayoría. Esto significa que el nodo se convierte en nodo hoja y se etiqueta con la clase más común que aparece en las tuplas del nodo. Otra opción es almacenar la distribución de clases de las tuplas en el nodo para futuras referencias.
3. No hay tuplas para una rama determinada, una partición D_j está vacía. Si ocurre esto se crea un nodo hoja con la clase mayoritaria en D.

Por último, se muestra el árbol de decisión resultante.

La complejidad computacional del algoritmo dado un conjunto de entrenamiento D viene dada por la siguiente expresión $O\left(n \cdot |D| \cdot \log\left(|D|\right)\right)$, donde n es el número de atributos que describen las tuplas en D y $|D|$ es el número de tuplas de entrenamiento en D.

También hay versiones incrementales de inducción de árboles de decisión, cuya función es que cuando se le den nuevos datos de entrenamiento, no realicen otro árbol de decisión sino que rees-

tructuren el que ya se había adquirido a partir del aprendizaje de datos de entrenamientos anteriores.

Las diferencias entre los distintos algoritmos de aprendizaje vienen dadas por cómo se seleccionan los atributos al crear el árbol, así como los procesos utilizados para la poda. El algoritmo descrito anteriormente puede provocar tiempos de espera largos y falta de memoria si se trabaja con grandes bases de datos, debido a que, para cada nivel del árbol, tiene en cuenta todas las tuplas de entrenamiento en D.

1.1.2. Medidas de selección de atributos

Ganancia de información

El algoritmo de aprendizaje ID3 utiliza como medida de selección de atributos la ganancia de información. Esta medida se basa en el trabajo pionero de Claude Shannon sobre teoría de la información, que estudió el valor o "contenido de la información" de los mensajes. El atributo con mayor ganancia de información será el elegido como atributo de división para el nodo N. El atributo elegido clasifica las tuplas en particiones resultantes minimizando la información, por lo tanto, muestra una menor aleatoriedad e impureza en el árbol. Con este criterio de división se reducen las pruebas que se necesitan para clasificar a las tuplas y disminuye también el tamaño del árbol resultante.

La información esperada necesaria para clasificar una tupla en D viene dada por

$$Info\,(D) = -\sum_{i=1}^{m} p_i \cdot \log_2\,(p_i)\,, \tag{1.1}$$

donde p_i es la probabilidad, distinta de cero, de que un tupla en D pertenezca a la clase C_i, y se estima mediante $\frac{|C_{i,D}|}{|D|}$. Como la información está codificada en bits, se usa el logaritmo en base 2. Por lo tanto, $Info\,(D)$ es la cantidad de información necesaria para determinar la etiqueta de clase de una tupla en D, también llamada entropía de D.

Vamos a suponer que se dividen las tuplas en D según un atributo A que tiene v valores distintos, $\{a_1, a_2, \ldots, a_v\}$. Si A tiene un valor discreto, este corresponderá a los v resultados de la prueba en A. Entonces el atributo A se utiliza para dividir D en v subconjuntos $\{D_1, D_2, \ldots, D_v\}$, donde D_j contiene todas las tuplas en D que tienen como resultado a_j de A. El nodo N se dividirá en tantas ramas como particiones se tengan. A la hora de crear el árbol de decisión lo ideal sería que estas particiones fueran puras, es decir, que se produjera una clasificación exacta de las tuplas. No obstante, es poco probable que eso ocurra, estas suelen ser impuras.

Para saber cuánta información más se necesitaría para conseguir una clasificación exacta se utiliza la siguiente fórmula:

$$Info_A\,(D) = \sum_{j=1}^{v} \frac{\left|D_j\right|}{|D|} \cdot Info\left(D_j\right),$$

donde $\frac{|D_j|}{|D|}$ se interpreta como el peso de la j-ésima partición. $Info_A(D)$ es la información que se requiere para clasificar una tupla de D según la partición A. La pureza será mayor cuanto mayor sea la información a la que se refiere.

La ganancia de información representa cuánto mejora nuestra comprensión de los datos al dividirlos en función de una característica específica, y se puede obtener como la diferencia entre la

información requerida inicialmente y la información necesaria después de la división en A. Esta ganacia se calcula por la sigueinte fórmula:

$$Gain\,(A) = Info\,(D) - Info_A\,(D)$$

El atributo de A que tenga la mejor ganacia de información, se eligirá como partición para el nodo N, es decir, que la información requerida para terminar de clasificar las tuplas sea mínima.

Sin embargo, si el atributo A tiene valores continuos, lo que se debe decidir es el mejor umbral en A. En primer lugar, se deberán ordenar los valores de A en orden creciente. El punto medio entre cada par de valores puede ser un posible punto de división. Por eso, si se tiene v valores de A, evaluaremos $v - 1$ posibles divisiones. El punto medio entre dos valores quedaría de la siguiente manera:

$$\frac{a_i + a_{i+1}}{2}$$

Para cada punto posible de división se halla $Info(A)$ donde en la ecuación $v = 2$. El punto con el menor requisito de información esperado es el punto que se elegirá como punto de división para A. D_1 es el conjunto de tuplas en D que satisfacen $A \leq$ punto de división y D_2 el conjunto de tuplas que satisfacen $A >$ punto de división.

1.1.3. Poda de árboles

A la hora de construir un árbol de decisión, en algunas ramas se reflejan algunas singularidades en los datos de entrenamiento debido al ruido o valores atípicos. Como se dijo anteriormente, para evitar el sobreajuste se utiliza la poda. Estos métodos de poda consisten en hacer uso de herramientas estadísticas para eliminar esas ramas. Los árboles podados tienden a ser menos complejos y más pequeños y, por lo tanto, más comprensibles y más fácil de clasificar datos de pruebas independientes.

Hay dos enfoques para la poda de árboles: pre-poda y post-poda.

En el enfoque de pre-poda, el árbol se poda deteniendo su construcción al decidir no dividir más el subconjunto de tuplas de entrenamiento por lo que ese nodo se convierte en nodo hoja e incluye la clase más frecuente entre los subconjutnos de tuplas o la distribución de probabilidad de esas tuplas.

Al construir un árbol se pueden utilizar medidas como la entropía y el índice de Gini entre otras, para evaluar la bondad de una partición. Si la división está por debajo de un umbral preespecificado, entonces se detendrá la divisón del subconjunto. A la hora de elegir el umbral hay que tener cuidado ya que, un umbral elevado puede provocar árboles muy simplificados y umbrales bajos, árboles muy pocos simplificados.

El enfoque más común es el de la post-poda de un árbol desarrollado, que elimina subárboles. Para ello, se poda eliminando las ramas y reemplazándolas por un nodo hoja que esté etiquetado con la clase más común del subárbol. Un ejemplo de post-poda puede ser el algoritmo de poda que usa CART. Este método postula que la complejidad del costo de un árbol está determinada por dos factores: la cantidad de hojas que tiene el árbol y la tasa de error del mismo, donde esta última se refiere al porcentaje de tuplas que el árbol clasifica incorrectamente. Para cada N se halla la complejidad del subárbol en N y la complejidad de costos si este fuera podado. Si al podarlo da menor la complejidad de costos, entonces el subárbol se poda; si no, se conserva.

Se emplea un grupo de tuplas etiquetadas con clases para podar de manera gradual los árboles de decisión, los cuales son independientes tanto del conjunto de entrenamiento usado para construir el árbol original, como de cualquier conjunto de prueba usado para evaluar la precisión. El objetivo del algoritmo es generar una serie de árboles podados de forma progresiva. En términos generales, se busca seleccionar el árbol de decisión más compacto que minimice los costos asociados a su complejidad.

El algoritmo C4.5 utiliza el método de poda pesimista que, al igual que el método de complejidad de costos, emplea estimaciones de la tasa de error para tomar decisiones respecto a la poda. La diferencia con la poda de complejidad de costos es que la poda pesimista no necesita el uso del juego de podas, lo que sí usa es el conjunto de entrenamiento para calcular las tasas de error. La estimación del error en un conjunto de entrenamiento es una medida altamente sesgada por ser optimista, por lo que el método añade una penalización para contrarrestar el sesgo.

Para lograr un enfoque combinado se pueden combinar las dos podas, la pre-poda y la post-poda, la pre-poda requiere de menos cálculos pero el árbol resultante es menos fiable.

Los árboles de decisión pueden volverse difíciles de interpretar debido a la repetición y la replicación. La repetición sucede cuando un mismo atributo se evalúa varias veces a lo largo de una misma rama del árbol. Es decir, se repite la misma pregunta sobre un atributo en diferentes niveles del árbol, lo que puede hacer que la interpretación sea más complicada y llevar a resultados menos claros. Esta situación impide la comprensibilidad y la precisión de los árboles de decisiones. Usar divisiones multivariadas en árboles de decisión implica basarse en combinaciones de atributos en lugar de solo uno, lo que evita problemas de repetición y replicación, al considerar múltiples aspectos en cada decisión del árbol. También se pueden usar reglas en vez de árboles de decisiones, construyendo un clasificador basado en estas reglas SI-ENTONCES a partir de ellos.

Las ventajas de los árboles de decisión son las siguientes: Son **fáciles de interpretar**, casi nunca se necesita preparar los datos y son más **flexibles** que otros algoritmos, porque, como se mencionó antes, se usan tanto para clasificación como para regresión.

Las desventajas de los árboles de decisión son las siguientes: Están **predispuestos** a que se produzca un **sobreajuste**, pequeños cambios en los datos puede producir un árbol muy diferente (**estimadores de alta varianza**) y son **más costosos** que otros algoritmos.

1.2. Bagging

El bagging es un método de aprendizaje supervisado (IBM, 2024a). Es utilizado para aumentar la precisión y para reducir la varianza en un árbol de decisión (James et al., 2017). Los 3 pasos del bagging son: El bootstrapping, que selecciona un conjunto de entrenamiento y toma muestras de datos de ese conjunto con reemplazamiento (los subconjuntos); tras este paso, el entrenamiento paralelo, que entrena estas muestras de manera independiente; y por último, la agregación, que toma el promedio de estas muestras para conseguir la estimación más precisa; esta decisión tiene que tener en cuenta si la tarea va a ser la clasificación o la regresión.

1.2.1. Explicación del bagging

Dado un conjunto de datos representado por D que consiste en d tuplas con características y etiquetas, para mejorar la predicción lo que se hace es aplicar la técnica del bagging, que consiste en la agregación bootstrap.

Con el bagging, se toma el conjunto de muestras original, D. En cada iteracción, i ($i = 1, 2, \ldots, k$)

se selecciona una muestra de entremaniento al azar, D_i, utilizando el muestreo con reemplazo. Debido a que el muestreo es con reemplazo, algunas de las tuplas de D no estarán incluídas en ningún D_i, o por el contrario algunas podrán ocurrir en más de un D_i.

Para cada conjunto de entrenamiento se entrena un modelo de clasificación, M_i. Se obtienen k modelo diferentes.

Para representar una nueva tupla de datos, X, se le pide a cada modelo, M_i que haga una predicción. Cada modelo tiene su voto y se registra la predicción de clase para X.

El clasificador bagged, M^*, toma la decisión final basándose en la mayoría de votos de los modelos individuales. Si hay mayoría de votos por una clase específica, M^* asigna esa clase como la predicción para X.

El bagging también se utiliza para la predicción de valores continuos, donde M^* calcula el promedio de las predicciones de cada modelo individual.

La precisión del clasificador bagged normalmente es significativamente mejor que el clasificador de los datos de entrenamiento originales, D. Este enfoque es más robusto a los efectos de los datos ruidosos y al sobreajuste. Se verá mejorada la precisión cuando el clasificador bagged reduzca la variabilidad de los clasificadores individuales. Esta variabilidad se reduce al construir árboles completos, sin podar y promediarlos.

Es un método por conjuntos, lo que hace que evite el sobreajuste y el subajuste de los árboles de decisión y así poder generalizar los resultados al conjunto de datos.

Se pueden utilizar las observaciones que no se han utilizado para el entrenamiento (“out-of-bag (OOB)”) para poder validar el algoritmo (Dorado-Díaz, 2023). Con este método no se hace uso de todas las observaciones en todas las muestras para el bootstrap.

Las ventajas del bagging son las siguientes: Es **fácil de interpretar** al igual que los árboles de decisión, **reduce** el principal problema de los árboles de decisión, el de **exceso de variabilidad** y proporciona información sobre la **importancia de las variables** al momento de construir el modelo.

Los desafíos/desventajas del bagging son los siguientes: Tiene un **coste elevado** y una**pérdida de interpretabilidad** debido a que las predicciones se basan en el promedio, aunque la salida sea más precisa que cualquier punto de datos individual, un conjunto de datos más completo o preciso también podría mejorar la precisión en un solo modelo de clasificación o regresión, a costa de **menos flexibilidad**..

1.3. Bosques aleatorios

Los bosques aleatorios (random forest) son algoritmos de aprendizaje supervisados (IBM, 2024c), reconocidos por Leo Breiman y Adele Cutler. Al igual que los árboles de decisión y el bagging, se utiliza tanto para clasificación como para regresión. El random forest es una ampliación del bagging, utiliza esta técnica y, además, la aleatoriedad de características para así buscar un subconjunto que reduzca la correlación entre los árboles de decisión. Tiene en cuenta toda la variabilidad de los datos para así reducir el sobreajuste, el sesgo y la varianza.

En el random forest cada uno de los clasificadores en el conjunto es un clasificador de árbol de decisión, de manera que el “bosque” es la agrupación de todos los clasificadores. Combina la salida de varios árboles de decisión para encontrar el resultado único. Cada árbol de decisión por individual se crea utilizando la selección aleatoria de atributos en cada uno de los nodos, y después

se decide cuál es el mejor atributo de división, mediante el método de la entropía, el índice de Gini, u otro. Todos los árboles siguen la misma distribución y dependen todos del vector aleatorio muestreado de forma independiente. A cada árbol le pertenece un voto y con el conjunto de todos ellos se tomará la decisión de la clase más común.

Se tiene un conjunto de entrenamiento D de d tuplas. Para cada iteracción $i\,(i = 1, 2, \ldots, k)$ se toma una muestra de entrenamiento llamada D_i, que consiste en d tuplas seleccionadas al azar del conjunto de datos original D, permitiendo repeticiones de un D_i para otro, y excluyendo algunas tuplas en cada D_i. Luego, en cada nodo del árbol de decisión, se elige al azar un pequeño número de atributos, denotado como F, para considerarlos como opciones de división en ese nodo. Se utiliza la metodología CART para construir los árboles, donde crecen hasta alcanzar su tamaño máximo sin poda. Este enfoque de construcción de bosques aleatorios que incorpora selección aleatoria de atributos se conoce como Forest-RI.

Hay otra forma de construcción de bosques aleatorios que se llama Forest-RC, este enfoque crea nuevas características mediante combinaciones lineales aleatorias de los atributos originales. Fundamentalmente, cada nueva característica se genera sumando aleatoriamente L atributos originales, multiplicados por coeficientes aleatorios. Estos coeficientes se eligen al azar entre $[-1, 1]$. Se repite este proceso para crear F combinaciones lineales en cada nodo. Posteriormente, se busca la mejor división basada en estas nuevas características generadas. Este método es especialmente útil cuando se dispone de pocos atributos originales y se desea reducir la correlación entre los clasificadores individuales del bosque.

Los bosques aleatorios son tan precisos como otras alternativas, pero son más robustos a errores y valores atípicos. Además, su capacidad para generalizar mejora con el tamaño del bosque, es decir, con el número de árboles que lo componen. Esto quiere decir que el sobreajuste no es un problema. La precisión de un bosque aleatorio depende de la calidad de los árboles individuales y de cómo están relacionados entre sí. Es importante mantener la calidad de cada árbol sin aumentar su correlación con los demás. Los bosques aleatorios no se ven afectados significativamente por la cantidad de atributos considerados en cada división, que habitualmente se eligen hasta $\log_2 +1$.

Una observación empírica interesante es que, a veces, usar un solo atributo de entrada aleatorio puede producir una precisión alta, incluso mayor que cuando se utilizan múltiples atributos. Debido a que los bosques aleatorios solo consideran un pequeño número de atributos para cada división, son eficientes en bases de datos muy grandes. En comparación con técnicas como bagging, los bosques aleatorios pueden ser más rápidos. También proporcionan estimaciones internas de qué atributos influyen más en la predicción del modelo.

La diferencia con el bagging es que hay que determinar tres hiperparámetros antes de entrenar los datos. Estos son: el tamaño de los nodos, el número de árboles y el número de características muestreadas.

Las ventajas del random forest son las siguientes: **Reduce el sobreajuste**, el promedio de los árboles no correlacionados **reduce la varianza** y el error, **facilita flexibilidad**, debido al uso de las tareas de clasificación, además de la aleatoriedad de características, y proporciona la **importancia de las variables** al modelo, las cuales son fácil de identificar y con ello hacer un mejor ajuste.

Los desafíos/desventajas del random forest son los siguientes: Es un **proceso lento**, posee mayor **complejidad** y **usa más recursos**.

Capítulo 2

Datos reales

La base de datos que se va a utilizar para el trabajo se encontró en la plataforma de Kaggle, y esta contiene parámetros para la predicción de enfermedades cardiovasculares (Hashimzade, 2020).

En la base de datos inicial se contaba con variables como *id*, *age in days*, *age in years*, *gender*, *height*, *weight*, *ap_hi*, *ap_lo*, *cholesterol*, *gluc*, *smoke*, *alco*, *active* y *cardio*, que pasan a explicarse ahora:

- El *id* se refiere al número de identificación del paciente (va en orden, del 1 al 68399). Es decir, contamos con 68399 pacientes.
- La variable *age in years* informa de la edad de los pacientes, en años.
- La variable *gender*, indica el género de los pacientes y está codificada con los valores 1 y 2, según sea mujer u hombre, respectivamente.
- La variable *height* se refiere a la altura de los pacientes, el máximo es 190 cm y el mínimo 140 cm.
- La variable *weight* se refiere al peso de los pacientes, el máximo es 135 kg y el mínimo 40 kg.
- La variable *ap_hi* informa sobre la presión arterial sistólica. Esta se mide cuando los ventrículos del corazón se contraen. Como valores medios se sitúan a los pacientes cuyos valores sean próximos a 120 milímetros de mercurio.
- La variable *ap_lo*, de forma similar a la variable explicada anteriormente, informa sobre la presión arterial diastólica. Esta se mide cuando los ventrículos del corazón se relajan. Como valores medios se sitúan a los pacientes cuyos valores sean próximos a 80 milímetros de mercurio.

  Por lo tanto, los valores entre los que se tiene que encontrar una persona deben estar entre 80 y 120. Si estos valores son menores o mayores a este rango se dirá que el paciente tiene la presión baja o alta, respectivamente.
- La variable *cholesterol* mide el nivel de colesterol del paciente, codificándose esta en 1, 2 o 3 dependiendo del nivel y si está muy por encima de lo normal, un poco por encima de lo normal o normal.
- Con la variable *gluc* pasa lo mismo, mide el nivel de glucosa del paciente, y esta está codificada con los valores 1, 2 o 3 dependiendo de si está muy por encima de lo normal, un poco por encima de lo normal o normal (70-100 mg/dl).
- La variable *smoke* indica si el paciente fuma o no fuma y se codifica en 0 si no fuma y 1 si fuma.

- Igual que la variable *smoke*, la de *alco* indica si el paciente toma alcohol o no toma, está codificado como 0 si no toma alcohol y 1 si lo toma.
- La variable *active* se refiere a si el paciente hace deporte y es activo, o si, por el contrario, no hace deporte y por lo tanto no es activo. Esta variable, como las dos anteriores, está codificada como 0 si no hace deporte y 1 si, por el contrario, sí que lo hace.
- Y por último, la variable que tiene como nombre *cardio*, es la dependiente y la más importante, puesto que indica si el paciente padece enfermedades cardiovasculares. Si es así se indicará con un 1 y si no las padece con un 0.

Para el estudio, la variable *age in days* se eliminó, ya que contiene la misma información que la variable de *age in year*, pero en otra unidad de medida. En las variables *height* y *weight* se hicieron las modificaciones oportunas para quitar los outliers. Estas modificaciones fueron quedarse con las alturas (*height*) de 140cm a 190cm y con el peso (weight) de 45kg a 135kg, y la decisión de estas modificaciones fueron tomadas al hacer un gráfico y ver valores que no cuadraban (personas que a la vez miden 120cm y pesan 140kg). Otras de las variables que se modificaron fueron las variables de presión arterial sistólica y la presión arterial diastólica; en estas medidas, al igual que en la altura y en el peso, se acotaron los valores, entre 60-240 y entre 40-180 respectivamente (MedlinePlus. Información de salud para usted, 2023). Estos valores no se decidieron por decisión propia, sino que son las medidas de presión que tiene que tener una persona para poder vivir, desde el valor más bajo (60 o 40) al más alto (240-180), dependiendo del tipo de presión arterial. Estos cambios se hicieron a fin de que la base de datos quedase lista y limpia para su posterior uso.

Las variables cuantitativas son *id*, *age in years*, *height*, *weight*, *ap_hi* y *ap_lo*, y las categóricas *gender* (1-mujer, 2-hombre), *cholesterol* (1-normal, 2-por encima de lo normal, 3-muy por encima de lo normal), *gluc* (1-normal, 2-por encima de lo normal, 3-muy por encima de lo normal), *smoke* (0-no, 1-sí), *alco* (0-no, 1-sí), *active* (0-vida pasiva, 1-vida activa), *cardio* (0-no, 1-sí).

Hay varios factores de riesgos, estos se clasifican en 2 categorías (MedlinePlus. Trusted health information for you, 2024):

- Principales: son factores que intervienen en el riesgo de sufrir la enfermedad
  - Hipertensión arterial
  - Colesterol elevado
  - Diabetes
  - Obesidad y sobrepeso
  - Tabaquismo
  - Inactividad física
  - Sexo
  - Edad
  - Herencia
  - ...
- Secundarios: son factores que pueden incrementar el riesgo de sufrir la enfermedad
  - Estrés
  - Hormonas sexuales
  - Anticonceptivos orales
  - Alcohol

- ...

En este problema, algunos de estos no pueden controlarse, pero otros sí. Algunas de las cosas que se puede hacer para reducir el riesgo de enfermedad cardíaca son:

1. Controlar la presión arterial. Se debe de controlar una vez al año, excepto que se tenga la frecuencia alta, que se tendrá que controlar a intervalos de tiempo menores.
2. Mantener los niveles de colesterol y triglicéridos bajo control. El colesterol puede llegar a obstruir las arterias, esto debe controlarse mediante medicación o cambiando el estilo de vida; otro nivel es el de triglicéridos, tener un nivel alto de estos aumenta el riesgo de sufrir la enfermedad, en concreto más en mujeres.
3. Mantenerse en un peso saludable.
4. Hacer ejercicio con regularidad. Esto fortalece el corazón y mejora la circulación.
5. Limitar el consumo de alcohol. Es recomendable reducir la cantidad de alcohol ingerido, ya que puede elevar la presión arterial y añadir calorías innecesarias. Se aconseja a los hombres no tomar más de dos bebidas alcohólicas al día, mientras que para las mujeres se recomienda no exceder una.
6. Evitar fumar. El fumar incrementa significativamente el riesgo de sufrir un ataque cardiaco y un derrame cerebral.

Capítulo 3

Software

Desarrollo de una aplicación web con Shiny, R y RStudio

En este proyecto, se creó una aplicación web funcional y versátil empleando el paquete "Shiny", junto con el lenguaje de programación R y la plataforma RStudio.

Ventajas de usar Shiny

Shiny fue elegido como el marco principal debido a sus capacidades únicas para desarrollar aplicaciones web interactivas en R. Esta herramienta permitió crear una interfaz intuitiva y dinámica, mejorando considerablemente la interacción del usuario con los datos presentados.

Integración de R y RStudio en el desarrollo

El entorno de R fue fundamental para llevar a cabo los análisis estadísticos y las visualizaciones necesarias en la aplicación. Con su potente capacidad para el manejo de datos y la generación de gráficos, R proporcionó todas las herramientas necesarias para realizar los cálculos requeridos. RStudio, por otro lado, se utilizó como plataforma de desarrollo debido a su entorno de trabajo integrado, que facilita la codificación, depuración y gestión de proyectos de manera eficiente.

Sinergia de herramientas

La integración de Shiny, R y RStudio ha sido crucial para lograr los objetivos de este proyecto. Esta combinación permitió desarrollar una aplicación web robusta y de alto rendimiento. El resultado es una herramienta que ofrece una experiencia de usuario enriquecida e interactiva, proporcionando un entorno accesible y fácil de usar para la exploración y visualización de datos.

3.1. Historia de R

R tiene su origen en el lenguaje de programación S, que fue desarrollado en los Laboratorios Bell en Estados Unidos (Mendoza-Vega, 2018). Debido a que S y sus especificaciones estaban restringidas por ser propiedad de los Laboratorios Bell, Ross Ihaka y Robert Gentleman, de la Universidad de Auckland en Nueva Zelanda, decidieron crear una versión libre y abierta de S. Este proyecto se inició en 1992, y para 1995 ya se había desarrollado una versión preliminar del lenguaje R, alcanzando una versión estable en el año 2000.

En los últimos años, R ha ganado popularidad entre programadores y desarrolladores de software, debido a su amplia gama de técnicas para el análisis estadístico y la gestión de grandes volúmenes de datos.

3.2. Exploración del lenguaje de R

R es un lenguaje de programación de software libre, de código abierto y de tipo interpretado, lo que significa que los comandos o instrucciones se ejecutan directamente sin necesidad de software adicional o de compilación a lenguaje de máquina. Esto simplifica la creación de herramientas para el análisis estadístico y la generación de gráficos.

Además, R permite el desarrollo de diversas técnicas de análisis, como análisis lineales y no lineales, agrupamientos, análisis de series temporales, clasificaciones y estadísticas clásicas.

El desarrollo de R está a cargo del R Development Core Team, aunque otros programadores pueden contribuir con paquetes de código adicionales.

Además de lo anteriormente mencionado, este lenguaje también permite:

- Gestión eficiente del almacenamiento de datos.
- Creación de herramientas para cálculos matriciales y análisis estadístico.
- Funcionalidades gráficas avanzadas.

  Un lenguaje de programación versátil, que incluye bucles, condiciones, funciones recursivas, entre otras características.
- La documentación se presenta en formato LaTeX, ofreciendo una guía completa tanto en versión física como digital.

En cuanto a las ventajas de utilizar este lenguaje de programación son las siguientes:

- En el análisis de big data, reducir el tiempo y esfuerzo es crucial. Este lenguaje disminuye el tiempo necesario para analizar datos al agrupar todas las herramientas esenciales en un solo programa, permitiendo análisis eficientes en menos tiempo.
- Facilita la interpretación de datos mediante el uso de modelos tanto lineales como no lineales.
- La inclusión de gráficos facilita estudios y permite obtener conclusiones rápidamente, posibilitando la comparación de datos en diferentes períodos y monitoreando la evolución de procesos con un margen de error mínimo.
- Este lenguaje permite escribir código limpio y eficiente, lo que mejora la gestión de datos.
- Es compatible con sistemas operativos UNIX, Linux, macOS y Windows sin inconvenientes.
- Es de uso gratuito.

Y entre los avances que se podrían llevar a cabo para mejorar R:

- No incluye soporte para gráficos en 3D ni gráficos dinámicos.
- Puede ser más lento en comparación con otros lenguajes de programación.
- Los algoritmos no están unificados, por tanto es necesario ajustar las opciones de lectura para procesar datos de diferentes paquetes.
- Tiene una compatibilidad limitada con otros lenguajes de programación.
- Es susceptible a ciberataques debido a la falta de sistemas de seguridad integrados.
- Está en constante actualización, lo que requiere aprender sus nuevas características regularmente.

Aunque es posible utilizar R directamente, se recomienda emplear un entorno de desarrollo integrado (EDI)

Si bien se puede ejecutar el código de R desde simples archivos de texto plano, este método resulta ineficiente, especialmente para proyectos complejos.

Un EDI brinda herramientas para escribir y revisar el código, gestionar los archivos que utilizamos, organizar el entorno de trabajo y acceder a otras funciones que aumentan la productividad. Trabajos que serían complicados de realizar sin un EDI se vuelven mucho más manejables.

Existen varias opciones de EDI para R, pero la más conocida es RStudio.

RStudio es muy popular para el desarrollo de aplicaciones, y una de sus herramientas fundamentales para este propósito es Shiny. Shiny es un paquete de R que optimiza el desarollo de aplicacion web interactivas de manera directa desde R. Permite a personas sin conocimientos en diseño web crear de manera rápida una página interactiva para analizar la información.

3.3. Creación de una aplicación

El software R está diseñado para realizar análisis basados en bases de datos existentes. Este software se encarga de dividir la base de datos en conjuntos de entrenamiento, testeo y predicción mediante muestreo aleatorio simple, asegurando resultados consistentes con la fijación de una semilla en cada ejecución del código.

La base de datos incluye información crucial con múltiples variables que influyen en el resultado final, reflejando diversos aspectos y características relevantes para el estudio. Es fundamental estandarizar y comparar uniformemente todas estas variables, realizando los ajustes necesarios para obtener un enfoque preciso y equitativo sobre cómo cada variable afecta al resultado final.

El software divide la base de datos en tres conjuntos, permitiendo al usuario especificar la proporción para cada uno:

- Conjunto de entrenamiento: Utilizado para entrenar y ajustar el modelo, donde el usuario selecciona la proporción de filas de la base de datos. Cada fila proporciona información detallada, incluida la variable objetivo.
- Conjunto de testeo: Empleado para evaluar la calidad del ajuste del modelo previamente entrenado, con el usuario determinando la proporción de filas de la base de datos. Este conjunto también contiene información detallada, incluida la variable objetivo.
- Conjunto de predicción: Utilizado para realizar predicciones sobre nuevos casos, con las filas restantes de la base de datos original. Este conjunto excluye la variable objetivo, ya que no se utiliza para las predicciones.

Este enfoque permite a los usuarios realizar análisis detallados y ajustados desde una base de datos, optimizando el proceso de modelado y predicción en R.

Posteriormente, se le pide al usuario qué algoritmo quiere aplicar a su conjunto de datos, tendrá dos opciones: el algortimo de bagging o el algoritmo de random forest. Dependiendo del algoritmo seleccionado, se le pedirá una cosa u otra al usuario. En el caso del bagging, el usuario tendrá que introducir cuántas filas querrá utilizar en cada árbol de decisión. En caso contrario, si elige el random forest, se le pedirá al usuario tanto el número de filas utilizadas en cada árbol como el número de columnas.

A continuación, se pasará a la ejecución del algoritmo elegido por el usuario. En el caso del bagging, con el conjunto de entrenamiento, se llevará a cabo el ajuste del modelo pidiendo al usuario previamente el número de árboles que quiera obtener. Para ajustar este modelo se usará el número de filas y el número de árboles que le pedimos al usuario y se guardará en una variable. Una vez que se tenga el modelo ajustado, con el conjunto de datos de testeo y con el modelo ajustado se pasará a hacer la evaluación del modelo. Esto se observa y se analiza a través de la bondad

de ajuste. Una vez evaluada la bondad de ajuste se pasa al último paso en el cual se utilizará el conjunto de predicción y el modelo ajustado, como su nombre indica, para hacer las predicciones.

En el caso de que el usuario elija el algoritmo de random forest, el procedimiento es el mismo, lo único que cambia es que a la hora de ajustar el modelo, habrá que tener en cuenta el número de filas y también el número de columnas.

3.4. Shiny

Shiny es un paquete de R que simplifica el desarrollo de aplicaciones web interactivas directamente desde R (Chang, 2020). Permite a personas sin experiencia en diseño web crear rápidamente una página dinámica para explorar datos, sin necesidad de tener conocimientos previos de HTML, CSS o JavaScript.

Es una herramienta poderosa y eficiente para el trabajo, ya que facilita el desarrollo y la prueba de aplicaciones con funcionalidades interactivas.

Una vez que la aplicación está completamente desarrollada y funcionando correctamente, se puede configurar para ejecutarse directamente desde Shiny con el soporte de RStudio, permitiendo a los usuarios acceder a la aplicación a través de un navegador web.

3.4.1. Estructura de una aplicación Shiny

Para iniciar el desarrollo de una aplicación web con Shiny, se comienza con una plantilla inicial (Figura 3.1) (Shiny, 2024):

- Interfaz de usuario (UI): Utiliza funciones integradas de R para construir una interfaz en HTML que permite a los usuarios interactuar con la aplicación. Incluye elementos interactivos para recibir entradas del usuario y mostrar resultados.
- Servidor: Define las instrucciones sobre cómo construir y actualizar los objetos de R que se presentan en la interfaz de usuario. Gestiona la lógica y el procesamiento de datos en respuesta a las acciones del usuario.
- ShinyApp: Es la combinación de la UI y el servidor en una aplicación web. Al ejecutar runApp(), se pone en marcha la aplicación Shiny.

Esta estructura modular de Shiny permite desarrollar aplicaciones web dinámicas y personalizadas con facilidad, utilizando R para la visualización y el análisis de datos de manera interactiva y accesible.

```r
#Cargamos el paquete Shiny
install.packages(shiny)
library(shiny)

#Definimos UI
UI <- fluidPage(

)

#Definimos el servidor
servidor <- function(input, output) {

}

#Ejecutamos la aplicación
shinyApp(ui = UI, server = servidor)
```

Figura 3.1: Aplicación web con Shiny.

Shiny tiene dos componentes principales que forman la estructura básica de la aplicación. Estos componentes son (Figura 3.2): la UI y el servidor. En el archivo “UI.R”, se establece la estética y los elementos de la interfaz de usuario, como botones, casillas de entrada, gráficos, tablas, y más (Kozlowski, 2024). Por otro lado, en el archivo “servidor.R”, se define la lógica y el comportamiento de la aplicación, que incluye la lectura y manipulación de datos, los procesos de cálculo y la generación de visualizaciones.

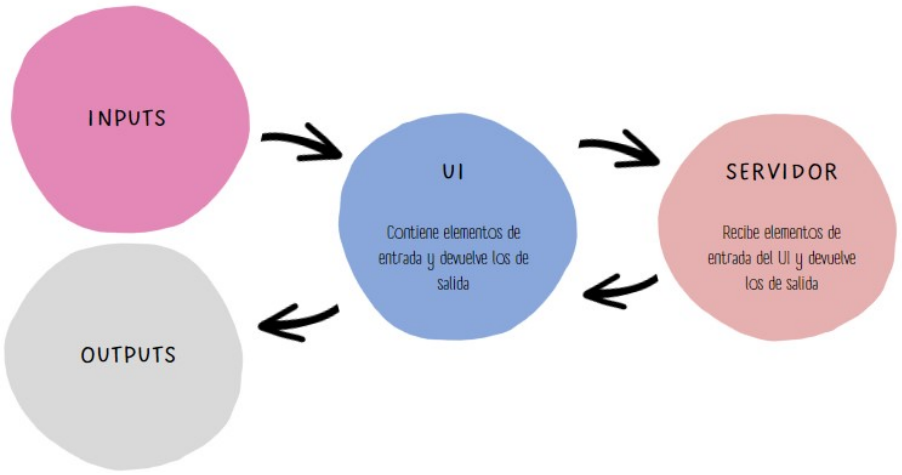

Figura 3.2: Esquema Shiny.

3.4.2. Características

Como se dijo anteriormente, la caracaterística principal de las aplicaciones que se desarrollan con Shiny es la funcionalidad interactiva, lo que facilita el uso de los datos sin el requerimiento de manejar el código subyacente. Shiny se basa en la programación reactiva, la cual consiste en vincular los valores de entrada con los de salida.

Los 3 objetos de esta programación reactiva son (Figura 3.3) (Armas, 2017):

- Fuentes reactivas: Es otro término usado para los inputs del usuario. El servidor Shiny puede acceder a los outputs de la UI mediante los widgets que se han configurado. Cada vez que estos valores se modifican, se transmiten al servidor. Estos pueden ser botonos, campos de texto, menús desplegables, casillas de verificación, ...

- Conductores reactivos: Estos son objetos que existen exclusivamente dentro del servidor Shiny. Generan elementos que solo pueden visualizarse dentro del servidor y ser usados en otras operaciones. Por lo general, dependen de fuentes reactivas.

- Puntos finales reactivos: Son los outputs obtenidos en el servidor y que se envían a UI para poder ser visualizados. Estos pueden ser gráficos, tablas, mapas, ...

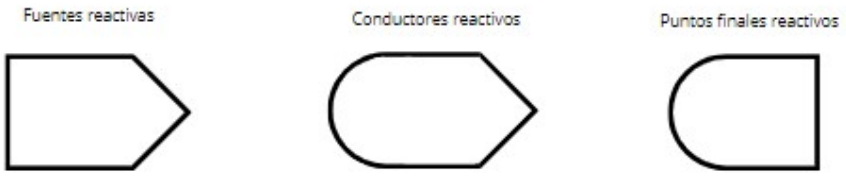

Figura 3.3: Elementos programación reactiva.

Utilizando estas expresiones se aisla el código y, de esta manera, solo se ejecutan cuando se haga algún cambio en las entradas de las cuales dependen y no realizar cálculos innecesarios intermedios que podrían hacer que la aplicación se ralentice. Este enfoque permite una interacción activa y adaptativa utilizando expresiones reactivas.

La aplicación más básica consiste en tomar una fuente reactiva como input y que a partir de esa entrada nos muestre un punto final reactivo. Las fuentes reactivas pueden utilizarse las veces que se quiera para así generar tantos outputs como se quiera.

Los parámetros necesarios para que la aplicación funcione se introducen a partir de widgets, los inputs más comunes se muestran en la Tabla 3.1:

Tabla 3.1: Inputs.

Función	**Widget**
actionButton	Botón de Acción
checkboxGroupInput	Grupo de verificación de casillas
checkboxInput	Casilla individual de verificación
dateInput	Calendario para seleccionar una fecha
dateRangeInput	Conjunto de calendarios para seleccionar un rango de fechas
fileInput	Control para enviar archivos
helpText	Texto de ayuda
numericInput	Campo para introducir números
radioButtons	Conjunto de botones de radio
selectInput	Caja con varias opciones para seleccionar
sliderInput	Barra deslizante
submitButton	Botón para enviar
textInput	Campo para intoducir texto

Y en cuanto a los outputs, estos se generan en el servidor. Shiny proporciona la salida de estos. Las salidas más comunes son las siguientes:

Tabla 3.2: Outputs.

Output	**Función render**	**Widget**
htmlOutput/uiOutput	renderUI	Salida HTML
imageOutput	renderImage	Imágenes
plotOutput	renderPlot	Gráficos
tableOutput	renderTable	Imprimir una tabla (dataframe, matrix, ...)
textOutput	renderText	Strings
verbatimTextOutput	renderPrint	Imprimir texto

3.5. Manual de la app

El objetivo de la UI de la aplicación es facilitar al usuario el proceso de carga de la base de datos y la configuración de los parámetros necesarios para el análisis. Ahora se va a explorar de manera detallada cómo funciona esta interfaz, paso a paso.

Inicialmente, al acceder a la aplicación que se ha desarrollado, los usuarios verán el título que describe la función de la aplicación y su propósito, junto con instrucciones relevantes sobre cómo cargar la base de datos y configurar los parámetros necesarios.

Figura 3.4: Cabecera de la aplicación.

A continuación, la interfaz presenta una sección dedicada a la carga de la base de datos, diseñada para que el usuario pueda introducir la información necesaria para realizar cálculos y análisis.

En esta área, hay un campo de entrada de archivo etiquetado como “Proporcione aquí su base de datos en formato .xlsx”. Los usuarios pueden interactuar con la aplicación y cargar un archivo en formato .xlsx desde su dispositivo. Para hacerlo, solo deben hacer clic en el botón “Browse” y seleccionar el archivo deseado de sus archivos locales para usarlo en el análisis.

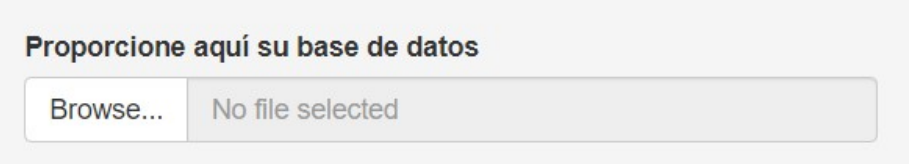

Figura 3.5: Carga de la base de datos.

Después de que el usuario haya introducido la base de datos correctamente, la interfaz muestra unas cajas para poder introducir los parámetros necesarios para que los cálculos y el análisis puedan llevarse a cabo y poder obtener los resultados. Estos parámetros son muy importantes porque permiten al usuario introducirlos según sus preferencias y sus necesidades.

El primer parámetro que se le pide es que nos dé la “Lista de las variables categóricas”. Tendrá que introducir el índice y no el nombre de la columna, esto se especificará, como se mencionó anteriormente, al principio, debajo del título de la aplicación.

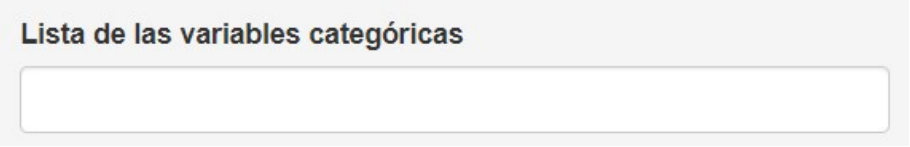

Figura 3.6: Lista de variables categóricas.

El segundo parámetro, es el de “Proporción de datos para el entrenamiento”, en este, también se encontrarán un campo donde tengan que introducir un valor numérico. Este ajuste brinda a los usuarios la posibilidad de determinar la proporción de datos que desea emplear para entrenar el modelo. Como se dijo antes, las especificaciones se pondrán en formato de texto al principio, esto será así con cualquier especificación que se tenga que hacer.

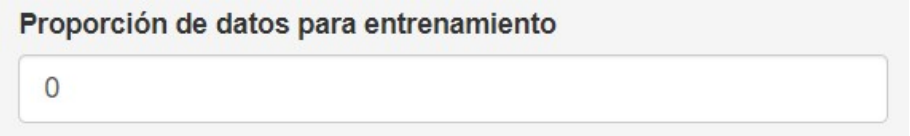

Figura 3.7: Proporción para entrenar.

A continuación, se encuentra el parámetro “Proporción de datos para el testeo”. Este parámetro permite al usuario definir la proporción de datos que se desea reservar para evaluar la precisión y efectividad del modelo. Al igual que con el parámetro anterior, los usuarios ingresarán un valor numérico en el campo correspondiente para indicar cuántos datos de prueba desean utilizar en el análisis.

Figura 3.8: Proporción para testeo.

Además, se usa la programación para calcular el conjunto de datos que se usará para hacer las predicciones, este cálculo será la cantidad restante de datos. La aplicación resta al conjunto de datos total, el conjunto de datos que se usará para el entrenamiento y el conjunto de datos utilizado para el testeo.

Otro parámetro importante es el de "Seleccionar el algoritmo". Con este parámetro se determinará el algoritmo usado para poder pedir otras medidas necesarias pero correspondientes al algoritmo elegido.

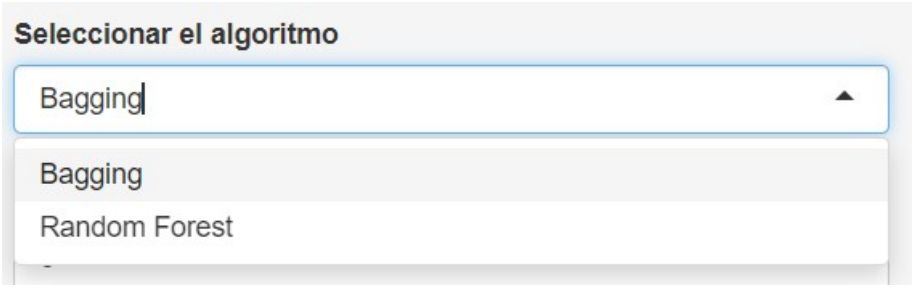

Figura 3.9: Selección del algoritmo.

Los últimos parámetros serán, en caso de haber escogido el algoritmo de bagging, el "Número de árboles" así como el "Número de filas" que el usuario quiere utilizar de la base de datos de entrenamiento. En estos dos casos, los valores que se introducirán serán valores númericos.

Figura 3.10: Datos para el bagging.

Y si, por el contrario, el usuario elige el algoritmo de random forest como algoritmo, aparte de pedirle estos dos últimos parámetros, adicionalmente la interfaz también le preguntará por el "Número de columnas" que quiere utilizar a la hora de ejecutar el algoritmo.

Figura 3.11: Datos para el random forest.

Después de que el usuario haya introducido la base de datos y todos los parámetros necesarios, tales como la lista de variables categóricas, el número de datos de entrenamiento, el número de datos de prueba, el algoritmo utilizado, el número de filas y columnas, y finalmente, el número de árboles, la interfaz muestra una sección donde se presentan los resultados del análisis realizado. Estos resultados están organizados en cuatro secciones distintas para facilitar su lectura y permitir la extracción de conclusiones pertinentes de manera más efectiva.

En la primera sección, se mostrará la proporción de datos que se va a utilizar para las predicciones.

En la segunda sección, se encontrarán dos gráficos resultantes de calcular el modelo usando cualquiera de los dos algoritmos: uno que muestra la importancia de las variables en el modelo, y como segundo las correlaciones entre las variables numéricas de la base de datos. Esto ayudará a comprender mejor los resultados observados en el primer gráfico.

En la tercera sección, se obtienen los resultados generados mediante una función específica. Este resultado corresponde a la bondad de ajuste, que indica qué tan bien se adapta el modelo a los datos de prueba. Para obtener esta medida, se realizan los cálculos y análisis pertinentes utilizando la base de datos y los parámetros establecidos. El resultado se presenta en un formato de texto, claro y accesible para los usuarios, facilitando la comprensión de las conclusiones derivadas del análisis.

En la última sección, se mostrará la matriz resultante obtenida con el modelo ajustado y la proporción de datos destinada a las predicciones. Esta matriz contendrá todas las variables independientes de la base de datos original, junto con una columna final que mostrará las predicciones obtenidas.

Esta aplicación está diseñada para que el usuario pueda cargar una base de datos, configurar los parámetros necesarios y examinar los resultados del análisis de manera sencilla y comprensible. La interfaz, con su diseño claro y organizado, junto con elementos interactivos y descriptivos, permite a los usuarios realizar sus tareas de manera eficiente y obtener información valiosa de los análisis realizados.

3.6. Explicación del código en R

El código para su ejecución está disponible en la sección de anexos de este documento. En esa sección, se incluyen diversas partes que cubren el algoritmo completo para el análisis de datos.

Como se dijo anteriormente, la aplicación Shiny está dividida en dos partes, el UI y el servidor. En la UI, en la interfaz gráfica, solo se añadirán los inputs que debe introducir el usuario para su posterior análisis, estos son: la carga de la base de datos, los índices de las variables categóricas, la proporción de datos para entrenamiento y testeo, la elección del algoritmo a utilizar y los valores de las variables correspondientes al algoritmo elegido.

```
ui <- fluidPage(
  useShinyjs(),
  titlePanel("Aplicación de Bagging y Random Forest"),
  mainPanel("La base de datos que se debe cargar debe tener la extensión .xlsx."),
  mainPanel("La proporción de datos de entrenamiento y de testeo debe ser un número entre 0 y 1."),
  mainPanel("Además, el número de filas debe ser cómo mínimo 1 y como máximo", datos_entrenamiento, "y  el número de columnas debe ser como
  mainPanel("Por último el número de árboles debe ser como mínimo 1."),
  sidebarLayout(
    sidebarPanel(
      fileInput("file", "Proporcione aquí su base de datos", accept = ".xlsx"),
      textInput("cat_vars", "Lista de las variables categóricas", ""),
      numericInput("prop_train", "Proporción de datos para el entrenamiento", value = 0, min = 0, max = 1, step = 0.1),
      numericInput("prop_test", "Proporción para testeo", value = 0, min = 0, max = 1, step = 0.1),
      selectInput("algorithm", "Seleccionar el algoritmo", choices = c("Bagging", "Random Forest")),
      conditionalPanel(
        condition = "input.algorithm == 'Bagging'",
        numericInput("num_arboles_b", "Número de árboles", value = 0, min = 1),
        numericInput("num_filas_b", "Número de filas", value = 0, min = 1)
      ),
      conditionalPanel(
        condition = "input.algorithm == 'Random Forest'",
        numericInput("num_arboles_r", "Número de árboles", value = 0, min = 1),
        numericInput("num_filas_r", "Número de filas", value = 0, min = 1),
        numericInput("num_columnas_r", "Número de columnas", value = 0, min = 1, max = 12)
      ),
      actionButton("run", "Ejecutar")
    ),
    mainPanel(
      plotOutput("importancia"),
      plotOutput("correlación"),
      verbatimTextOutput("bondad_ajuste"),
      tableOutput("matriz_prediccion")
    )
  )
)
```

Figura 3.12: UI.

Por otro lado, en el servidor es donde se llevan a cabo todos los cálculos relacionados con el análisis. En primer lugar, el código utiliza la función “read_excel” de la biblioteca “readxl” para leer un archivo Excel. Los datos de este archivo se almacenan en un objeto llamado “datos”. Esta función simplifica la carga de la base de datos en R, preparándola para su procesamiento y análisis posterior.

```
server <- function(input, output, session) {
  observeEvent(input$ok, {
    updateNumericInput(session, "prop_train", value = input$initial_prop_train)
    updateNumericInput(session, "prop_test", value = input$initial_prop_test)
    updateNumericInput(session, "num_arboles_b", value = input$initial_num_arboles_b)
    updateNumericInput(session, "num_filas_b", value = input$initial_num_filas_b)
    updateNumericInput(session, "num_arboles_r", value = input$initial_num_arboles_r)
    updateNumericInput(session, "num_filas_r", value = input$initial_num_filas_r)
    updateNumericInput(session, "num_columnas_r", value = input$initial_num_columnas_r)
    removeModal()
  })

  observeEvent(input$run, {
    req(input$file)

    datos <- read_excel(input$file$datapath)
```

Figura 3.13: Encabezado del servidor.

Una vez cargados los datos en R, con la función “lapply” se convertirán las variables categóricas de la base de datos a factor. Este cambio es útil y necesario cuando se va a trabajar con variables categóricas en el análisis y modelado de los datos, para así poder aplicar el algoritmo para clasificación.

```
variables_categoricas <- as.numeric(unlist(strsplit(input$cat_vars, ",")))
nombres_categoricas <- colnames(datos)[variables_categoricas]
datos[nombres_categoricas] <- lapply(datos[nombres_categoricas], factor)
```

Figura 3.14: Variables categóricas.

Después de preparar los datos, se pasará a la división de estos en conjunto de entrenamiento, testeo y predicción. Esta división se hará pidiéndole al usuario la proporción de datos que quiere para entrenamiento y para testo. La cantidad de datos para predicción se hará con la diferencia de estos dos conjuntos anteriores y la base de datos original. Esto se hará con la función "round" para que no salgan decimales y el número de filas salga exacto. Una vez se tiene cuántas filas se quieren dedicar para cada conjunto de datos, con la función "sample" se tomará una muestra aleatoria para el primer conjunto, por ejemplo, el de entrenamiento.

A continuación, con la función "setdiff" se obtendrá la diferencia entre las filas del conjunto inicial menos las filas que se utilizarán para el conjunto de entrenamiento. Con el conjunto de testeo se hará lo mismo, primero se calcula el número de filas que se utilizará para dicho conjunto con la función "sample" y, después, de nuevo, con la función "setdiff" se hará la diferencia entre el conjunto de datos inicial menos las filas de entrenamiento restándole ahora las filas del conjunto de datos de testeo.

Y con el conjunto de datos de testeo, las filas serán las sobrantes, que se hará de nuevo con la función "setdiff".

```
prop_train <- input$prop_train
prop_test <- input$prop_test
proporcion <- c(prop_train, prop_test, 1 - prop_train - prop_test)
num_individuos <- nrow(datos)
num_variables <- ncol(datos)
datos_entrenamiento <- round(num_individuos * proporcion[1])
datos_testeo <- round((num_individuos - datos_entrenamiento) * proporcion[2])
datos_prediccion <- num_individuos - datos_entrenamiento - datos_testeo

set.seed(123456)
filas_entrenamiento <- sample(1:num_individuos, datos_entrenamiento, replace = FALSE)
matriz_entrenamiento <- datos[filas_entrenamiento,]
num_individuos2 <- setdiff(1:num_individuos, filas_entrenamiento)
filas_testeo <- sample(num_individuos2, datos_testeo, replace = FALSE)
matriz_testeo <- datos[filas_testeo,]
filas_prediccion <- setdiff(num_individuos2, filas_testeo)
matriz_prediccion <- datos[filas_prediccion, 1:(num_variables - 1)]
```

Figura 3.15: Entrenamiento, testeo y predicción.

Una vez realizadas las divisiones, se obtienen las variables "matriz_entrenamiento", "matriz_testeo" y "matriz_prediccion" que contienen solo las filas correspodientes a los conjuntos de entrenamiento, testeo y predicción. Estos conjuntos de datos serán empleados en fases posteriores del análisis y modelado.

Después de haber hecho esta división, se pasa a los algoritmos utilizados para el análisis.

3.6.1. Funciones

En esta parte del código se hablará de las funciones utilizadas en el algoritmo para el análisis de datos. Primero, se explicará la función llamada "randomForest". Toda esta parte de explicación de funciones se sigue encontrando dentro del servidor.

Primera función: randomForest

La función "randomForest" del paquete "randomForest" implementa el algoritmo de bagging (Bootstrap Aggregating) y el de random forest, para la creación de modelos de agregación, específicamente los bosques aleatorios. Se explicará cómo trabaja esta función si el usuario elige el algoritmo de bagging.

En primer lugar, bootstrap sampling, la función "randomForest" selecciona múltiples muestras de bootstrap del conjunto de datos de entrenamiento teniendo en cuenta el número de filas que tiene que tomar, este número es introducido por el usuario y se pondrá como argumento en la función como "sampsize". Cada muestra de bootstrap es una muestra aleatoria con reemplazo, lo que significa que algunos ejemplos pueden aparecer varias veces en la misma muestra, mientras que otros pueden no aparecer. Otro argumento que se indicará en la función será el de "mtry" que informa del número de columnas que se quiere utilizar, y, para este algoritmo, se usan todas menos la dependiente, que se obligará al usuario que sea la última columna de la base de datos original.

En segundo lugar, para cada muestra de bootstrap, "randomForest" construye un árbol de decisión. Este número de árboles también será introducido por el usuario y guardado en la variable "num_arboles_b" y se pondrá en la función como argumento "ntree". A diferencia de los árboles individuales, que se ajustan completamente a los datos, los árboles en un bagging no se podan. Esto permite que cada árbol pueda aprender relaciones complejas en sus respectivas muestras.

Esta función se almacenará con el nombre de "modelo_b", que se utilizará posteriormente para hacer las predicciones de este algoritmo.

```
modelo_b <- randomForest(x = data.frame(matriz_entrenamiento[1:(num_variables-1)]),
                         y = data.frame(matriz_entrenamiento[num_variables])[,1],
                         ntree = input$num_arboles_b,
                         sampsize = input$num_filas_b,
                         mtry = num_variables - 1,
                         importance = TRUE)
```

Figura 3.16: Modelo bagging.

En tercer lugar, se llevará a cabo el promediado de predicciones. Para la clasificación, cada árbol de decisión en el bagging vota por una clase y la clase final se determina por la mayoría de votos entre todos los árboles.

Con la función "randomForest" también se ha medido la importancia de cada variable en el modelo. Esto se ha hecho calculando cuánto disminuye la precisión de las predicciones cuando los valores de esa variable se permutan aleatoriamente.

En resumen, la función "randomForest" en R es utilizada para aplicar el bagging, crear múltiples árboles de decisión a partir de muestras de bootstrap del conjunto de entrenamiento y luego combinar las predicciones de estos árboles para mejorar la precisión y la robustez del modelo final.

Y en cuanto a la utilización de la función "randomForest" para el algoritmo random forest, al igual que el algoritmo del bagging, el primer paso es el bootstrap sampling, este genera varias muestras bootstrap a partir del conjunto de datos de entrenamiento teniendo en cuenta el número de filas y de columnas que tomará el algoritmo, esto vendrá determinado con el argumento "sampsize" y "mtry". Estos valores vienen dados por el usuario y se guardarán en las variables "num_filas_r" y "num_columnas_r". Cada una de estas muestras son una selección aleatoria con reemplazo, lo que significa que algunos datos pueden repetirse y otros pueden no estar presentes.

Después, el algoritmo pasa a construir los árboles de decisión. Hace tantos árboles como lo indique el usuario. Este número se guardará en la variable "num_arboles_r". A diferencia de los árboles de decisión tradicionales, en random forest, cada vez que se divide un nodo durante la construcción del árbol, solo se considera un subconjunto aleatorio de las variables en lugar de todas las disponi-

bles. Esto introduce una mayor variabilidad entre los árboles y ayuda a reducir la correlación entre ellos. Este modelo se guardará en un objeto llamado "modelo_r".

```
modelo_r <- randomForest(x = data.frame(matriz_entrenamiento[1:(num_variables-1)]),
                         y = data.frame(matriz_entrenamiento[num_variables])[,1],
                         ntree = input$num_arboles_r,
                         sampsize = input$num_filas_r,
                         mtry = input$num_columnas_r,
                         importance = TRUE)
```

Figura 3.17: Modelo random forest.

Los árboles de decisión en un random forest crecen completamente y no se podan. Esto significa que cada árbol se ajusta tanto como sea posible a la muestra de bootstrap sin ningún tipo de restricción para reducir su tamaño.

A continuación, se procederá al promediado de predicciones. Para la clasificación, cada árbol de decisión en el bosque aleatorio hace una predicción y la clase final se determina por votación por mayoría entre todos los árboles al igual que en el algoritmo del bagging.

Por último, con esta función también se mide la importancia de las variables en el modelo. Se calcula evaluando cuánto disminuye la precisión de las predicciones cuando los valores de esa variable se permutan aleatoriamente.

En resumen, la función "randomForest" en R construye un conjunto de árboles de decisión a partir de muestras de bootstrap y subconjuntos aleatorios de características para crear un modelo más preciso y robusto. La combinación de múltiples árboles, que cada uno ofrece predicciones individuales, ayuda a reducir el sobreajuste y mejora la generalización del modelo a nuevos datos.

Segunda función: barplot

Esta función se ha empleado para obtener el gráfico de la importancia de las variables en ambos algoritmos. Utilizará como argumentos la traspuesta de las importancias que se obtienen al calcular el modelo.

Tercera función: corrplot

La función "corrplot" del paquete "corrplot" en el trabajo se ha implementado en el código para obtener las correlaciones entre las variables numéricas de la base de datos y así poder obtener una mejor interpretación del "barplot" de las variables por su importancia. Únicamente se debe indicar con un "sapply" que solo coja las variables numéricas.

Se ha decidido calcular la correlación entre las variables numéricas del conjunto de datos, pues este paso es esencial para entender la relación lineal entre las variables. La correlación nos permite identificar variables que están altamente relacionadas entre sí.

Cuarta función: predict

Esta función se ha utilizado para calcular la bondad de ajuste y las predicciones del modelo ajustado. Para el bagging, en primer lugar, se utilizará esta función en la que los argumentos son el modelo ajustado y los datos de testeo.

En segundo lugar, para las predicciones, se tendrán como argumentos el modelo ajustado, que ha sido entrenado con múltiples muestras de bootstrap y los datos de predicción.

```
predicciones_b1 <- predict(modelo_b, matriz_testeo[1:(num_variables-1)])
bondad_ajuste_b <- 100 * sum(predicciones_b1 == data.frame(matriz_testeo[num_variables])[,1]) / datos_testeo

predicciones_b2 <- predict(modelo_b, matriz_prediccion)
predicciones_b2 <- matriz_prediccion[,num_variables]
```

Figura 3.18: Bondad de ajuste del bagging.

La función "predict" es utilizada para el random forest de manera muy similar. Para la bondad de ajuste se sigue el mismo procedimiento y para las predicciones también. Tanto en el bagging como en el random forest el modelo utilizado como argumento es el resultante de haber aplicado la función "randomForest".

```
predicciones_r1 <- predict(modelo_r, matriz_testeo[1:(num_variables-1)])
bondad_ajuste_r <- 100 * sum(predicciones_r1 == data.frame(matriz_testeo[num_variables])[,1]) / datos_testeo

predicciones_r2 <- predict(modelo_r, matriz_prediccion)
predicciones_r2 <- matriz_prediccion[,num_variables]
```

Figura 3.19: Bondad de ajuste del random forest.

3.6.2. Salida final

Una vez definidas las funciones anteriores, se emplean para generar los resultados finales. A continuación, se describe paso a paso lo que sucede en esta parte:

En primer lugar, se mostrará por pantalla la proporción de datos destinados a la predicción teniendo en cuenta la proporción de datos introducida para el entrenamiento y para el testeo.

En segundo lugar, se llama a la función "randomForest" tanto para el algoritmo de bagging como el random forest. Esta función coge el conjunto de datos de entrenamiento y calcula el modelo entrenado. Este modelo se guarda en la variable "modelo_b" o "modelo_r" respectivamente. Esta función construye árboles de decisión a partir de los datos de entrenamiento así como la importancia de cada una de las variables. Como salidas en la interfaz se tendrán el gráfico de la importancia de las variables y el gráfico de correlaciones para entender mejor el primer gráfico.

Por último, se llama a la función "predict". Con esta función se obtiene tanto la bondad de ajuste como las predicciones. Para la bondad de ajuste se utilizará el modelo obtenido previamente con la función "randomForest" y los datos de testeo, y para la segunda utilización, para obtener las predicciones, se hace uso también del modelo ajustado anteriormente pero esta vez aplicado a los datos de predicción.

Para la bondad de ajuste, "predict" se usa para generar predicciones sobre el conjunto de datos de testeo y luego se comparan estas predicciones con los valores reales. Este proceso ayuda a concluir cuán bien el modelo se ajusta a los datos que no fueron utilizados durante el entrenamiento. Y para obtener las predicciones, "predict" toma el modelo ajustado y aplica los parámetros de este modelo a los datos de predicción para generar predicciones.

La bondad de ajuste y la matriz de predicciones obtenidas serán las dos últimas salidas que se mostrarán en la interfaz. Con la matriz de predicciones el usuario podrá tomar las conclusiones oportunas respecto a su estudio y al diagnóstico.

Capítulo 4

Resultados

En esta sección, se mostrarán los resultados obtenidos utilizando la aplicación Shiny. Esta herramienta permite cargar una base de datos en formato Excel y realizar análisis y cálculos basados en los datos proporcionados. Sin embargo, antes de iniciar el análisis, se lleva a cabo un paso crucial en el proceso de análisis de datos: la limpieza y preparación de las variables. Este paso facilitará la ejecución y la interpretación del análisis posterior.

La base de datos utilizada en el estudio proporciona parámetros para la predicción de enfermedades cardiovasculares y contiene información de 68.399 pacientes, con los cuales se ha trabajado en su totalidad. Es importante para estos análisis contar con una base de datos de gran tamaño, ya que, en el análisis de predicción de una enfermedad médica, cuanto más pacientes tengamos mejor será la precisión a la hora de introducir uno nuevo y darle el diagnóstico.

Después de introducir los datos, se realizaron análisis detallados y cálculos basados en la información proporcionada. Se obtuvieron resultados precisos y confiables que se pudieron explicar de manera clara y efectiva.

Tras explorar diversas opciones mediante una investigación exhaustiva, se ajustaron las proporciones de registros asignados a los conjuntos de entrenamiento, prueba y predicción, así como se modificaron los valores de filas, columnas y árboles para obtener el modelo, resultando ser una estrategia efectiva. Se determinó que la configuración más óptima es la siguiente:

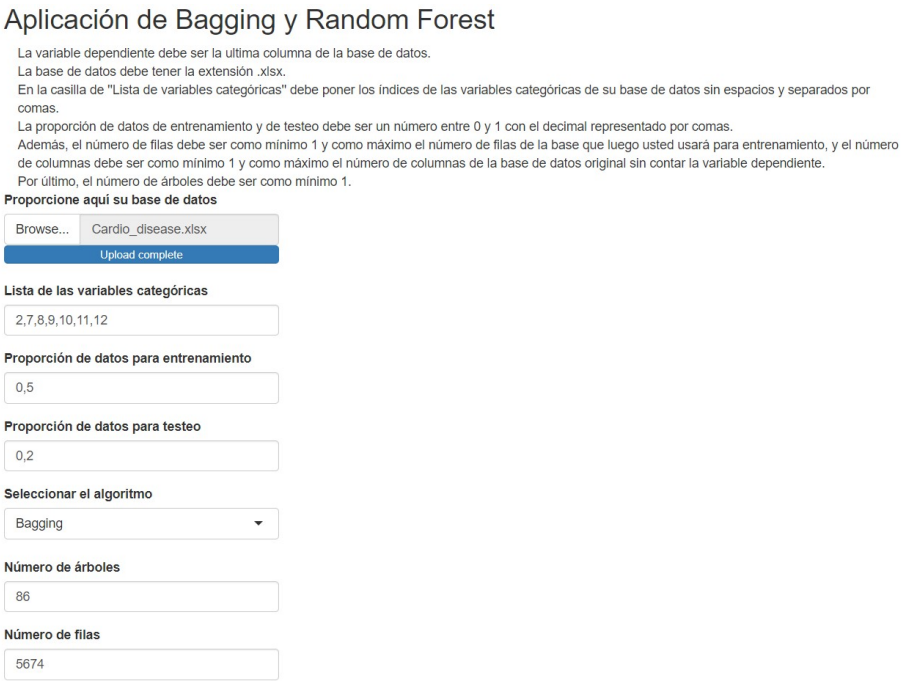

Figura 4.1: Configuración del bagging.

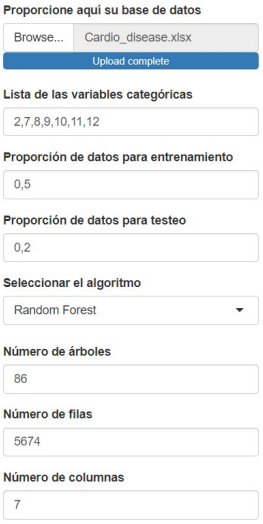

Figura 4.2: Configuración del random forest.

- Lista de variables categóricas: 2,7,8,9,10,11,12
- Proporción de datos para entrenamiento: 0.5
- Proporción de datos para testeo: 0.2
- Número de árboles: 86
- Número de filas: 5674
- Número de columnas: 7

En las Figuras 4.1 y 4.2, se observa la elección de la base de datos llamada “Cardio_disease”. Contiene, como se ha dicho anteriormente, 68.399 pacientes. Los datos se dividieron en tres con-

juntos: entrenamiento, testeo y predicción, para así poder llevar a cabo el análisis. La división de estos conjuntos la hace el usuario, dependiendo de la proporción de datos que quiera para los conjuntos de entrenamiento y de testeo. El usuario podrá ver la proporción para la predicción, en la salida que se puede observar en la Figura 4.3.

```
[1] "Proporción de datos para predicción:  0.3"
```

Figura 4.3: Proporción para predicción.

Al igual que el usuario elije la proporción de datos para cada conjunto, también elije todos los demás parámetros que se utilizarán a la hora de obtener el modelo.

Una vez aplicada la app, se obtuvieron los siguientes resultados.

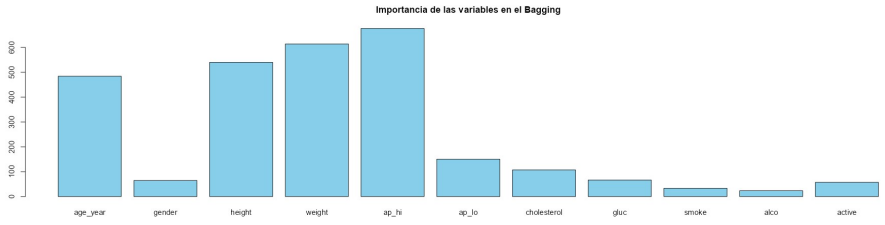

Figura 4.4: Importancia de las variables bagging.

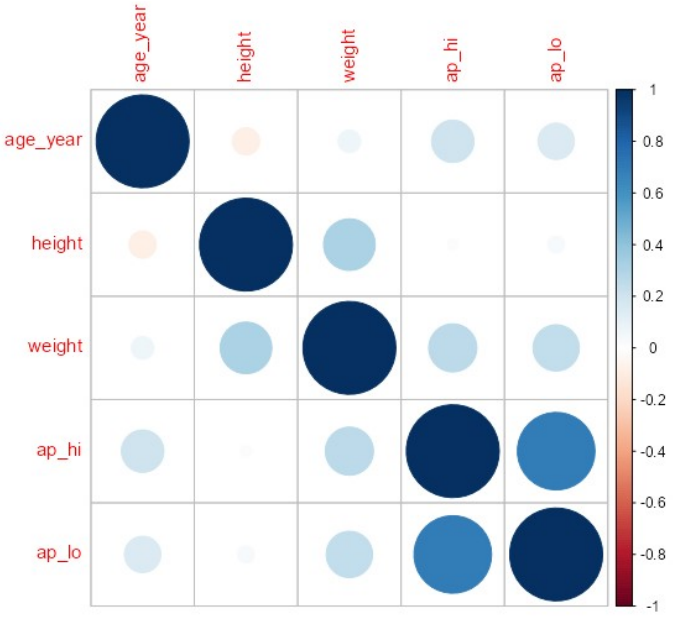

Figura 4.5: Correlaciones bagging.

Los resultados obtenidos, si se elije el algoritmo de bagging, son los que se muestran en las Figuras 4.4 y 4.5. Entonces, las conclusiones que se obtuvieron observando estos dos gráficos, en un modelo de bagging, indican que la presión arterial sistólica (“ap_hi”) es la más influyente, seguida por el peso (“weight”) y por la altura (“height”). El gráfico de correlaciones muestra una moderada correlación entre peso y altura, y entre las presiones arteriales sistólica (“ap_hi”) y diastólica (“ap_lo”). La edad (“age_year”) también es una variable importante, aunque en menor medida que

las anteriores. Otras variables como la presión arterial diastólica, el colesterol, la glucosa, el fumar, el consumo de alcohol, la actividad física y el género tienen menor importancia en el modelo. Este análisis sugiere que el control del peso y la presión arterial son cruciales para la prevención de enfermedades cardiovasculares.

```
[1] "Bondad de ajuste: 72.7924%"
```

Figura 4.6: Bondad de ajuste bagging.

En esta segunda salida se observa la bondad de ajuste del modelo, que en este caso fue un valor del 72.7924 %. Es importante tener en cuenta que, para que la bondad de ajuste sea considerada buena, este valor tiene que ser entre el 60 % y el 80 %. El que se ha obtenido está en ese rango, por lo que se considera que el modelo es bueno y ajusta bien.

Matriz de predicciones

age_year	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	respuesta
40.00	1	140.00	68.00	100.00	70.00	1	1	0	0	0	0
59.00	1	140.00	50.00	110.00	70.00	1	1	0	0	1	0
61.00	1	140.00	52.00	130.00	80.00	1	1	0	0	1	1
50.00	2	140.00	90.00	140.00	90.00	1	1	0	0	1	1
53.00	1	140.00	68.00	140.00	70.00	1	1	0	0	1	1
63.00	1	140.00	60.00	130.00	80.00	1	1	0	0	1	1
44.00	1	140.00	44.00	100.00	70.00	1	1	0	0	1	0
47.00	1	140.00	50.00	140.00	90.00	3	1	0	0	1	1
52.00	2	140.00	90.00	140.00	90.00	1	1	0	0	1	1
48.00	1	140.00	62.00	120.00	80.00	1	1	0	0	1	0
43.00	1	140.00	64.00	120.00	80.00	1	1	0	0	1	1
47.00	1	140.00	75.00	120.00	80.00	1	1	0	0	0	1
60.00	1	140.00	73.00	150.00	90.00	2	2	0	0	1	1
53.00	2	140.00	70.00	140.00	70.00	2	2	0	0	1	0
62.00	1	140.00	41.00	130.00	90.00	1	1	0	0	1	1
54.00	1	140.00	80.00	140.00	80.00	1	1	0	0	1	1
52.00	1	140.00	77.00	140.00	90.00	1	1	0	0	1	1
57.00	2	140.00	100.00	140.00	100.00	2	1	0	0	1	1
55.00	1	140.00	80.00	100.00	60.00	2	1	0	0	0	1
53.00	2	140.00	57.00	130.00	90.00	1	1	0	0	1	1

Figura 4.7: Matriz de predicciones bagging.

Y como última salida, se encuentra una tabla en la que se muestran todas las variables independientes de la base de datos inicial en columnas, y en filas se tiene cada uno de los pacientes que han formado parte del estudio, y, como última columna, la de predicciones obtenida con los datos de predicción y el modelo. Evaluar estos resultados es fundamental para medir el rendimiento y la precisión del modelo empleado. La Figura 4.7 muestra estos resultados en detalle.

Si por el contrario, se elije el algoritmo de random forest, el tipo de salidas que se obtendrán será el mismo.

En primer lugar, los datos proporcionados en cuanto a la proporción de datos para entrenamiento, proporción de datos para testeo, el número de filas y el número de árboles serán los mismos. Lo que tendrá que introducir el usuario como nuevo parámetro será el número de columnas utilizadas.

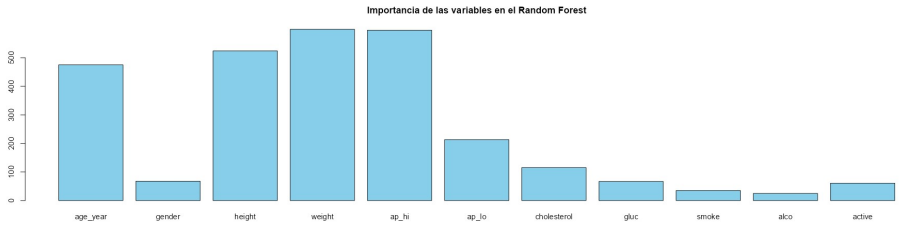

Figura 4.8: Importancia de las variables random forest.

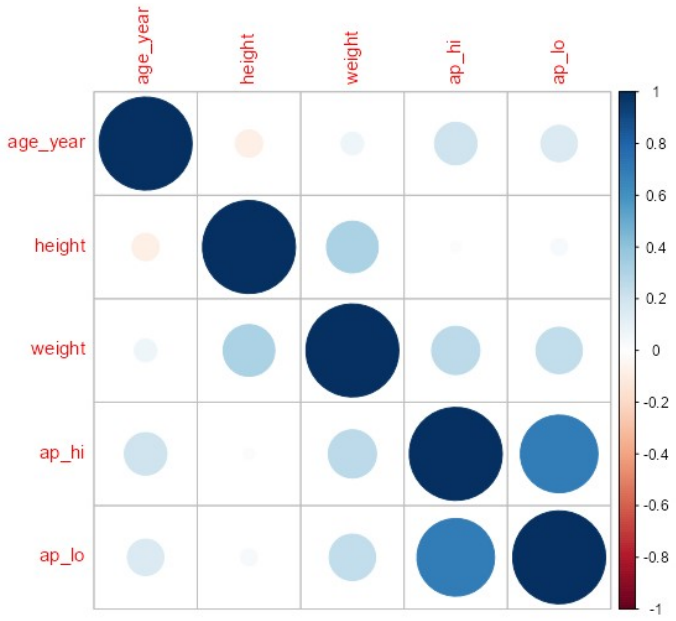

Figura 4.9: Correlaciones random forest.

Entonces, observando las Figuras 4.8 y 4.9, las conclusiones que se obtienen de este algoritmo son que en un modelo para el algoritmo random forest, el peso es la variable más influyente, seguida por la altura y la presión arterial sistólica (“ap_hi”). La edad (“age_year”) y la presión arterial diastólica también son importantes, aunque en menor medida, sobre todo esta última. Otras variables como el colesterol, la glucosa, el fumar, el consumo de alcohol, la actividad física y el género tienen menor importancia en el modelo. El gráfico de correlaciones nos muestra lo mismo que en el algoritmo del bagging.

```
[1] "Bondad de ajuste: 73.1871%"
```

Figura 4.10: Bondad de ajuste random forest.

En la segunda salida se observa la bondad de ajuste del modelo, que ahora es de un valor del 73.1871 %. Este valor está entre 60 % y 80 %, por lo que el modelo es bueno y ajusta bien. También se observa que es algo mejor que el obtenido con el algoritmo anterior. Este resultado es algo que se esperaba debido a que el algortimo de random forest es una mejora del bagging.

Matriz de predicciones

age_year	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	respuesta
40.00	1	140.00	68.00	100.00	70.00	1	1	0	0	0	0
59.00	1	140.00	50.00	110.00	70.00	1	1	0	0	1	0
61.00	1	140.00	52.00	130.00	80.00	1	1	0	0	1	1
50.00	2	140.00	90.00	140.00	90.00	1	1	0	0	1	1
53.00	1	140.00	68.00	140.00	70.00	1	1	0	0	1	1
63.00	1	140.00	60.00	130.00	80.00	1	1	0	0	1	1
44.00	1	140.00	44.00	100.00	70.00	1	1	0	0	1	0
47.00	1	140.00	50.00	140.00	90.00	3	1	0	0	1	1
52.00	2	140.00	90.00	140.00	90.00	1	1	0	0	1	1
48.00	1	140.00	62.00	120.00	80.00	1	1	0	0	1	0
43.00	1	140.00	64.00	120.00	80.00	1	1	0	0	1	1
47.00	1	140.00	75.00	120.00	80.00	1	1	0	0	0	0
60.00	1	140.00	73.00	150.00	90.00	2	2	0	0	1	1
53.00	2	140.00	70.00	140.00	70.00	2	2	0	0	1	1
62.00	1	140.00	41.00	130.00	90.00	1	1	0	0	1	1
54.00	1	140.00	80.00	140.00	80.00	1	1	0	0	1	1
52.00	1	140.00	77.00	140.00	90.00	1	1	0	0	1	1
57.00	2	140.00	100.00	140.00	100.00	2	1	0	0	1	1
55.00	1	140.00	80.00	100.00	60.00	2	1	0	0	0	1
53.00	2	140.00	57.00	130.00	90.00	1	1	0	0	1	1

Figura 4.11: Matriz de predicciones random forest.

Y como última salida, como antes, se encuentra una tabla en la que se muestran todas las variables independientes de la base de datos inicial en columnas y, como última, las predicciones que se han calculado con los datos de predicción y el modelo. En filas se tiene cada uno de los pacientes que han formado parte del estudio. La Figura 4.11 muestra estos resultados en detalle.

Estos resultados son útiles para realizar análisis estadísticos adicionales, comparar variables, identificar patrones y profundizar en la comprensión de los datos dentro del ámbito del estudio de la salud cardiovascular. La clasificación de los datos proporciona información crucial que puede asistir a profesionales e investigadores de la salud en la realización de análisis más detallados y significativos de los datos recolectados.

Capítulo 5

Conclusiones

Con respecto al trabajo realizado y los objetivos mencionados anteriormente, se ha llegado a las siguientes conclusiones:

- Se ha desarrollado una aplicación web robusta utilizando R, que integra de manera efectiva las técnicas de bagging y random forest para la predicción de variables respuesta. La aplicación proporciona una interfaz accesible, permitiendo a los usuarios cargar datos, seleccionar parámetros de modelado y visualizar resultados de predicción. La implementación de estas técnicas en la aplicación se realizó de manera que los usuarios, incluidos aquellos con menos experiencia en programación, puedan beneficiarse de los potentes métodos de aprendizaje automático para realizar predicciones precisas.
- Se ha definido de manera clara y precisa la técnica de bootstrap (bagging), explicando cómo se generan múltiples subconjuntos de datos a partir de un conjunto de datos original mediante muestreo con reemplazo. Además, se ha demostrado matemáticamente cómo esta técnica contribuye a la reducción de la varianza en los estimadores. La demostración incluyó la descripción del proceso mediante el cual se agregan y se obtienen los diferentes modelos hasta llegar a la predicción.
- En cuanto a la técnica de random forest, se ha proporcionado una definición completa y detallada, explicando cómo se construye un conjunto de árboles de decisión utilizando el método de bagging y la selección aleatoria de características en cada división del árbol. La demostración matemática incluyó la justificación teórica de cómo la combinación de múltiples árboles de decisión reduce el riesgo de sobreajuste y mejora la precisión del modelo.
- Se ha realizado una búsqueda cuidadosa y completa para seleccionar una base de datos adecuada para nuestro estudio. Finalmente, se eligió una base de datos de enfermedades cardiovasculares debido a su relevancia clínica y la calidad de los datos disponibles. La base de datos seleccionada proporciona información que permite que sea efectiva la aplicación de las técnicas de bagging y random forest para la predicción de enfermedades cardiovasculares.
- Se han implementado y aplicado los algoritmos de bagging y random forest utilizando R a la base de datos seleccionada. Los resultados obtenidos demostraron que estas técnicas pueden mejorar la precisión de las predicciones, proporcionando una herramienta valiosa para los médicos en el diagnóstico de estas enfermedades. Estas técnicas pueden ser útiles en la práctica clínica.

En resumen, se ha cumplido con todos los objetivos propuestos, desarrollando una aplicación web funcional y definiendo las técnicas de bagging y random forest, además de aplicarlas a una base de datos real para contribuir al campo de la medicina predictiva.

Bibliografía

Armas, M. (2017). *Clase 3 - Shinys* [Accessed: 27-06-2024]. https://rstudio-pubs-static.s3.amazonaws.com/339312_4e857d2d6a054cfd88cf749afc9f8a95.html

Beltrán-Martínez, M. (2001). *Minería de datos*. Benemérita Universidad Autónoma de Puebla. Facultad de Ciencias de la Computación.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees (CART)*. Belmont, CA, USA: Wadsworth International Group.

Chang, W. (2020). *shiny: Web Application Framework for R* [Accessed: 24-06-2024]. https://cran.r-project.org/web/packages/shiny/index.html

Cutro, A. (2010). *Introducción a la Minería de Datos* [Accessed: 27-06-2024]. https://www.dataprix.com/es/mineria-datos-aplicada-encuesta-permanente-hogares/introduccion-mineria-datos

Dorado-Díaz, P. (2023). *Árboles de decisión*. Universidad de Salamanca.

Fundación Española del Corazón (FEC). (2024). *Tratamientos* [Accessed: 27-06-2024]. https://fundaciondelcorazon.com/informacion-para-pacientes/tratamientos.html

Hashimzade, R. (2020). *Cardio Disease* [Accessed: 24-06-2024]. https://www.kaggle.com/datasets/raminhashimzade/cardio-disease

IBM. (2024a). *¿Qué es bagging?* [Accessed: 27-06-2024]. https://www.ibm.com/es-es/topics/bagging

IBM. (2024b). *¿Qué es un árbol de decisión?* [Accessed: 27-06-2024]. https://www.ibm.com/es-es/topics/decision-trees

IBM. (2024c). *¿Qué es un bosque aleatorio?* [Accessed: 27-06-2024]. https://www.ibm.com/es-es/topics/random-forest

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning: with applications in R*. Springer Nature.

Jiawei, H., Micheline, K., & Jian, P. (2006). *Data Mining. Concepts and Techniques*. Morgan Kaupmann.

Kozlowski, D. (2024). *Shiny apps* [Accessed: 27-06-2024]. https://diegokoz.github.io/EEA2019/clase%203/shiny/clase_shiny/explicacion.nb.html

MedlinePlus. Información de salud para usted. (2023). *Presión arterial* [Accessed: 27-06-2024]. https://medlineplus.gov/spanish/ency/anatomyvideos/000013.htm

MedlinePlus. Información de salud para usted. (2024). *Qué es la enfermedad cardiovascular* [Accessed: 27-06-2024]. https://medlineplus.gov/spanish/ency/patientinstructions/000759.htm

MedlinePlus. Trusted health information for you. (2024). *How to Prevent Heart Disease* [Accessed: 24-06-2024]. https://medlineplus.gov/howtopreventheartdisease.html

Mendoza-Vega, J. B. (2018). *R para principiantes* [Accessed: 27-06-2024]. https://bookdown.org/jboscomendoza/r-principiantes4/un-poco-de-historia.html

Shiny. (2024). *Welcome to Shiny* [Accessed: 27-06-2024]. https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/index.html

Ye, N. (2013). *Data mining: theories, algorithms, and examples*. CRC press.