UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

Proyecto de Investigación previo a la obtención del título de Ingeniero en Sistemas

Título del Proyecto de Investigación:

"COMPARACIÓN DE TÉCNICAS DE CLASIFICACIÓN EN LA RECOMENDACIÓN DE PRODUCTOS BANCARIOS"

Autor:

Ricardo Rafael Villarroel Molina

Director del Proyecto de Investigación:

Ing. Iván Fredy Jaramillo Chuqui

Quevedo – Los Ríos – Ecuador.

2017

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo, **Ricardo Rafael Villarroel Molina**, declaro que la investigación aquí descrita es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Técnica Estatal de Quevedo, puede hacer uso de los derechos correspondientes a este documento, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Ricardo Rafael Villarroel Molina

C.C # 0503679631

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

CERTIFICACIÓN DE CULMINACIÓN DEL PROYECTO DE INVESTIGACIÓN

El suscrito, Ing. Iván Jaramillo, Docente de la Universidad Técnica Estatal de Quevedo, certifica que el aspirante **Villarroel Molina Ricardo Rafael,** realizó el Proyecto de Investigación de grado titulado "**COMPARACIÓN DE TÉCNICAS DE CLASIFICACIÓN EN LA RECOMENDACIÓN DE PRODUCTOS BANCARIOS"** previo a la obtención del título de Ingeniero en Sistemas bajo mi dirección, habiendo cumplido con las disposiciones reglamentarias establecidas para el efecto.

_____

Ing. Iván Fredy Jaramillo Chuqui

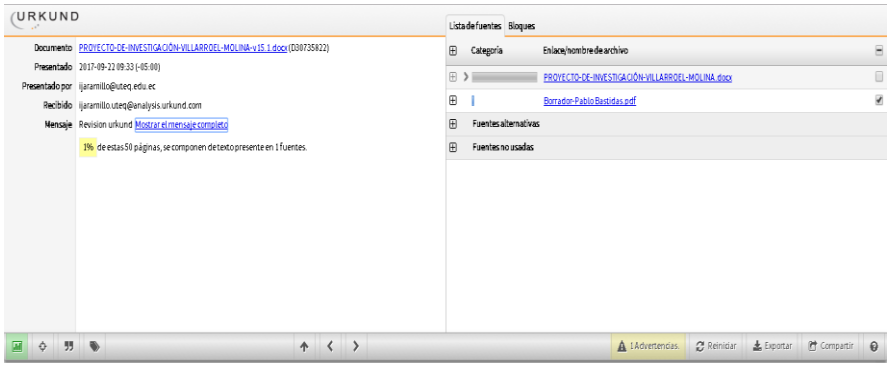
DIRECTOR DE PROYECTO DE INVESTIGACIÓN

CERTIFICADO DEL REPORTE DE LA HERRAMIENTA DE PREVENCIÓN DE COINCIDENCIA Y/O PLAGIO ACADÉMICO

INFORME DEL DIRECTOR DE TESIS SOBRE EL SISTEMA URKUND

Ing. Iván Fredy Jaramillo Chuqui, en calidad de director del proyecto de investigación "**COMPARACIÓN DE TÉCNICAS DE CLASIFICACIÓN EN LA RECOMENDACIÓN DE PRODUCTOS BANCARIOS"**, me permito manifestar a usted lo siguiente:

Que, el Sr. Villarroel Molina Ricardo Rafael, egresado de la facultad de Ciencias de la Ingeniería, ha cumplido con las correcciones pertinentes, e ingresado su proyecto de investigación al sistema URKUND, tengo a bien certificar la siguiente información sobre el informe del sistema anti plagio con un porcentaje del 1%**.**



Ing. Iván Fredy Jaramillo Chuqui

DIRECTOR DE PROYECTO DE INVESTIGACIÓN

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

PROYECTO DE INVESTIGACIÓN

Título:

"COMPARACIÓN DE TÉCNICAS DE CLASIFICACIÓN EN LA RECOMENDACIÓN DE PRODUCTOS BANCARIOS"

Presentado a la Comisión Académica como requisito previo a la obtención del título de Ingeniero en Sistemas.

Aprobado por:

PRESIDENTE DEL TRIBUNAL
Ing. Vicuña Pino Ariosto Eugenio

MIEMBRO DEL TRIBUNAL
Ing. Mora Secaira Janeth Inés

MIEMBRO DEL TRIBUNAL
PhD. Puris Cáceres Amilkar Yudier

QUEVEDO – LOS RÍOS – ECUADOR
2017

AGRADECIMIENTOS

En especial a Dios por bendecirme en cada paso que doy y darme la fuerza necesaria para llegar a este punto de mi vida y cumplir cada uno de mis objetivos.

A mi familia por ser un pilar muy importante al brindarme su apoyo incondicional, sus concejos y la manera en que cada día me inspiraron a la finalización de mi carrera, sin ellos nada de esto sería posible.

A mi director de proyecto Ing. Iván Jaramillo que con sus amplios conocimientos me han guiado para la consecución del mismo, a los demás docentes por impulsar el desarrollo de mi vida profesional, y a cada una de las personas que de una u otra manera me brindaron su apoyo durante este tiempo de estudio, agradezco a mis amigos Kevin Wong, Ronald Chica, Jacob Reyes, José Mosquera y Darwin Álava con los que compartí muchos momentos, siempre ayudándonos mutuamente en el transcurso de nuestra carrera.

DEDICATORIA

Dedico este trabajo a mis

queridos padres:

Freddy Villarroel y Marisel Molina.

A mis hermanas:

Lourdes, Lisbeth y Lucy Villarroel.

A mi novia

Cristina Loor.

Porque todos ellos siempre

apoyaron cada una de mis decisiones.

RESUMEN EJECUTIVO Y PALABRAS CLAVES

La presente investigación tiene como título*: "*Comparación de Técnicas de Clasificación en la Recomendación de Productos Bancarios", en la cual se aborda la generación de recomendaciones como un problema de clasificación, realizando una comparación de los clasificadores proporcionados por los algoritmos de Naïve Bayes (NB), Redes Neuronales Multicapa Feed-forward (NNET), Máquinas de Soporte Vectorial (SVM) y Multivariate Adaptive Regression Splines (MARS), los cuales se seleccionaron en base a su aplicabilidad en el presente problema de clasificación; para lo cual se utilizaron las métricas de AUC, tiempo de entrenamiento y de prueba, teniendo como finalidad el poder determinar un algoritmo que proporcione clasificadores con un alto desempeño para generar las recomendaciones de productos bancarios.

Palabras clave: minería de datos, clasificación, recomendación.

ABSTRACT AND KEYWORDS

The present research is entitled "Comparison of Classification Techniques in the Recommendation of Banking Products", which addresses the generation of recommendations as a classification problema, comparing the classifiers provided by the Naïve Bayes (NB), Multi-layer Feed-forward Neural Networks (NNET), Support Vector Machines (SVM) and Multivariate Adaptive Regression Splines (MARS) algorithms, which were selected based on their applicability in the present classification problem; for this we used the AUC, training and test time, with the purpose of being able to determine an algorithm that provides classifiers with a high performance to generate recommendations of banking products.

Keywords: Data mining, classification, recomendations.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

CÓDIGO DUBLÍN

Título:	"COMPARACIÓN DE TÉCNICAS DE CLASIFICACIÓN EN LA RECOMENDACIÓN DE PRODUCTOS BANCARIOS"		
Autor:	Villarroel Molina, Ricardo Rafael		
Palabras claves:	*minería de datos*	*clasificación*	*recomendación*
Fecha de publicación:			
Editorial:			
Resumen:	Resumen.- La presente investigación aborda la generación de recomendaciones como un problema de clasificación, realizando una comparación de los clasificadores proporcionados por los algoritmos de Naïve Bayes (NB), Redes Neuronales Multicapa Feed-forward (NNET), Máquinas de Soporte Vectorial (SVM) y Multivariate Adaptive Regression Splines (MARS). **Abstract.- The present research addresses the generation of recommendations as a classification problem, comparing the classifiers provided by the Naïve Bayes (NB), Multi-layer Feed-forward Neural Networks (NNET), Support Vector Machines (SVM) and Multivariate Adaptive Regression Splines (MARS) algorithms.**		
Colaborador:	Ing. Iván Fredy Jaramillo Chuqui		
Descripción:	153 hojas: Dimensiones, 29 x 21 cm+ CD-ROM		
URI:	(en blanco hasta cuando se dispongas los repositorios)		

Comentado [V941]:

INTRODUCCIÓN

Las entidades bancarias y financieras tienen estrategias para gestionar las necesidades o intereses de sus clientes, causando que se planteen objetivos para poder satisfacer de una manera eficaz esas necesidades [1]. Diversas son las posibilidades para lograr esos objetivos, una de estas es la aplicación de minería de datos [2].

La minería de datos se define como el proceso de descubrir patrones interesantes y conocimientos en grandes cantidades de datos [3, p. 8]. Este descubrimiento de patrones se lo realiza mediante el uso de diferentes técnicas estadísticas, inteligencia artificial, machine learning, etc [3, p. 23]. Las cuales se aplican mediante la implementación de diversos algoritmos de acuerdo al tipo de problema o la dimensión del mismo.

La clasificación dentro de la minería de datos es la tarea que permite diferenciar entre distintas clases, existiendo varias técnicas para realizar esta tarea. Las aplicaciones de técnicas de clasificación incluye los sistemas de recomendación [4, p. 48], aprobación de créditos, marketing de segmentos, diagnósticos médicos, efectividad de un tratamiento, ubicación de tiendas, entre otros [5].

El uso de recomendaciones de productos generadas por algoritmos es usado ya en algunos bancos del mundo como el Georgia Bank [6]. Esta recomendación de productos también puede ser vista como un problema de clasificación, por lo que al existir varias técnicas de clasificación y que cada una de estas técnicas estén implementadas en uno o varios algoritmos; es importante determinar que algoritmo de clasificación es el más adecuado para generar estas recomendaciones.

En esta investigación se compara mediante las métricas del área bajo la curva ROC (AUC), el tiempo de entrenamiento (tiempo que necesita un algoritmo en construir el clasificador) y el tiempo de prueba (tiempo que necesita el modelo en generar las predicciones del conjunto de pruebas) a cuatro algoritmos de clasificación supervisada: Naïve Bayes (NB), Redes Neuronales Multicapa Feed-forward (NNET), Máquinas de Soporte Vectorial (SVM) y Multivariate Adaptive Regression Splines (MARS). Para luego determinar un algoritmo apropiado para realizar la recomendación de productos bancarios.

También se evalúan las necesidades de preprocesado que tenían los datos, para poder obtener un conjunto de datos de mejor calidad y más adecuado para construir los clasificadores a partir de ellos. Siendo el tratamiento de valores faltantes, la reducción de datos y la discretización procesos fundamentales en el preprocesamiento de los datos.

CAPÍTULO I

CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN

1.1. Problema de investigación.

1.1.1. Planteamiento del problema.

La clasificación conforma parte de las tareas de minería de datos [7] y puede ser clasificación supervisada o no supervisada, la clasificación supervisada es utilizada generalmente para asignar un conjunto de datos a una clase (grupo) o categoría predefinida, con el principal objetivo de predecir con exactitud la clase a la que pertenece cada elemento [8], [9, p. 14], para la tarea de clasificación supervisada se realizan dos pasos: primero es la construcción del modelo de clasificación (o clasificador) a partir de un conjunto de datos de entrenamiento y segundo es usar ese modelo en un conjunto de datos de prueba para medir el rendimiento y precisión del clasificador [9, p. 14].

Como en toda tarea de minería de datos, el rendimiento y la precisión del clasificador depende de las características del conjunto de datos de entrenamiento como el tamaño o el nivel de ruido presente [3, p. 83,99], razón por la cual surgen diferentes técnicas para la tarea de clasificar como las basadas en árboles de decisión, reglas de asociación, redes neuronales, máquinas de soporte vectorial, entre otros [8]; en cada una de estas técnicas se han desarrollado algoritmos los cuales tienen ventajas y desventajas al utilizarlas [9, p. 18], [10].

Por ejemplo, los algoritmos de clasificación basados en árboles de decisión tienen la desventaja que cuando el tamaño de los datos de entrenamiento contienen millones de instancias, estos no caben en la memoria y se vuelve ineficiente la construcción del clasificador [3, p. 347], [4], esto es sumamente importante analizar ya que estos mismos algoritmos en la fase de entrenamiento necesitan de grandes cantidades de datos para poder proporcionar un clasificador de calidad [11, p. 298]. En cambio los basados en redes neuronales proporcionan modelos que son difíciles de entender para el ser humano [3, p. 406] y experimentalmente se ha demostrado que son sensibles al ruido [11, p. 331].

Los algoritmos de clasificación que se han aplicado a varios problemas en el ámbito bancario, tales como la recomendación [12], marketing directo en bancos [13]–[15], evaluación individual de créditos bancarios [16], etc; se eligieron en base al tipo de problema

de clasificación y a la calidad de predicción que los clasificadores proporcionaron a partir de los datos de entrenamiento en los que fueron construidos.

Por lo que al tener a disposición un conjunto de datos de los clientes de un banco y los productos que esos clientes han accedido; se plantea la determinación de un algoritmo de clasificación apropiado para la recomendación de productos bancarios, teniendo en cuenta las características que este conjunto de datos presenta, tales como estar conformados por gran cantidad de registros (instancias), presencia de datos faltantes y el desbalanceo de clases.

1.1.1.1.Diagnóstico.

La alta cantidad de instancias y otras características que el conjunto de datos presenta hace que un algoritmo de clasificación pueda disminuir o aumentar su desempeño en la generación de recomendaciones.

1.1.1.2.Pronóstico.

La elección inadecuada de un algoritmo de clasificación supervisada generará recomendaciones que no concuerden con los intereses o necesidades de los clientes del banco. Siendo esto un gran problema en el objetivo de servir mejor a sus clientes.

1.1.2. Formulación del problema.

¿Cuál de los algoritmos de clasificación supervisada es apropiado para generar recomendaciones de productos bancarios?

1.1.3. Sistematización del problema.

¿Cómo preparar los datos para usarlos en los algoritmos de clasificación?
¿Cómo se seleccionarán los algoritmos de las diferentes técnicas de clasificación?
¿Cómo se determinará que un algoritmo de una técnica de clasificación supervisada es apropiado para la recomendación de productos bancarios?

1.2. Objetivos.

1.2.1. Objetivo general.

Determinar que algoritmo de clasificación supervisada es apropiado para la recomendación de productos bancarios mediante el análisis de datos provenientes de una entidad financiera europea.

1.2.2. Objetivos específicos.

- ✓ Aplicar las técnicas de limpieza, reducción, transformación y discretización de datos, para obtener un conjunto de datos apropiado para el proceso de clasificación.

- ✓ Seleccionar los algoritmos de clasificación que mejores resultados han obtenido en investigaciones similares en los últimos años.

- ✓ Comparar los clasificadores obtenidos por los algoritmos seleccionados mediante la métrica del área bajo la curva ROC (AUC), el tiempo de entrenamiento y el tiempo de prueba.

1.3. Justificación

La minería de datos puede ser usada por varias industrias incluyendo la financiera y la bancaria. La existencia del conjunto de datos de un banco europeo hace posible la realización de muchas investigaciones que empleen técnicas de minería de datos, ya que esos datos se asemejan a los que se encuentran habitualmente en los registros de los bancos de nuestra región y contienen registros de los datos de los clientes y los productos del banco que ellos han accedido. La recomendación de productos, segmentación de clientes y el análisis de cesto de compra; podrían ser análisis válidos para realizar a partir de este conjunto de datos para poder ayudar a servir de mejor manera a los clientes del banco.

El determinar un algoritmo de clasificación que proporcione clasificadores que se construyan a partir de las características e intereses del cliente para recomendar productos bancarios proporcionaría una ayuda al banco a satisfacer las necesidades que tienen sus clientes, las cuales son específicas respecto a préstamos y servicios financieros que proporciona un banco [1]. Además que dentro de las técnicas de filtrado de recomendaciones existentes, la técnica de recomendación basada en modelos puede recomendar un conjunto de elementos rápidamente debido a que tiene ya un modelo pre-calculado [17] y esto es favorable debido a que los clientes de los bancos generalmente acceden a productos bancarios durante largos periodos de tiempo como meses o años.

Este estudio pretende constituir una fuente de referencia para futuros proyectos de investigación o proyectos de minería de datos aplicados a la tarea de clasificación supervisada en datos de los bancos de la región y ser una fuente de información del comportamientos de los algoritmos frente a datos de origen bancario y cuan escalable son, ya que es difícil conocer el comportamiento de los algoritmos frente a determinados problemas o circunstancias, esto es demostrado por el teorema del “No free lunch” [18].

CAPÍTULO II

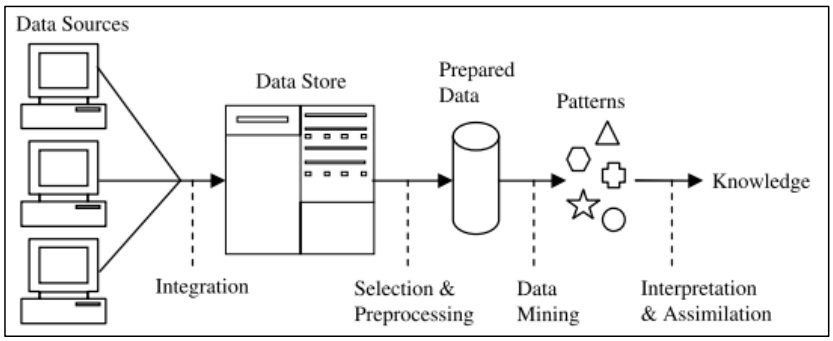
FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

2.1. Marco conceptual.

2.1.1. Descubrimiento del conocimiento.

Descubrimiento del conocimiento (Knowledge Discovery) ha sido definido como la "extracción no trivial de información implícita, desconocida y potencialmente útil de los datos". Es un proceso del cual la minería de datos forma sólo una parte, aunque central [19]. En la *Ilustración 1* muestra una versión ligeramente idealizada del proceso completo del Descubrimiento del Conocimiento.

Ilustración 1: Proceso de descubrimiento del conocimiento.



FUENTE: BRAMER [19].

2.1.2. Minería de datos.

La minería de datos se define como el proceso de descubrir patrones en los datos [20, p. 5]. En [21] se define a la minería de datos como "el proceso de utilizar una variedad de herramientas de análisis de datos para descubrir patrones y relaciones en los datos que se pueden utilizar para hacer predicciones válidas".

Minería de datos se puede decir que es un proceso creativo que requiere un número de diferentes habilidades y conocimientos, este proceso necesita un enfoque estándar para poder traducir los problemas de negocio en tareas de minería de datos, sugerir transformaciones de

datos apropiadas y técnicas de minería de datos, y proveer maneras de evaluar la efectividad de los resultados y documentar la experiencia [22, p. 29].

En minería de datos, dos tipos de datos hay que distinguir, los datos que no tienen ninguna variable especialmente designada para predecir el valor de esa variable para las instancias que aún no se han visto y los que sí tienen esa variable designada. En el primer caso se trata de datos no etiquetados y en el segundo de datos etiquetados [19, p. 5].

Según los tipos de datos utilizados en la minería de datos se conoce dos enfoques: aprendizaje supervisado y aprendizaje no supervisado [19, p. 5].

- ✓ **Aprendizaje supervisado**: Es también conocida como minería de datos dirigida. En este enfoque, las variables observadas se dividen en variables explicativas y variables dependientes. El principal objetivo es determinar una relación entre estos dos tipos de variables. En las técnicas de minería de datos dirigida, los valores de la variable dependiente deben estar bien definidos para una parte suficientemente grande del conjunto de datos [8].

- ✓ **Aprendizaje no supervisado**: En este, no hay distinción entre variables dependientes y variables explicativas, y por lo que son tratadas igual. La principal diferencia entre el aprendizaje supervisado y el aprendizaje no supervisado es que el aprendizaje supervisado requiere que el valor de la variable dependiente sea conocido y bien definido, mientras que en el aprendizaje no supervisado se desconoce la variable dependiente o se conoce su valor para un pequeño número de casos [8].

Típicamente un proceso de minería de datos consiste en 3 etapas: preprocesamiento de datos, análisis de datos y la interpretación de resultados [4, p. 40].

2.1.2.1.Procesos de minería de datos.

A continuación, se detallan los procesos que son realizados en la minería de datos.

- ✓ **Preprocesamiento.**

Las bases de datos del mundo real de hoy en día son altamente susceptibles a datos ruidosos, perdidos e incoherentes debido a su tamaño normalmente enorme (a menudo varios gigabytes o más) y su probable origen de múltiples fuentes heterogéneas. Los datos de baja calidad dan lugar a resultados de minería de baja calidad [3]. Existen varias técnicas de pre procesamiento de datos:

- ✓ **Limpieza de datos:** Se puede aplicar para eliminar o remplazar valores faltantes y corregir inconsistencias en los datos [3].
- ✓ **Integración de datos:** Combina datos de múltiples fuentes en un almacén de datos coherente, como un almacén de datos [3].
- ✓ **Reducción de datos:** Puede reducir el volumen de los datos, pero manteniendo estrechamente la integridad de los datos, lo cual produciría resultados parecidos o casi parecidos si se aplicara a los datos originales sin aplicar la reducción.[3].
- ✓ **Transformación y discretización de datos:** La normalización por ejemplo, es donde los datos se escalan para caer dentro de un intervalo más pequeño como 0,0 a 1,0 [3].

El uso de estas técnicas puede mejorar la precisión y la eficiencia de los algoritmos de minería que implican mediciones de distancia. Estas técnicas no son mutuamente excluyentes; Pueden trabajar juntas. Por ejemplo, la limpieza de datos puede implicar transformaciones para corregir datos erróneos, como por ejemplo transformando todas las entradas para un campo de fecha en un formato común [3].

- ✓ **Análisis de datos.**

Los patrones frecuentes son patrones (conjuntos de elementos, subsecuencias o subestructuras) que aparecen con frecuencia en un conjunto de datos. Por ejemplo, un conjunto de elementos, como leche y pan, que aparecen frecuentemente juntos en un conjunto de datos de transacción es un conjunto de elementos frecuente. Una subsecuencia, como comprar primero una PC, luego una cámara digital, y luego una tarjeta de memoria, si ocurre frecuentemente en una base de datos de historial de compras, es un patrón secuencial (frecuente) [3].

Una subestructura puede referirse a diferentes formas estructurales, tales como subgrafos, subárboles o subestaciones, que pueden combinarse con conjuntos de elementos o subsecuencias. Si una subestructura ocurre con frecuencia, se llama patrón estructurado (frecuente). Encontrar patrones frecuentes desempeña un papel esencial en minería de asociaciones, correlaciones y muchas otras relaciones interesantes entre los datos. Además, ayuda en la clasificación de datos, el agrupamiento y otras tareas de minería de datos. Por lo tanto, la extracción de patrones frecuentes se ha convertido en una importante tarea de minería de datos y un tema enfocado en la investigación de minería de datos [3].

2.1.2.2.Fuentes de datos.

Las fuentes de datos que se utilizan para la minería de datos pueden incluir bases de datos, almacenes de datos, la Web, otros repositorios de información o datos que se transmiten dinámicamente a un sistema [3].

2.1.2.3.Tipos de variables.

En general, hay muchos tipos de variables que se pueden utilizar para medir las propiedades de un objeto [19]. Al menos existen seis tipos de variables, entre las principales tenemos las siguientes:

- ✓ **Variable nominal** es una variable que se utiliza para colocar objetos en categorías, ejemplo el nombre o el color de un objeto [19].

- ✓ **Variable binaria** es un caso especial de una variable nominal que toma sólo dos valores posibles: verdadero o falso, 1 o 0, etc. [19].

- ✓ **Variable ordinal** es similar a la variable nominal, excepto que una variable ordinal tiene valores que se pueden ordenar en un orden significativo, por ejemplo pequeño, mediano y grande [19].

- ✓ **Variable entera** es una variable que toma valores que son enteros genuinos, por ejemplo “número de hijos” [19].

- ✓ **Variable escalonada en intervalos** es una variable que toma valores numéricos que se miden a intervalos iguales desde un punto cero u origen. Sin embargo, el origen no implica una ausencia real de la característica medida, por ejemplo la temperatura medida en Fahrenheit o Celsius [19].

- ✓ **Variable de escala proporcional (ratio)** es similar a la variable escalonada, excepto que el punto cero refleja la ausencia de la característica medida, por ejemplo la temperatura de Kelvin y el peso molecular [19].

2.1.2.4.Atributos categóricos y continuos

Aunque la distinción entre diferentes categorías de variables puede ser importante en algunos casos, muchos sistemas de minería de datos dividen las variables también llamados (atributos) en sólo dos tipos prácticos [19]:

- ✓ **Categórico** correspondiente a atributos nominales, binarios y ordinales [19].

- ✓ **Continuo** correspondiente a atributos enteros, de escala en intervalos y de escala proporcional [19].

2.1.2.5.Funcionalidades o tareas de minería de datos.

Existe una serie de funcionalidades o tareas de minería de datos. Estas incluyen la caracterización y la discriminación; la extracción de patrones frecuentes, asociaciones y correlaciones; clasificación y regresión; análisis de agrupamiento; y el análisis de valores anómalos. Estas se utilizan para especificar los tipos de patrones que se encuentran en las tareas de minería de datos [3].

Las tareas en minería de datos pueden clasificarse en dos categorías: descriptivas y predictivas. Las tareas descriptivas caracterizan las propiedades de los datos en un conjunto

de datos de destino. Las tareas predictivas realizan inducción sobre los datos actuales con el fin de hacer predicciones [3], [7].

2.1.3. Clasificación en minería de datos.

La clasificación es una tarea predictiva muy popular de la minería de datos, que por lo general usa técnicas de aprendizaje supervisado [23]. La clasificación apunta a aprender desde patrones etiquetados un modelo (que de aquí en adelante denominaremos clasificador) capaz de predecir la etiqueta (o clase) para muestras de datos futuros, nunca vistas antes [23].

El conjunto de atributos de un conjunto de datos de clasificación se divide en dos subconjuntos: el primero que contiene las características de entrada, los atributos que actuarán como predictores y el segundo subconjunto que contiene los atributos de salida, las llamadas clases o etiquetas asignadas a cada instancia [23]. Según el tipo del segundo subconjunto de atributos, el que contiene las clases, se puede distinguir los diferentes tipos de problemas de clasificación [23]. Los cuales principalmente son los problemas de clasificación binaria, clasificación multi-clase y clasificación multi-etiqueta [24]. A continuación, una descripción de cada uno de estos tipos de problemas de clasificación y sus características:

- ✓ Los problemas de clasificación binaria tienen en el conjunto de datos un solo atributo de salida, y este puede tomar solo uno de dos valores posibles no superpuestos, por ejemplo 1 o 0, verdadero o falso, etc. Estos valores también son conocidos como clase positiva y la otra como clase negativa [23].

- ✓ Los problemas multi-clase también tienen un solo atributo de salida, pero este atributo puede tomar uno y solo uno de un conjunto de n valores de clases posibles no superpuestas [23].

- ✓ Los problemas de clasificación multi-etiqueta al contrario de los anteriores dos tipos, tienen un subconjunto de L atributos de salida y cada uno de estos atributos solo pueden contener valores binarios [23].

Para cada uno de estos tipos de problemas existen algoritmos específicos para poder realizar la clasificación de los datos ya sea por medio de adaptaciones de algoritmos o implementaciones propias para poder abordar tales problemas, pero también existen métodos de transformación de problemas, los cuales generalmente consisten en simplificar los problemas multi-clase o los problemas multi-etiqueta a problemas de clasificación binaria, ya que este último tipo es el más fácil que podemos encontrar [23], un ejemplo de estos métodos de transformación de problemas es el uno-vs-todos el cual transforma un problema de clasificación multi-clase de n clases a n problemas de clasificación binaria [25], también existen métodos de transformación de un problema de clasificación multi-etiqueta a múltiples problemas de clasificación binarios [23, Ch. 2.4.1].

De aquí en adelante el contenido se centrará exclusivamente en la clasificación binaria, abordando los temas de las técnicas de clasificación binarias, las medidas de evaluación de un clasificador binario, la comparación de clasificadores por medio de sus medidas de evaluación y la comparación estadística de clasificadores.

2.1.3.1.Técnicas de clasificación.

La clasificación se aplica para la clasificación de conjuntos de datos multivariados y univariados, algunas de las técnicas básicas usan árboles de decisión, clasificadores bayesianos, redes de creencias bayesianas y clasificadores basados en reglas. Algunos acercamientos más recientes a la clasificación son las máquinas de soporte vectorial (SVM), clasificación basada en minería de reglas de asociación, los clasificadores K-NN, razonamiento basado en casos, y los algoritmos genéticos [7].

2.1.3.1.1. Técnica de clasificación bayesiana.

Los clasificadores Bayesianos son clasificadores probabilísticos. Ellos pueden predecir probabilidades de pertenencia a la clase tales como la probabilidad de que una instancia determinada pertenezca a una clase particular. Estos clasificadores suponen que el efecto de un valor de un atributo en una clase dada es independiente de los valores de los otros atributos [3, Ch. 8.3].

Estos clasificadores se construyen en base al teorema de Bayes, algunos estudios que comparan algoritmos de clasificación han encontrado que el rendimiento de un clasificador bayesiano simple puede ser comparable con el de un árbol de decisión y clasificadores de redes neurales. Los clasificadores bayesianos también han exhibido alta precisión y velocidad cuando se aplican a bases de datos grandes [3, Ch. 8.3]. Uno de los algoritmos pertenecientes a esta técnica es el algoritmo Naïve Bayes.

El algoritmo de Naïve Bayes (NB de aquí en adelante) da una forma de combinar la probabilidad previa y las probabilidades condicionales en una sola fórmula, que se pueden usar para calcular la probabilidad de cada una de las posibles clasificaciones a su vez. Hecho esto se elige la clasificación con el valor más grande [19].

La probabilidad de que un evento x ocurra en un conjunto de datos es calculada usando la frecuencia de la ocurrencia del evento x en el conjunto de datos dividido por el número total de instancias se conoce como probabilidad previa. En cambio la probabilidad de que ocurra un evento si sabemos que un atributo tiene un valor particular (o que varios atributos tienen valores particulares) se denomina probabilidad condicional de que ocurra el evento [19].

Básicamente el algoritmo NB se basa en el siguiente proceso: dado un conjunto de k clasificaciones mutuamente excluyentes y exhaustivas c_1, c_2, …, c_k, que tienen probabilidades previas $P(c_1)$, $P(c_2)$, …, $P(c_k)$, respectivamente, y n atributos a_1, a_2, …, a_n, que para una instancia dada tienen valores v_1, v_2, …, v_n, respectivamente [19]. Puede demostrarse que la probabilidad posterior de la clase c_i que ocurre para la instancia especificada es proporcional a:

$$P(c_i) \times P(a_1 = v_1 \; y \; a_2 = v_2 \ldots y \; a_n = v_n \mid c_i)$$

Ecuación 1: Cálculo de la probabilidad posterior de la clase $\boldsymbol{c_i}$.

Lo que es igual a la siguiente ecuación que es una representación matemática más técnica y simplificada de la *Ecuación 1*.

$$P(c_i) \times \prod_{j=1}^{n} P(a_j = v_j \mid clase = c_i)$$

Ecuación 2: Cálculo de la probabilidad posterior simplificada de la clase $\boldsymbol{c_i}$.

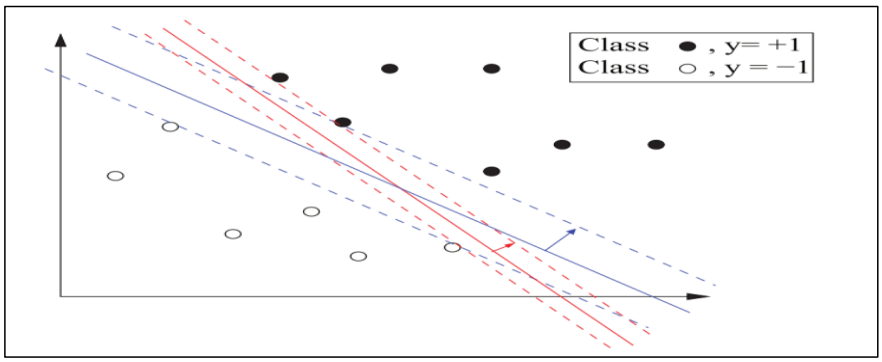
Cuando se utiliza el algoritmo de NB para clasificar una serie de nuevas instancias, la manera más eficaz de comenzar es calcular todas las probabilidades previas y también todas las probabilidades condicionales que implican a un atributo, aunque no todas pueden ser necesarias para clasificar una instancia particular [19].

Hay que tener presente que al usar este tipo de algoritmo es obviamente necesario que todos sus atributos sean del tipo categórico y que el estimar las probabilidades por frecuencias relativas puede dar una mala estimación si el número de instancias con una combinación de atributo/valor dada es pequeña [19].

2.1.3.1.2. Técnica basada en máquinas de soporte vectorial.

El uso de la técnica de máquinas de vectores de soporte en la tarea de clasificación binaria se lo realiza mediante la separación lineal (por un hiperplano[1]) maximizando el margen entre esa separación lineal utilizando una formulación de programación cuadrática [26], como ejemplo de este proceso tenemos la *Ilustración* ***1****Ilustración 2*.

Ilustración 2: Maximización del margen del hiperplano.



FUENTE: TORGO [26].

[1] Hiperplano en geometría es una extensión del concepto de plano.

Cuando las clases no son linealmente separables, este problema necesita ser resuelto en un espacio dimensional muy alto. Por los altos costos computacionales de trasladarse a un espacio de alta dimensionalidad, se utiliza el truco del núcleo (kernel por su nombre en inglés) [26]. A continuación, se encuentran las funciones más comunes utilizadas como kernel:

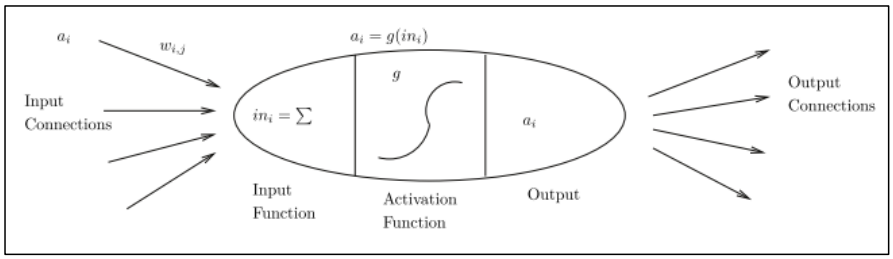
- ✓ Núcleo Gaussiano: $K(x_i, x_j) = e^{(-\frac{\|x_i - x_j\|^2}{2\sigma^2})}$
- ✓ Núcleo polinomial con *d* grados: $K(x_i, x_j) = (x_i \cdot x_j)^2$
- ✓ Núcleo radial: $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$

Esta técnica es altamente costosa desde el punto de vista computacional, ya que requiere realizar grandes operaciones matriciales, así como cálculos numéricos. Tanto para el proceso de entrenamiento del clasificador como el de prueba requieren mucho tiempo y un uso grande de memoria [9].

2.1.3.1.3. Técnica basada en redes neuronales artificiales.

Las técnicas de clasificación basadas en redes neuronales artificiales son modelos no lineales que pueden ser usados para la tarea de clasificación. Una red de neuronas artificiales está compuesta de un conjunto de unidades de procesamiento (neuronas) que están conectadas entre sí, con cada de estas conexiones tienen un peso asociado (conexión ponderada). Cada de estas unidades tiene un nivel de activación y realiza un cálculo lineal que provee medios para actualizar este nivel de activación [26].

Ilustración 3: Representación gráfica de una neurona artificial.



FUENTE: TORGO [26].

El cálculo lineal de la función de entrada es esencialmente una suma ponderada de las entradas a la neurona, donde k es el número de neuronas entrantes a la neurona i, $w_{x,y}$ es el peso de la conexión entre las neuronas x y la neurona y, y a_r es la salida de la neurona r [26].

$$in_i = \sum_{j=1}^{k} [(w_{j,y})(a_j)]$$

Ecuación 3: Función de entrada de una neurona artificial.

El cálculo no lineal en la neurona i es aplicado al valor de in_i. Ésta consiste en una función de activación que podría determinar la salida de la neurona i. Según la función de activación es el comportamiento de la red de neuronas [26]. Entre las funciones de activación más comunes tenemos a:

- ✓ Función de Step: $step(x) = \begin{cases} 1 \; si \; x \geq t \\ 0 \; si \; x < t \end{cases}$
- ✓ Función de Sign: $sing(x) = \begin{cases} +1 \; si \; x \geq 0 \\ -1 \; si \; x < 0 \end{cases}$
- ✓ Función de Sigmoid: $sigmoid(x) = \frac{1}{1 + exp^{-x}}$

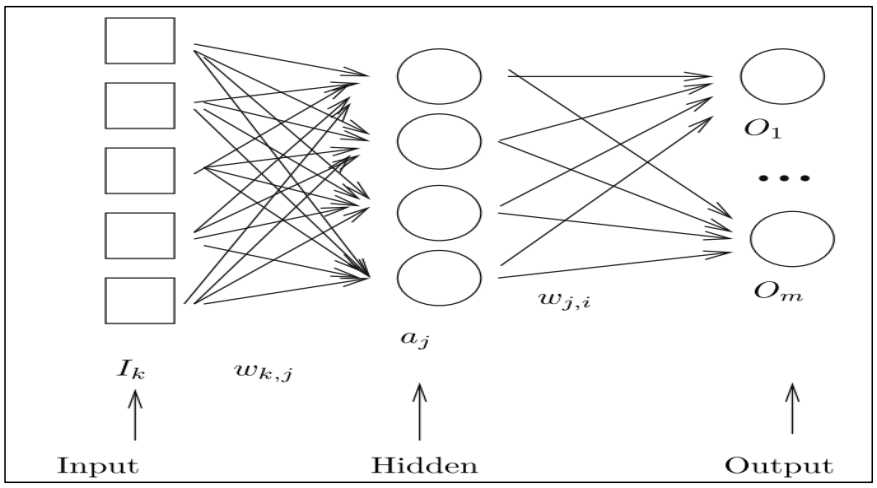
Existen diferentes clases de algoritmos que utilizan este método como medio de aprendizaje, utilizando una estructura simple de una sola capa, o múltiples capas [27]. Las redes neuronales de una sola capa son un conjunto de neuronas que están conectados unas con

otras [26]. En cambio con una red de multicapas están estructuradas por sucesivas capas que contienen unidades de procesamiento, con conexiones que corren desde todas las unidades de una capa a todas la unidades de la capa siguiente, pero sin otras conexiones permitidas [27].

Una red neuronal multicapa puede ser según la dirección de estas conexiones, de dos tipos: feed-forward (por su nombre en inglés) y recurrentes. La primera consiste en conexiones unidireccionales entre las neuronas desde las entradas hasta las neuronas de salida (conexiones no cíclicas). Las de tipo recurrente en cambio podrían permitir conexiones arbitrarias entre neuronas (conexiones cíclicas) [26].

Dentro de una red neuronal multicapa existen tres tipos de capas: la capa de entrada, la capa oculta y la capa de salida. La capa de entrada está compuesta por tantas neuronas como atributos predictores tenga el problema. La capa de salida en el caso de un problema de regresión estaría conformada por una sola neurona y en el caso de una clasificación tendría tantas neuronas como clases. Entre estas dos capas anteriores esta la capa oculta, la arquitectura más común es la de una sola capa oculta [26]. En la *Ilustración 4* se muestra una arquitectura de una red multicapa no cíclica (feed-forward).

Ilustración 4: Arquitectura de una red neuronal multicapa no cíclica.

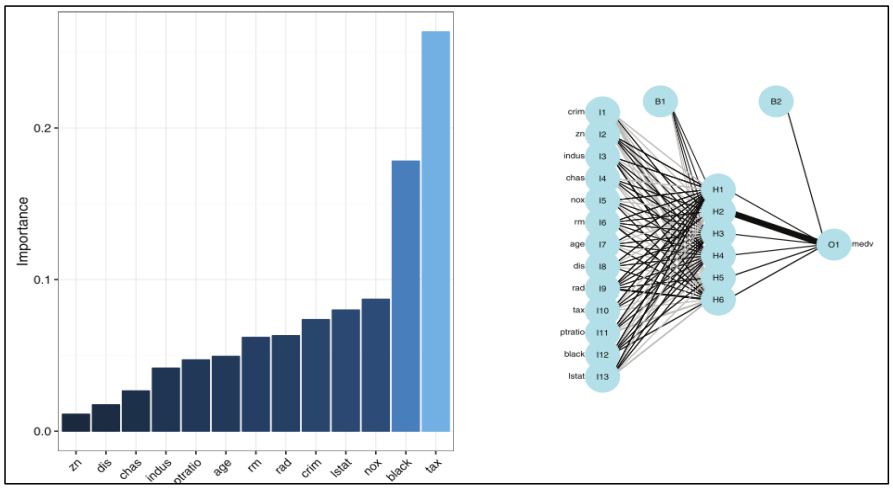


FUENTE: TORGO [26].

El entrenamiento de un clasificador de redes neuronales feed-forward es de forma interactiva unidireccional desde la capa de entrada hasta la capa de salida. Cuando se obtiene un valor de salida de las neuronas de salida, este valor es comparado con el valor real del atributo objetivo del conjunto de entrenamiento. Si hay un error, es decir no converge[2], este error es utilizado para actualizar los pesos de las conexiones de la red utilizando una regla, todo este proceso iterativo sigue hasta que los pesos de la red convergen o algún otro criterio de terminación es alcanzado. El proceso de actualización de los pesos consiste en intentar propagar el error desde las neuronas de salida a las neuronas en capas anteriores. Esto se logra generalmente mediante la aplicación de un algoritmo conocido como backpropagation [26].

Los modelos obtenidos por una red neuronal artificial son considerados modelos black-box en el sentido que los seres humanos no podemos comprender [3, p. 406] [26] y experimentalmente se ha demostrado que son sensibles al ruido [11, p. 331]. Para tratar de solucionar este problema de comprensión de los modelos existen herramientas que intentan explorar tales modelos, como ejemplo tenemos a la *Ilustración 5* que es resultado del uso de las herramientas del paquete Neural-NetTools del software estadístico de R.

Ilustración 5: Representación gráfica del resultado de una red neuronal artificial.



FUENTE: TORGO [26].

[2] Se dice que un algoritmo iterativo converge cuando, a medida que las iteraciones avanzan, la salida se acerca cada vez más a un valor específico.

2.1.3.1.4. Métodos de modelación no paramétrico.

Multivariate Adaptive Regression Splines (MARS) es un método de modelación no paramétrico que extiende el modelo lineal incorporando no linealidades e interacciones de variables. Es una generalización de la Recursive Partitioning Regression (RPR), que divide el espacio de las variables predictoras en diferentes subregiones [28].

MARS siendo modelo de regresión aditiva, puede descomponer una función compleja de una manera aditiva tal que cada término tenga una forma más simple [26], en este caso funciones lineales básicas de la forma:

$$(x-t)_+ = \begin{cases} x-t, & si\ x > t, \\ 0, & caso\ contrario, \end{cases} \quad y \quad (t-x)_+ = \begin{cases} t-x, & si\ x < t, \\ 0, & caso\ contrario. \end{cases}$$

Ecuación 4: Funciones lineales básicas usadas por MARS.

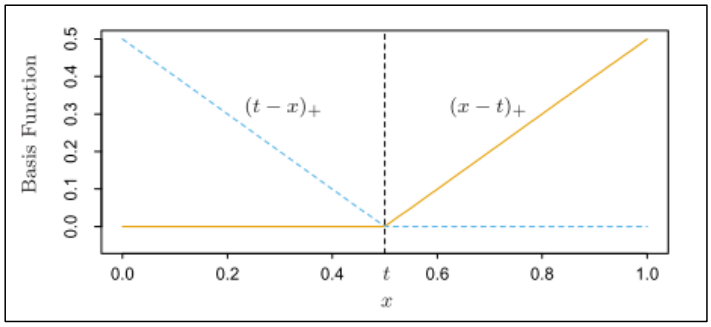
Los modelos aditivos son generalmente considerados muy interpretables, ya que se puede entender fácilmente la contribución de cada término hacia las predicciones del modelo. Los modelos de MARS tienen generalmente la siguiente forma [26]:

$$mars(x) = c_0 + \sum_{i=1}^{k} c_i \times B_i(x)$$

Ecuación 5: Forma general de un modelo construido por MARS.

Donde c_i son los parámetros de los modelos para las respectivas variables que van desde i $=$ 1, ... , k. El valor c_0 representan al intercepto, B_i son las funciones básicas y k que representa el número de explicativas, que incluye interacciones de las variables predictoras [28]. Las funciones básicas son usualmente seleccionadas de una de las siguientes formas: (i) la constante 1 (para el intercepto); (ii) una función de bisagra con la forma max $(0, X - c)$ o max $(0, c - X)$, donde c es un valor umbral; o (iii) un producto de dos o más funciones de bisagra, que intenta capturar las interacciones entre dos o más variables [26]. En la *Ilustración 6* se muestra dos funciones de bisagras.

Ilustración 6: Funciones básicas $(x-0.5)_+$ *(línea anaranjada) y* $(0.5-x)_+$ *(línea entrecortada azul) usada por MARS.*



FUENTE: HASTIE, TIBSHIRANI Y FRIEDMAN [29].

Los puntos de partición del espacio y los parámetros de los modelos se obtienen a partir de los datos analizados. El número de funciones base resultantes indica la complejidad del modelo. MARS genera puntos de corte para las diferentes variables los cuales son identificados a través de las funciones base, las que indican el inicio y el término de una región. En cada región en que se va dividiendo el espacio se ajusta una función base de una variable, la cual es lineal. El modelo final se constituye como una combinación de las funciones base generadas. Para determinar estos puntos de corte usa un algoritmo forward/backward stepwise por etapas. Primero, mediante el algoritmo forward stepwise se genera un modelo sobreestimado con un gran número de funciones base; posteriormente, mediante el algoritmo backward stepwise, se eliminan los nodos que menos contribuyen al ajuste global. El algoritmo se detiene cuando la aproximación construida incluye un número máximo de funciones fijadas por el investigador [28].

2.1.3.2.Desbalance de clases

Las clases en un conjunto de datos están bien balanceadas cuando todas las clases están representadas con la misma proporción, pero en la práctica esto sucede muy raramente, y en el dominio de la tarea de clasificación están caracterizados por la tener una pequeña proporción de la clase positiva y una larga proporción de la clase negativa. Siendo la clase positiva generalmente el punto de interés del estudio. Este problema es comúnmente conocido como problema de desbalanceo de clases [30].

Usualmente el interés es clasificar las instancias positivas, y es donde el clasificador falla porque caen en la clase mayoritaria (clase negativa). Varios son los métodos para tratar con datos desbalanceados, entre esos tenemos los métodos a nivel de datos que se usan generalmente en el preprocesamiento y que usualmente utilizan diversas formas de re-muestreo [30].

Entre estos métodos que utilizan las formas de re-muestreo tenemos principalmente al sub-muestreo y al sobre-muestreo. El método del sub-muestreo aleatorio equilibra las distribuciones de clase descartando, al azar, instancias de la clase mayoritaria. Y el método de sobre-muestreo aleatorio es un método ingenuo, que equilibra distribuciones de clase por replicación, al azar, de las instancias de la clase minoritaria [30].

2.1.3.3.División de datos.

Cuando se tiene un conjunto de datos para realizar el entrenamiento de un modelo y otro conjunto de datos para la evaluación del desempeño de tal modelo, no se necesita realizar una división de datos. Pero en la mayoría de las veces se necesita realizar una división del único conjunto de datos que se tiene a disposición. Para realizar esta división existen estrategias, las cuales cuentan con algoritmos o métodos [30]. A continuación, se describen los más comunes.

2.1.3.3.1. Holdout.

Es uno de los métodos más simples que toma el conjunto de datos original y lo divide aleatoriamente en dos conjuntos. La práctica común es utilizar un tercio para las pruebas y el resto para el entrenamiento. Este método es ineficiente en el uso de los datos. Por ejemplo, en un problema de clasificación, una o más clases pueden faltar en uno de los subconjuntos, lo que conduce a una mala estimación del modelo, así como a su evaluación [30].

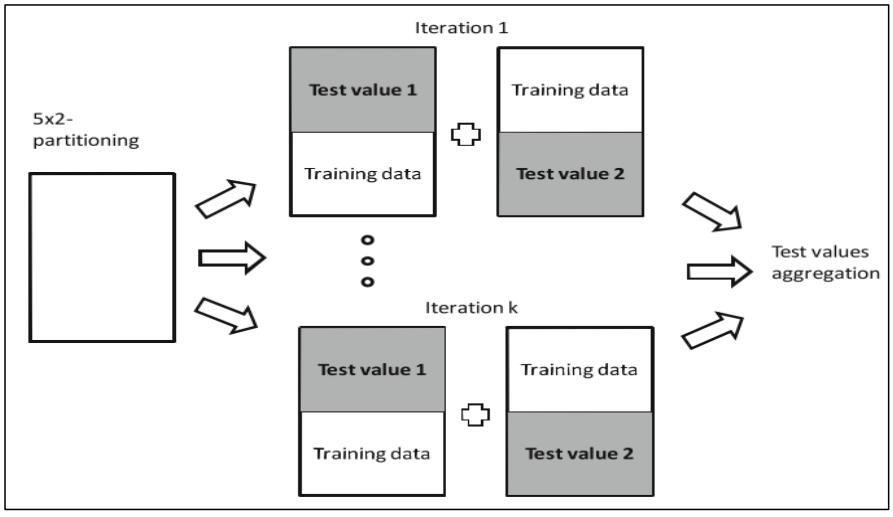
En relación con esto, algunas versiones avanzadas usan la denominada estratificación. El muestreo estratificado es un muestreo probabilístico, donde un conjunto de datos original se divide en grupos no solapados llamados estratos, y las instancias se seleccionan de cada estrato proporcionalmente a la probabilidad apropiada. Asegura que cada clase esté representada con la misma frecuencia en ambos subconjuntos. Pero todavía no impide el

inicio del sesgo en los conjuntos de entrenamiento y pruebas. Para una mejor fiabilidad de la estimación del error, los métodos se repiten y la precisión resultante se calcula como un promedio sobre todas las iteraciones [30].

2.1.3.3.2. Validación cruzada.

Este método divide al conjunto de datos original en k particiones disjuntas del mismo tamaño, donde k es un parámetro del método. En cada una de las k vueltas, se utiliza una partición para la evaluación y las particiones restantes para el aprendizaje del modelo. La precisión resultante es la media de todas las vueltas [30]. Este método es ampliamente utilizado ya que se ha demostrado que estos esquemas se aproximan continuamente a un valor estable, lo que permite comparaciones realistas entre el clasificador [18]. Uno de los esquemas que evita el solapamiento entre los conjunto de datos y muestra una mayor independencia es el esquema con un valor de $k = 2$ y de un numero de 5 repeticiones [31], a continuación la del proceso de validación cruzada de 2 particiones con 5 repeticiones, comúnmente conocido como 5x2CV.

Ilustración 7: Validación cruzada 5x2CV.



FUENTE: GARCÍA, LUENGO Y HERRERA [18].

2.1.3.4.Métricas de evaluación de un clasificador.

Existen métricas (medidas) para evaluar cuán bueno o cuán "exacto" es un clasificador en la predicción de la etiqueta de clase de las tuplas (instancias), esta sección es basada en el apartado 8.5.1 de [3].

En la clasificación existen los términos de tuplas positivas (tuplas de la clase principal de interés) y tuplas negativas (todas las demás tuplas). Dadas dos clases, por ejemplo, las tuplas positivas pueden ser “compra computadora = si” mientras que las tuplas negativas son “compra computadora = no” [3]. Cuando se usa un clasificador en un conjunto de prueba de tuplas etiquetadas surgen los términos:

- ✓ **Tuplas positivas (P):** Se refiere al número de tuplas positivas.
- ✓ **Tuplas negativas (N):** Es el número de tuplas negativas.

Para cada tupla, comparamos la predicción de la etiqueta de clase del clasificador con la etiqueta de clase conocida de la tupla, surgiendo cuatro términos adicionales que necesitamos saber que son los "elementos básicos" utilizados en el cálculo de muchas medidas de evaluación [3]:

- ✓ **Verdaderos positivos (TP):** Se refieren a las tuplas positivas que fueron correctamente etiquetadas por el clasificador. Sea TP el número de verdaderos positivos [3].
- ✓ **Verdaderos negativos (TN):** Éstos son las tuplas negativas que fueron correctamente etiquetadas por el clasificador. Sea TN el número de verdaderos negativos [3].
- ✓ **Falsos positivos (FP):** Estas son las tuplas negativas que fueron etiquetadas incorrectamente como positivas (por ejemplo, las tuplas de clase compran computadoras = no para las cuales el clasificador predecía compran computadoras = si) [3]. Sea FP el número de falsos positivos.

- ✓ **Falsos negativos (FN):** Éstas son las tuplas positivas que fueron mal etiquetadas como negativas (por ejemplo, las tuplas de clase compran_computadoras = si, para las cuales el clasificador predijo compran_computadoras = no) [3]. Sea FN el número de falsos negativos.

Aunque la exactitud es una medida específica, la palabra "exactitud" también se utiliza como un término general para referirse a las capacidades predictivas de un clasificador [3]. Las medidas de evaluación más utilizadas son:

- ✓ **Tasa de error:** También llamada tasa de clasificación errónea, que es simplemente 1 – exactitud [3]. Esto también puede ser calculado como:

$$\textbf{Tasa de error} = \frac{\boldsymbol{FP + FN}}{\boldsymbol{P + N}}$$

Ecuación 6: Fórmula de la tasa se error.

- ✓ **Sensibilidad:** Esta también se denomina tasa de positivos verdaderos (TPR) y es la proporción de tuplas positivas que se identifican correctamente [3].

$$\textbf{Sensibilidad} = \textbf{TPR} = \frac{\boldsymbol{TP}}{\boldsymbol{P}}$$

Ecuación 7: Fórmula de la sensibilidad.

- ✓ **Tasa de alarma falsa:** Esta también se denomina tasa de positivos falsos (FPR) es decir, la proporción de tuplas negativas que son erróneamente identificadas como positivas [19, p. 179].

$$\textbf{Tasa de alarma falsa} = \textbf{FPR} = \frac{\boldsymbol{FP}}{\boldsymbol{N}}$$

Ecuación 8: Fórmula de la tasa de alarma falsa.

- ✓ **Especificidad:** Es la tasa de negativos verdaderos (es decir, la proporción de tuplas negativas que se identifican correctamente) [3].

$$\textbf{Especificidad} = \frac{\boldsymbol{TN}}{\boldsymbol{N}}$$

Ecuación 9: Fórmula de la especifidad.

- ✓ **Precisión:** puede considerarse como una medida de la exactitud (es decir, qué porcentaje de tuplas etiquetadas como positivas lo son realmente) [3].

$$\textbf{Precisión} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}}$$

Ecuación 10: Fórmula de la precisión.

Además de las medidas basadas en la exactitud, a los clasificadores también pueden evaluarse con respeto a los siguientes aspectos [3]:

- ✓ **Velocidad**: Se refiere a los costos computacionales involucrados en la generación y uso del clasificador dado [3].

- ✓ **Robustez**: Esta es la capacidad del clasificador para hacer predicciones correctas dados datos ruidosos o datos con valores faltantes. La robustez se evalúa normalmente con una serie de conjuntos de datos sintéticos que representan grados cada vez mayores de ruido y valores faltantes [3].

- ✓ **Escalabilidad:** Se refiere a la capacidad de construir el clasificador de manera eficiente dada una gran cantidad de datos. La escalabilidad se evalúa típicamente con una serie de conjuntos de datos de tamaño creciente [3].

- ✓ **Interpretabilidad**: Se refiere al nivel de entendimiento y comprensión que proporciona el clasificador o predictor. La interpretabilidad es subjetiva y por lo tanto más difícil evaluar. Los árboles de decisión y las reglas de clasificación pueden ser fáciles de interpretar, la interpretabilidad de los modelos puede disminuir cuanto más complejos se vuelven [3].

2.1.3.5.Comparación de clasificadores.

Cuando se tienen varios clasificadores y se necesita seleccionar el mejor, es necesario escoger una medida para estimar el desempeño del clasificador y simplemente escoger uno que tenga el desempeño más alto en esa medida [30]. A continuación se expondrán herramientas y análisis que se utilizan para la comparación de clasificadores.

- ✓ **Matriz de confusión**

Las más básicas de las medidas de rendimiento pueden describirse en una matriz de confusión o tabla de contingencia. La matriz de confusión contiene información sobre clasificaciones actuales y previstas por un sistema de clasificación. Las columnas representan las clasificaciones previstas y las filas representan las clasificaciones reales (verdaderas) para cada uno de las instancias [32]. En la *Ilustración 8* se puede ver una matriz de confusión empleada para comparar clasificadores.

Ilustración 8: Matrices de confusión de los clasificadores de MLPNN y C5.0 para subconjuntos de entrenamiento y de prueba.

Model	Training Data			Testing Data		
	Desired output	Yes	No	Desired output	Yes	No
MLPNN	Yes	TP=1,913	FP=1,799	Yes	TP=769	FP=808
	No	FN=1,093	TN=26,784	No	FN=510	TN=11,535
C5.0	Yes	2,258	1,454	Yes	740	837
	No	684	27,193	No	513	11,532

FUENTE: ELSALAMONY y ELSAYAD [14].

- ✓ **Análisis ROC**

En análisis ROC (Características de Funcionamiento del Receptor) se emplea una gráfica de dos dimensiones llamado la gráfica ROC, donde TPR es graficada en el eje Y y FPR en el eje X, la tasa de verdaderos positivos (TPR) se representa frente a la tasa de falsos positivos (FPR) [3, p. 870].

Los clasificadores discretos son los que su salida da solo una etiqueta de clase. Cada clasificador discreto proporciona un par de valores para (FPR, TPR), los que corresponden a un solo punto en el espacio ROC. Estos puntos en el espacio ROC pueden servir para comparar, visualizar y seleccionar clasificadores en base a su desempeño [33].

Para otros clasificadores, como un clasificador bayesiano o una Red neuronal artificial, la salida son valores de probabilidad que representan hasta qué punto una instancia pertenece a una de las dos clases. Para estos métodos se debe fijar un valor umbral de decisión que determinará un punto en el espacio ROC. Por ejemplo, si ante una determinada magnitud fijamos ese umbral en 0.8, la probabilidad de las instancias iguales o superiores serán predichas como positivas, y los valores por debajo serán predichos como negativos. Por tanto podremos calcular una tabla de contingencia (o matriz de confusión) para ese umbral de 0.8, y encontrar el punto correspondiente en el espacio ROC [34].

La curva ROC es la curva interpolada de puntos cuyas coordenadas son funciones del umbral:

$$\boldsymbol{Umbral\ de\ decisión} = \boldsymbol{\theta} \in \mathbb{R}, \boldsymbol{donde}\ \boldsymbol{\theta} \in [\mathbf{0}, \mathbf{1}]$$

$$\mathbf{ROC}_{\boldsymbol{x}}(\boldsymbol{\theta}) = \mathbf{FPR}(\boldsymbol{\theta}) = \frac{\boldsymbol{FP}(\boldsymbol{\theta})}{\boldsymbol{FP}(\boldsymbol{\theta}) + \boldsymbol{TN}(\boldsymbol{\theta})} = \frac{\boldsymbol{FP}(\boldsymbol{\theta})}{\boldsymbol{N}}$$

Ecuación 11: Fórmula del punto en x de la curva ROC

$$\mathbf{ROC}_{\boldsymbol{y}}(\boldsymbol{\theta}) = \mathbf{TPR}(\boldsymbol{\theta}) = \frac{\boldsymbol{TP}(\boldsymbol{\theta})}{\boldsymbol{FN}(\boldsymbol{\theta}) + \boldsymbol{TP}(\boldsymbol{\theta})} = \frac{\boldsymbol{FP}(\boldsymbol{\theta})}{\boldsymbol{P}} = \mathbf{1} - \frac{\boldsymbol{FN}(\boldsymbol{\theta})}{\boldsymbol{P}} = \mathbf{1} - \boldsymbol{FNR}(\boldsymbol{\theta})$$

Ecuación 12: Fórmula del punto en y de la curva ROC

- ✓ **Área bajo la curva de ROC (AUC)**

Es una porción del área de la curva de ROC, que va desde 0 a 1.0. Sin embargo, debido a que la suposición aleatoria produce la línea diagonal entre (0,0) y (1, 1), que tiene una área de 0.5, ningún clasificador realista debe tener un AUC inferior a 0.5. El AUC tiene

una propiedad estadística importante: el AUC de un clasificador es equivalente a la probabilidad de que el clasificador clasifique una instancia positiva elegida al azar es más alta que una instancia negativa sea elegida al azar. Esto es equivalente a la prueba de posición de Wilcoxon. AUC también está estrechamente relacionado con el coeficiente de Gini que es el doble del área entre la diagonal y la curva ROC, y señalan que $Gini + 1 = 2 * AUC$ [33].

AUC es un mejor método para comparar clasificadores. Un algoritmo que tiene una alta precisión puede no ser suficiente porque puede haber un alto número de falsos positivos en los resultados. Se ha demostrado que AUC es estadísticamente consistente y más discriminante que la precisión [35].

2.1.3.6.Comparación estadística de clasificadores.

Aunque es posible establecer la superioridad de una técnica de clasificación sobre las otras mediante las medidas de evaluación de sus clasificadores, su uso es desaprobado, por ello se disponen de pruebas estadísticas para obtener conclusiones significativas y duraderas [18].

Cuando se necesita demostrar la mejora en el rendimiento del modelo, especialmente si queremos demostrar que un modelo realmente supera a otro en una tarea de aprendizaje en particular, se tiene que utilizar una prueba de significación estadística[3] y verificar la hipótesis de la mejora del rendimiento [30], [36].

Estas pruebas pueden ser paramétricas o no paramétricas, pero es recomendable las pruebas no paramétricas para la comparación de algoritmos [18]. Las pruebas no paramétricas empleadas para la comparación de técnicas de minería de datos son usadas dependiendo del número de muestras (conjunto de datos) y técnicas a comparar.

- ✓ **Prueba de Wilcoxon.**

Esta prueba no paramétrica se utiliza cuando se compara dos algoritmos sobre múltiples conjunto de datos, el uso correcto de esta prueba es cuando estamos interesados en encontrar

[3] Demuestra que existe una diferencia estadísticamente significativa, lo cual solamente significa que hay evidencias estadísticas de que hay una diferencia.

la diferencia de dos métodos, pero no cuando estamos interesados en comparar múltiples algoritmos [18].

- ✓ **Prueba de Friedman**

También conocido como "Análisis bidireccional de las varianzas por rangos de Friedman". La prueba de Friedman es el análogo de la prueba de medidas repetidas ANOVA en procedimientos estadísticos no paramétricos; por lo tanto, es la prueba de comparaciones múltiple que apunta a detectar diferencias significativas entre el comportamiento de dos o más algoritmos [18]. Se calcula con la *Ecuación 13* [37], donde D ($D \geq 2$) es el número de conjuntos de datos utilizados en el estudio, K ($K \geq 2$) es el número total de clasificadores construidos por los algoritmos y r_j^i es la posición del clasificador j con relación a los otros clasificadores sobre el conjunto de datos i.

Friedman posiciona los algoritmos para cada conjunto de datos por separado, el algoritmo de mejor rendimiento obtiene la posición 1, el segundo mejor la posición 2...; para después comparar las posiciones promedios de los algoritmos $R_j = \frac{1}{D}\sum_{i=1}^{D} r_j^i$. Bajo la hipótesis nula h_0, que establece que todos los algoritmos son equivalentes y por lo tanto sus rangos R_j deben ser iguales, la estadística de Friedman. El valor de X_F^2 se distribuye de acuerdo a la distribución de Chi-cuadrado con $K - 1$ grados de libertad [38].

$$X_F^2 = \frac{12D}{K(K+1)}\left[\sum_{j=1}^{K} R_j^2 - \frac{K(K+1)^2}{4}\right]$$

Ecuación 13: Prueba estadística de Friedman.

Para evaluar la hipótesis nula h_0 se compara el valor de p con el $nivel\ de\ significancia$ elegido (generalmente 0.05 o 0.1). Un nivel de significancia de 0,01 indica un riesgo del 1% de concluir que existe una diferencia cuando no hay diferencia real. Si el valor de p es menor al $nivel\ de\ significancia$ ($p < nivel\ de\ significancia$) entonces se rechaza la hipótesis nula. En este caso, se procede a aplicar una prueba post hoc para detectar qué que par de clasificadores difieren significativamente [38].

La prueba post hoc Friedman-Nemenyi sirve para reportar cualquier diferencia significativa individual entre todos los pares de clasificadores, esto se da si sus posiciones medias difieren al menos por la diferencia crítica (CD), la cual se calcula con la *Ecuación 14*, donde el valor de $q_{\propto,\infty,K}$ está basado en la estadística del rango studenizado[4] [37].

$$CD = q_{\propto,\infty,K}\sqrt{\frac{K(K+1)}{6D}}$$

Ecuación 14: Prueba estadística post hoc de Friedman-Nemenyi.

Cuando múltiples clasificadores son comparados, los resultados de los test de post-hoc pueden ser representados en un diagrama simple llamado diagrama de diferencias críticas. En el eje x es graficado una línea donde se ubica la posición promedio de los algoritmos. El algoritmo con el valor más bajo es el mejor del grupo. Al comparar todos los algoritmos entre sí, se conectan los grupos de algoritmos que no son significativamente diferentes. En este diagrama también se muestra la diferencia critica [38].

2.1.4. Limpieza de datos.

Los conjuntos de datos recolectados en una organización o industria pueden tener datos incompletos también llamados valores faltantes, la presencia de éstos se asocia a tres tipos de problemas: 1) pérdida de eficiencia; 2) complicaciones en el manejo y análisis de los datos; Y 3) sesgo resultante de las diferencias entre los datos faltantes y completos [39].

La falta de datos es un problema común, las tasas de menos del 1% de valores faltantes se consideran generalmente triviales, 1-5% manejables. Sin embargo, el 5-15% requiere métodos sofisticados para manejar, y más del 15% puede afectar severamente cualquier tipo de interpretación [40, p. 1].

Existen modos diferentes que conducen a la introducción de valores faltantes que podrían ser: 1) Faltantes completamente aleatorios (MCAR); 2) Faltantes aleatorios (MAR); Y 3) Faltantes no aleatorios (NMAR) [41].

[4] Rango studenizado es la diferencia entre los datos más grandes y más pequeños de una muestra medidos en unidades de la desviaciones estándar de la muestra [61].

Por lo general, el tratamiento de valores faltantes en un proceso de minería de datos se puede manejar de tres maneras diferentes [18]:

- La primera aproximación es descartar las instancias con valores faltantes en sus atributos. Por lo tanto, la supresión de atributos con niveles elevados de valores faltante se incluye también en esta categoría [18].

- Otro enfoque es el uso de procedimientos de máxima verosimilitud, donde se calculan los parámetros de un modelo para la porción completa de los datos y se usan posteriormente para imputación por medio de muestreo [18].

- Finalmente, la imputación de valores faltantes son una clase de procedimientos que apunta a llenar los valores faltantes con valore estimados [18].

La eliminación de valores faltantes se refiere a descartar las instancias con valores faltantes o eliminar atributos que tienen entradas ausentes. Este último sólo se puede aplicar cuando los atributos eliminados no son necesarios para realizar el análisis de datos. Tanto la eliminación de instancias como de los atributos reducen el contenido informativo de los datos [39].

Los métodos de imputación tienen una ventaja fundamental, la cual es que el tratamiento de valores faltantes es independiente del algoritmo de aprendizaje utilizado. Estos métodos se pueden dividir en tres categorías: 1) basado en datos; 2) basado en modelos; Y 3) basado en Machine Learning (ML). Los métodos basados en ML utilizan todos los datos disponibles y consideran algún algoritmo de ML para realizar la imputación [39].

Existen métodos de imputación simple y múltiple, dentro de los métodos de imputación simple basados en Machine Learning se encuentran los que utilizan los algoritmos de K-Vecinos más Cercanos, árboles de decisión C4.5 y el algoritmo Naïve Bayes (NB). Se ha comparado el desempeño de es estos tres algoritmos antes mencionados, en la tarea de imputar, dando como resultado que la imputación realizada con el algoritmo NB es de mayor exactitud que los otros algoritmos empleados en la comparación [41].

2.1.4.1.Imputación utilizando el algoritmo Naïve Bayes.

La utilización de un clasificador de Naïve Bayes para el proceso de imputar valores faltantes se basa en que este algoritmo utiliza la probabilidad para representar cada clase y tiende a encontrar la clase más posible para cada muestra. Su insensibilidad a la cantidad de valores faltantes, buen rendimiento, forma simple y alta velocidad de cálculo; lo hacen un gran candidato para la tarea de imputar [41].

La imputación por medio del algoritmo de clasificación NB (NBI), se la realiza definiendo el atributo a ser imputado, este atributo se lo utiliza como atributo clase para la construcción del clasificador. La mayoría de conjunto de datos tienen más de un atributo con valores faltantes, por lo que en estos casos se procede primero a identificar los atributos con valores faltantes y después ordenarlos en forma a su relevancia (si existe) [41].

2.1.5. Sistema recomendador.

El objetivo de un sistema recomendador (SR) es generar recomendaciones significativas para un conjunto de usuarios para artículos o productos que podrían ser de interés para ellos. Sugestiones para los libros en Amazon, o películas en Netflix, son ejemplos del mundo real de la operación de la fuerza en la industria de SR. El diseño de los motores de esa recomendación depende del dominio y las características particulares de los datos disponibles [42].

2.1.5.1.Técnicas para realizar recomendaciones

Los SR utilizan principalmente dos técnicas: Basada en contenido y basada en filtrado colaborativo (FC). FC es una técnica de predicción para contenido que no pueden ser descritos fácil y adecuadamente por metadatos como películas y música. La técnica de filtrado colaborativo funciona construyendo una base de datos (matriz de elementos del usuario) de las preferencias de los usuarios hacia los elementos [42]. Dentro de la técnica de FC se pueden dividir dos categorías: basadas en memoria y basadas en modelos.

2.1.5.2.Técnicas basadas en modelos

Esta técnica generalmente emplea las calificaciones anteriores para aprender un modelo con el fin de mejorar el rendimiento de la técnica de FC. El proceso de construcción del modelo puede usar machine learning o data mining (DM). Estas técnicas pueden recomendar rápidamente un conjunto de elementos por el hecho de que utilizan un modelo pre-calculado y han demostrado producir resultados de recomendación que son similares a las técnicas de recomendación basadas en vecinos más cercanos. Algunas de las técnicas basadas en modelos analizan la matriz de elementos de usuario para identificar las relaciones entre los elementos; estos usan estas relaciones para comparar la lista de recomendaciones top-N. La técnica basada en modelos resuelve el problema de escasez (sparsity) asociado con los SR [42].

2.1.6. Marketing directo.

Existen dos enfoques principales para que las empresas promuevan productos y / o servicios, a través de campañas de masas dirigidas a un público general indiscriminado o al marketing dirigido, dirigidas a un conjunto específico de contactos [43].

2.1.7. Metodologías para proyectos de minería de datos.

Diversas son las metodologías para poder realizar un proyecto de minería de datos. Por lo cual se han comparado las metodologías de SEMMA, KDD y CRISP-DM [44], y como resultado se determinó que CRISP-DM tiene cada una de sus fases claramente estructuradas, definiendo de tal forma las actividades y tareas que se requieren para lograr el objetivo planteado es decir es la más completa entre las metodologías comparadas, es flexible por ende se puede hacer uso en cualquier herramienta de minería de datos [44].

2.1.7.1.Metodología CRISP-DM.

La metodología CRISP-DM se describe en términos de un modelo de proceso jerárquico, que comprende cuatro niveles de abstracción (de general a específico): fases, tareas genéricas, tareas especializadas e instancias de proceso [45].

CRISP- DM tiene como objetivo hacer de grandes proyectos de minería de datos, menos costosos, más confiables, más repetibles, más administrables y más rápidos [22, p. 30].

2.1.7.1.1. Fases de CRISP-DM.

La descripción de fases y tareas como pasos discretos, realizados en un orden específico representa una secuencia idealizada de eventos. En la práctica, muchas de las tareas se pueden realizar en un orden diferente y con frecuencia será necesario retroceder a las tareas anteriores y repetir ciertas acciones [22]. Las fases que componen a la metodología de CRISP-DM son:

- ✓ **Comprensión del negocio**: Esta fase se centra en la comprensión de los objetivos del proyecto y los requisitos desde una perspectiva empresarial, convirtiendo luego este conocimiento en la definición del problema de DM y un plan preliminar diseñado para lograr los objetivos [22].

- ✓ **Comprensión de los datos**: La fase de comprensión de los datos comienza con una primera recopilación de datos y procede con actividades para familiarizarse con los datos, identificar los problemas de calidad en los datos, para descubrir primeros conocimientos sobre los datos o para detectar subconjuntos interesantes para formar hipótesis de información oculta [22].

- ✓ **Preparación de datos**: La fase de preparación de datos abarca todas las actividades para construir el conjunto de datos final a partir de los datos iniciales sin procesar. Las tareas de preparación de datos se realizará repetidamente y no en cualquier orden prescrita [22].

- ✓ **Modelado**: En esta fase, se seleccionan y aplican varias técnicas de modelado y sus parámetros se calibran a valores óptimos. Típicamente, existen varias técnicas para el mismo tipo de problema de DM. Algunas técnicas tienen requisitos específicos en la forma de los datos. Por lo tanto, a menudo es necesario retroceder a la fase de preparación de datos [22].

- ✓ **Evaluación**: Cuáles son, desde una perspectiva de análisis de datos, modelos aparentemente de alta calidad que han sido construidos hasta esta etapa del proyecto. Antes de continuar con el desarrollo del modelo final, es importante evaluar el modelo más exhaustivamente y revisar el las medidas adoptadas en su construcción para asegurarse de que cumple adecuadamente los objetivos del negocio. Al final de esta fase, se debe tomar una decisión sobre cómo usar los resultados de la DM [22].

- ✓ **Implementación**: La construcción del modelo no es generalmente el final del proyecto. Incluso si el propósito del modelo es aumentar el conocimiento de los datos, los conocimientos necesitan ser organizados y presentados de manera que el cliente pueda usarlo. Dependiendo de los requisitos, la fase de despliegue puede ser tan simple como generar un informe o tan compleja como implementar un proceso de minería de datos repetible. En muchos casos será el usuario, no el analista de datos, quien llevará a cabo las etapas de despliegue [22].

2.1.8. Herramientas de software para la minería de datos.

La minería de datos es usada por muchas compañías multinacionales, medianas y pequeñas empresas; para lo cual en su mayoría utilizan herramientas de minería de datos ya sean software propietario o de software libre [46].

2.1.8.1.RapidMiner.

RapidMiner es sin duda el sistema líder mundial de código abierto para la minería de datos. Está disponible como una aplicación independiente para el análisis de datos y como un motor de minería de datos para la integración en productos propios [46]. Se usa en más de 150 países, cuenta con la versión gratuita y versiones de pago. La versión gratuita tiene las limitaciones en cuanto al número de instancias a analizar y también está limitado por el uso de tan solo un procesador lógico.

2.1.8.2.Weka.

Weka es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o llamados desde su propio código Java. Weka contiene herramientas para pre-procesamiento de datos, clasificación, regresión, agrupamiento, reglas de asociación y visualización. También es adecuado para desarrollar nuevos esquemas de aprendizaje de máquinas, es sin duda el sistema líder mundial de código abierto para la minería de datos [46].

2.1.8.3.KNIME.

KNIME es un espacio de trabajo gráfico fácil de usar para todo el proceso de análisis: acceso a datos, transformación de datos, investigación inicial, análisis predictivo, visualización y generación de informes. La plataforma de integración abierta ofrece más de 1000 módulos (nodos) [46].

2.1.8.4.R Statistic.

R es un lenguaje de programación y un entorno para la informática estadística. Hay versiones de R para las familias de los sistemas operativos de Unix, Windows y MacOS. Además, R se ejecuta en diferentes arquitecturas como Intel, PowerPC, sistemas Alpha y sistemas Spark. R se aprovecha de una creciente comunidad que coopera en su desarrollo debido a su filosofía de código abierto, donde cada componente está disponible para la inspección y/o adaptación. Este hecho le permite comprobar y probar la fiabilidad de cualquier cosa que utilice en R y esta capacidad puede ser crucial en muchos dominios de aplicación críticos. Hay muchos excelentes documentos, libros y sitios que proporcionan información gratuita sobre R [26].

2.1.8.5.Microsoft R Open.

Microsoft R Open (MRO), antes conocido como Revolution R Open (RRO), es la distribución mejorada de R de Microsoft Corporation. La versión actual, Microsoft R Open 3.3.3, se basa en el lenguaje estadístico R-3.3.3 e incluye capacidades adicionales para

mejorar el rendimiento, la reproducibilidad y el soporte de la plataforma [47]. Este software incluye:

- ✓ El lenguaje de código abierto R, el software de estadísticas más utilizado en el mundo [47].

- ✓ Compatibilidad con todos los paquetes, scripts y aplicaciones que funcionan con R-3.3 [47].

- ✓ La instalación de muchos paquetes incluye todos los paquetes de base de R, más un conjunto de paquetes especializados lanzados por Microsoft Corporation para mejorar aún más su experiencia de MRO [47].

- ✓ Soporte para plataformas basadas en Windows y Linux; bibliotecas matemáticas multi-hilos que trae cálculos multi-hilos a R [47].

- ✓ Un repositorio CRAN por defecto de alto rendimiento que proporciona un conjunto coherente y estático de paquetes para todos los usuarios de MRO [47].

- ✓ El paquete de punto de control que facilita compartir código R y replicar resultados utilizando versiones de paquetes R específicas [47].

2.2. Marco referencial.

2.2.1. Minería de datos en la industria bancaria.

Investigaciones de la aplicación de técnicas de minería de datos en la industria bancaria y financiera se han realizado desde muchos años atrás, por ejemplo, Pang y Gong [16] hablan del algoritmo de clasificación C5.0 y su aplicación en la evaluación individual de créditos de los bancos. Trabajan sobre un conjunto de datos de un banco de Alemania, ese conjunto de datos fue obtenido desde el portal de UCI Machine Learning. Ellos luego de establecer el coste de clasificación errónea, seleccionar el grado de poda y analizar el efecto práctico del refuerzo, llegan a la conclusión que el clasificador construido basado en el algoritmo C5.0 es de alta precisión y bajo costo de riesgo en comparación con otro modelo en la práctica.

Wu y Guan [48], en su investigación sobre el valor de la segmentación de clientes del banco nos dice que "con la feroz competencia en el sector financiero el encontrar una forma de analizar eficazmente los datos del cliente se ha convertido en la clave de la gestión de relaciones banco-cliente". Ellos realizan clusters (grupos), segmentando a los clientes de un banco en tres clases: clientes de valor alto, clientes de valor general y clientes de valor bajo. En su trabajo proponen la tarea de clasificar el resultado de segmentación según las clases encontradas. De esta manera podemos evaluar y predecir cualquier cliente, establecer los correspondientes paquetes de negocios y la gestión de los clientes de valor alto, y reducir el costo de la gestión de relaciones con los clientes de los bancos.

2.2.2. Minería de datos y marketing del banco.

Moro, Laureano y Cortez en su investigación [49], aplican minería de datos para el marketing directo del banco a partir de datos reales de 17 campañas de marketing, para la suscripción a depósitos bancarios en un banco de Portugal, este conjunto de datos es posteriormente subido a un repositorio en UCI Machine Learning con el nombre de "Bank Marketing Data Set". Utilizaron la herramienta R para realizar el proceso de minería. Mediante el análisis de la curva ROC, el área bajo la curva Roc (AUC) y el análisis de la área bajo la curva Lift(Alift), ellos comparan los algoritmos basados en Naïve Bayes, Árboles de decisión y Máquinas de soporte vectorial (SVM). Donde encuentran que el mejor clasificador es el

proporcionado por SVM. También se proponen recopilar más datos basados en el cliente con el fin de verificar modelos predictivos de alta calidad sin información de contacto.

A partir de tal estudio Elsalamony y Elsayad [14], en el 2013 realizaron una investigación titulada “Marketing Directo en Bancos Basado en Redes Neuronales”, utilizando el mismo conjunto de datos de Moro [49]. Los autores proponen a la Red Neuronal de Perceptrones Multicapa (MLPNN) como solución ya que es una de las más famosas técnicas de minería de datos, ya que MLPNN se desarrolla y organiza para poder tratar con datos grandes. Dentro del estudio comparan los modelos proporcionados por las técnicas de C5.0 que es una versión mejorada de los algoritmos C4.5 y ID3, ambos basados en la técnica de árboles de decisión; y el algoritmo MLPNN. C5.0 es un algoritmo de uso comercial, por tal motivo los autores utilizaron el software llamado SPSS Clementine. Tres medidas estadísticas utilizaron para evaluar la clasificación; precisión, sensibilidad y especificidad. Dividieron el conjunto de datos en dos partes: una de entrenamiento y otra de pruebas, por la relación de 70%:30% respectivamente. Resultados experimentales mostraron la eficacia de ambos modelos. Sin embargo, C5.0 logro un rendimiento ligeramente mejor que MLPNN.

Moro, Cortez y P. Rita [13] proponen un sistema de soporte de decisiones para la selección previa de clientes para obtener mejores resultados en las campañas por medio de llamadas telefónicas para vender depósitos a largo plazo. Ellos utilizan los algoritmos de regresión logística (LR), árboles de decisión (DT), redes neuronales (NNET) y máquinas vectores de soporte (SVM); para obtener los clasificadores y después los comparan mediante la métrica de área bajo la curva de ROC (AUC) y el área bajo la curva acumulativa (ALIFT). Métricas con las que NNET obtuvo el mejor desempeño entre los cuatro algoritmos comparados.

En el 2014 Elsalamony [15], trata nuevamente el tema del marketing directo en el banco, cuya investigación la título “Análisis de técnicas de minería de datos en marketing directo del banco”, donde realiza una comparación con los algoritmos de las técnicas de redes neuronales (MLPNN), uno de regresión logística (LR), otro de árboles de decisión (C5.0) y uno basado en Naïve Bayes (NB). Resultando como mejor proveedor de clasificadores al algoritmo C5.0, ya que obtuvo los valores más altos en las medidas de precisión de la clasificación, sensibilidad y especificidad.

2.2.3. Recomendación como clasificación.

Recomendar es proporcionar a los usuarios una serie de sugerencias personalizadas sobre un determinado tipo de elementos. Por tal motivo Basu et al [12] pretenden predecir las preferencias del usuario en un sistema de recomendación de películas, aplicando en él, aprendizaje inductivo, para que la recomendación sea capaz de usar tanto la información de calificación como otras formas de información sobre cada película, como las listas de casting, los comentarios o críticas a las películas, por ejemplo. Ellos demuestran que su método de recomendación supera al método de filtrado colaborativo social existente en el dominio de las recomendaciones. Usan un conjunto de datos al cual lo divide en proporción de 90% para entrenamiento y 10% para pruebas, mediante la toma de una muestra aleatoria estratificada y utilizan las medidas de precisión y sensibilidad. Como resultado tienen una precisión de clasificación del 83% para el método que ellos proponen y de un 78% para el método de recomendación mediante el filtrado social existente.

En cambio HengSong et al [50], abordan una solución para la dispersión o escases de las calificaciones de los usuarios. Ellos proponen un algoritmo de recomendación de filtrado colaborativo basado en la clasificación de elementos para volver a producir las calificaciones. Este enfoque clasifica los elementos para predecir las calificaciones de los valores vacantes cuando sea necesario y, a continuación, utiliza el filtrado colaborativo basado en elementos para producir las recomendaciones. Para la medición del desempeño del algoritmo en las recomendaciones emplean la medida estadística de error medio absoluto. Ellos comparan el algoritmo propuesto de filtrado colaborativo basado en clasificación de artículos (ítems) y el filtrado colaborativo tradicional, obteniendo un mejor desempeño de parte del filtrado colaborativo propuesto.

En [25] realizan un marco genérico para el uso de un clasificador binario para la generación de recomendaciones, ellos investigan los diferentes modelos de clasificación estándar para los sistemas de recomendación y comparan dos de los tipos de métodos de transformación de clasificación multi-clase a una clasificación binaria y dos métodos estándar para sistemas de recomendación. Como resultado obtienen que el clasificador SVM mediante el método de transformación de clasificación multi-clase a binaria 1-vs-1 presenta valores superiores de la métrica de recuerdo (recall) que es la medida de evaluación que utilizaron.

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

A continuación, se presenta la localización del desarrollo de la investigación en la Sección 3.1, posteriormente se describe el tipo de investigación a emplear en la Sección 3.2. En la Sección 3.3, se formalizan los métodos y técnicas usados, luego en 3.4 se darán a conocer las fuentes de recopilación de información. En la Sección 3.5 se expone el diseño de la investigación, en la Sección 3.6 se establece el tratamiento de los datos y finalmente en el apartado 3.7 se darán a conocer los recursos humanos y materiales.

3.1. Localización.

La Universidad Técnica Estatal de Quevedo es el lugar en el que se desarrolló la presente investigación. Se encuentra ubicada en la Avenida Quito km. 1 1/2 vía a Santo Domingo de los Tsáchilas. En la calle transversal central entre la Avenida Carlos J. Arosemena y la calle Patria Nueva.

Se encuentra en la parroquia 24 de Mayo del Cantón Quevedo en la provincia de Los Ríos en la República del Ecuador. Las coordenadas geográficas de su ubicación son -1º 0' 45" en latitud y a -79º 28' 10" en longitud.

3.2. Tipo de investigación.

3.2.1. Empírica.

En esta investigación es de carácter empírico porque se realizan varios experimentos con la finalidad de determinar el algoritmo de clasificación supervisada más adecuado para generar recomendaciones de productos bancarios.

3.2.2. Descriptiva.

En esta investigación se identificaron las características que tienen los algoritmos pertenecientes a las diferentes técnicas de clasificación supervisada que se utilizaron en la comparación. También se describe el comportamiento de los clasificadores construidos por los algoritmos, en cuanto a su capacidad de predicción y el uso de recursos computacionales al aplicarlos a los datos.

3.3. Métodos y técnicas a usar en la investigación.

Los métodos utilizados en la investigación se presentan en la siguiente tabla.

Tabla 1: Métodos usados en la investigación.

Método documental	**Método analítico**	**Método comparativo**	**Método inductivo**
Se realizó la búsqueda de información en textos de diferentes autores y fuentes como libros, artículos de revistas especializadas, diarios, entre otros. La información obtenida en la investigación se utilizó para la elaboración del marco conceptual.	Este método permitió la construcción, la revisión, el análisis y la síntesis de la información de documental recabada. Los datos de la investigación sirvieron para determinar los algoritmos de las técnicas de clasificación que han obtenido mejores resultados en investigaciones similares a la presente.	Se utilizó este método para realizar un análisis de los diferentes modelos de clasificación y para seleccionar los algoritmos de las técnicas de clasificación supervisada que aparecen en las investigaciones revisadas como referencia.	Se utilizó este método al aplicar las técnicas de preprocesamiento y los algoritmos de las técnicas de clasificación supervisadas; y para analizar los resultados obtenidos.

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.4. Fuentes de recopilación de información.

A continuación de detalla las fuentes de información necesarias que se utilizaron para poder cumplir con los objetivos de la investigación.

Tabla 2: Fuentes de recopilación de información.

Fuente primaria	Fuente secundaria
Conjunto de datos de un sitio web donde se realizan competencias entre científicos o investigadores en ciencia de datos llamado Kaggle [51], los cuales son datos de un banco europeo de los clientes y de los productos que ellos han accedido.	La fuente de información secundaria está constituida por los resultados de aplicar las técnicas de preprocesamiento y los resultados de evaluar el desempeño de los clasificadores construidos. Una fuente secundaria de información también fueron los resultados de la comparación mediante las pruebas de hipótesis de Friedman y post hoc Friedman-Nemenyi; así como la información recabada en los libros y artículos de revistas referentes a minería de datos.

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5. Diseño de la investigación.

La presente investigación tiene un diseño experimental intra-sujeto, por el cual se aplicó a los algoritmos de clasificación repetidamente bajo circunstancias variantes. También se utilizó como guía a la metodología para proyectos de minería de datos CRISP-DM, por lo que se dividen los procesos ejecutados en seis fases que a continuación se detallan mediante una breve descripción y un diagrama de flujo de datos.

3.5.1. Descripción de las fases.

Primera fase - comprensión de los datos (objetivo específico 1): En esta fase se realiza el proceso de lectura de datos desde el archivo "train_ver2.csv" y se almacenan en el archivo "data.Rda" para que estos datos sean manipulables en R. Otro de los procesos es el análisis estadístico descriptivo de los datos.

Segunda fase - preprocesamiento primera iteración (objetivo específico 1): Se realiza la aplicación de técnicas de preprocesamiento para la reducción, limpieza y transformación de los datos almacenados en el archivo "data.Rda". El conjunto de datos preprocesados en esta fase se almacena en el archivo "data_clean.Rda".

Tercera fase - modelado primera iteración (objetivo específico 2): Se selecciona un conjunto de algoritmos de clasificación candidatos y se realiza la construcción y evaluación de sus clasificadores a partir de tareas (problemas de clasificación), que están conformadas por muestras de los datos almacenados en "data_clean.Rda". Esta fase se la ejecutará tantas veces como sean necesarias hasta determinar las necesidades de preprocesamiento que aun requieran los datos para poder ser analizados por el conjunto de algoritmos candidatos.

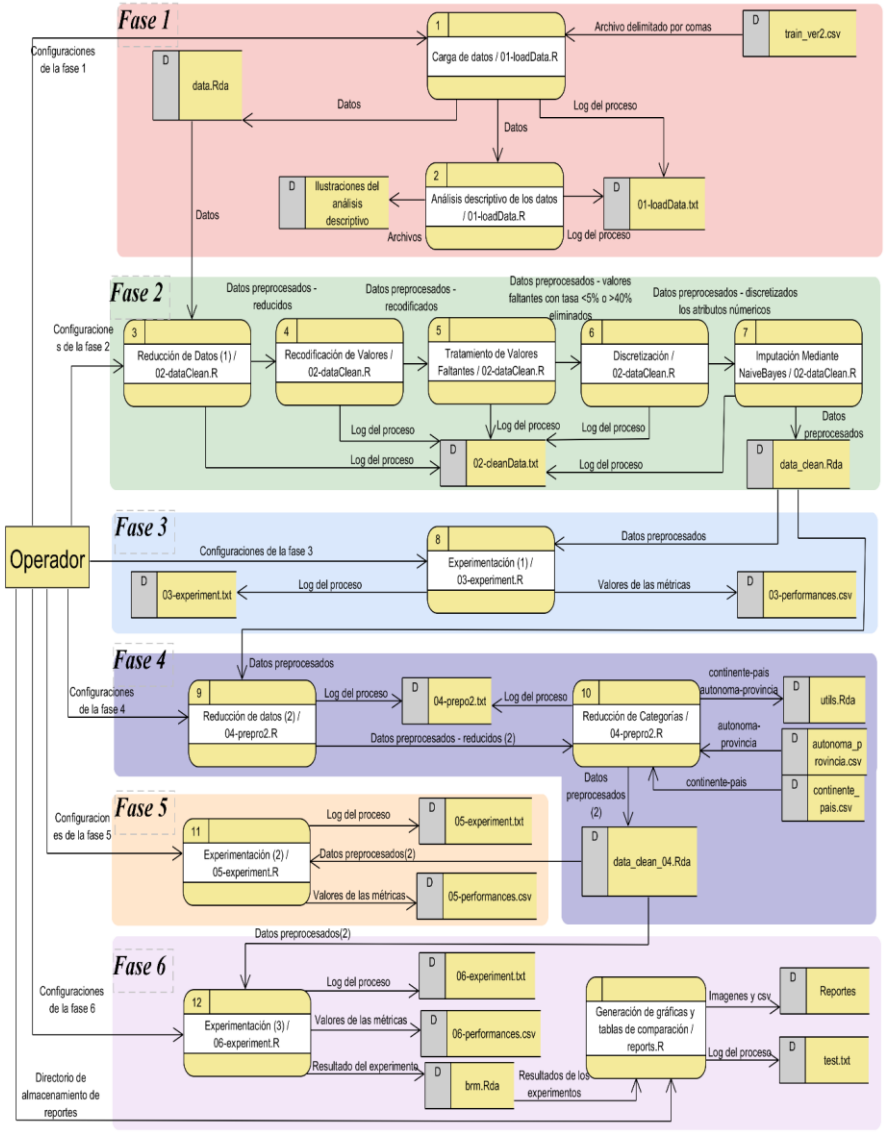
Cuarta fase - preprocesamiento segunda iteración (objetivo específico 1): Se aplican nuevos procesos de preprocesamiento de datos, de acuerdo a los resultados de la ejecución de los algoritmos en la fase anterior. Para ello a los datos almacenados en el archivo "data_clean.Rda", se les realiza una reducción de datos y la reducción de categorías, para posteriormente almacenarlos en el archivo "data_clean_04.Rda".

Quinta fase - modelado segunda iteración (objetivo específico 2): Esta fase consiste en la construcción de los clasificadores de los mismos algoritmos seleccionados en la tercera fase pero aplicándolos sobre la totalidad de los datos de "data_clean_04.Rda". En base a los resultados de esta fase se seleccionan cuatro algoritmos que se van a comparar.

Sexta fase - modelado tercera iteración (objetivo específico 3): Se realiza la experimentación realizando cambios en la cantidad de instancias y el nivel de balanceo de las clases de los datos de "data_clean_04.Rda", para posteriormente entrenar los

clasificadores, evaluar las métricas seleccionadas (AUC, tiempo de entrenamiento y tiempo de prueba) y realizar la comparación de los algoritmos. Y poder así determinar el algoritmo de clasificación supervisada apropiado para recomendar productos bancarios.

Ilustración 9: Diagrama de flujo de datos nivel 1 de la investigación.



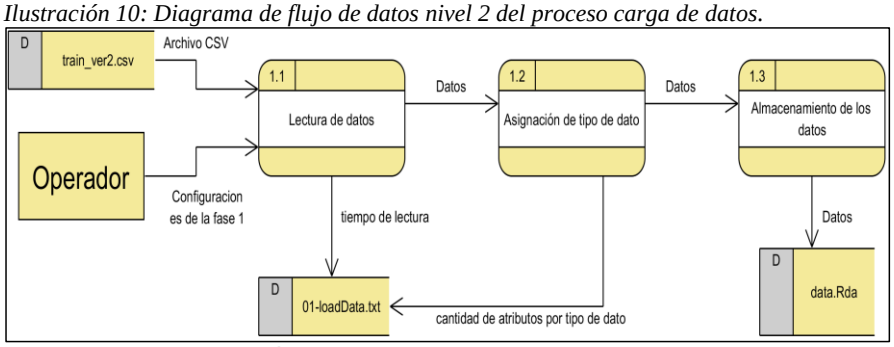
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

A continuación, como parte del diseño de la investigación, se detallan los procesos de cada una de las fases. Indicando de cada proceso las acciones a realizar, el script de R donde se encuentra el código de la implementación del proceso, el anexo donde se encuentra el core del código implementado y un diagrama de flujo de datos del proceso.

3.5.2. Procesos de la primera fase (comprensión de los datos).

Carga de datos: La lectura de los datos desde el archivo "train_ver2.csv", descargado desde el sitio web Kaggle, se lo realiza mediante la función "*read.table"* y posteriormente se lo guarda en el archivo "data.Rda" mediante la función *"save"*. Este proceso se encuentra implementado en el script "01-loadData.R" y el core del código se muestra en el Anexo **3**.

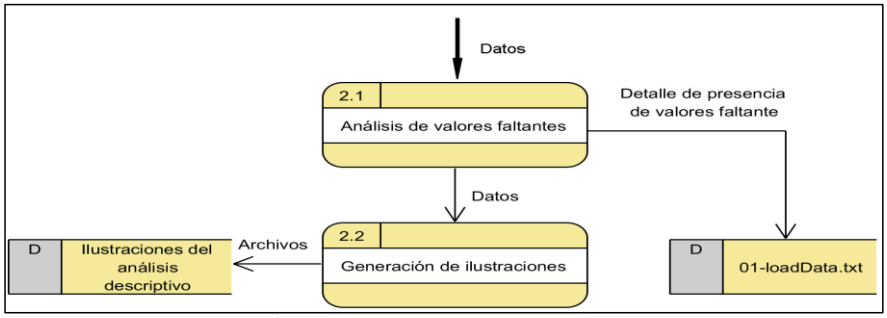
Ilustración 10: Diagrama de flujo de datos nivel 2 del proceso carga de datos.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Análisis descriptivo de los datos: Se realiza un breve análisis de la presencia de valores faltantes en los datos y mediante la función *"ggplot2::ggplot"* se generan las ilustraciones que servirán para describir las características de los datos, estas ilustraciones se las almacenan en archivos d formato png mediante la función *"ggplot2::ggsave*. Este proceso se encuentra implementado en el script "01-loadData.R" y el core del código se muestra en el Anexo 4.

Ilustración 11: Diagrama de flujo de datos nivel 2 del proceso de análisis descriptivo de los datos.

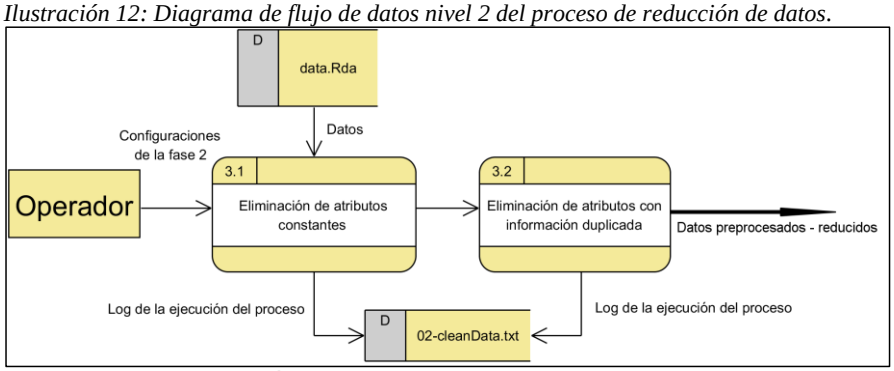


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5.3. Procesos de la segunda fase (preprocesamiento primera iteración).

Reducción de datos: La carga de los datos se la realiza desde el archivo "data.Rda" mediante la función *"load"* y se procede a la eliminación de atributos con datos constantes y atributos con información duplicada. Este proceso se encuentra implementado en el script "02-dataClean.R" y el core del código muestra en el Anexo 5.

Ilustración 12: Diagrama de flujo de datos nivel 2 del proceso de reducción de datos.

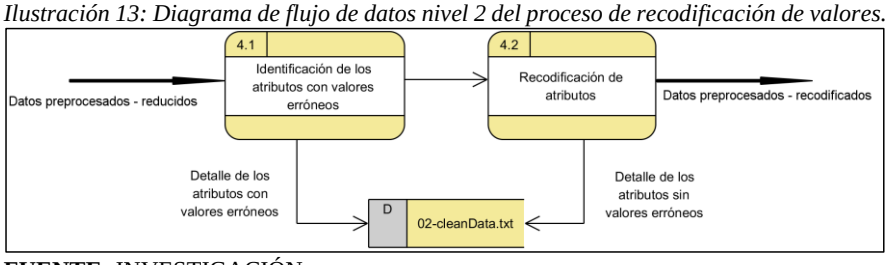


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Recodificación de valores: Se recodifican valores erróneos mediante la función *"dplyr::mutate"*. Estos valores erróneos se pudieron producir durante el ingreso de la información o la transmisión de la misma [18, Ch. 3], por lo que es importante corregir estos

valores. Este proceso se encuentra implementado en el script “02-dataClean.R” y el core del código se muestra en el Anexo 6.

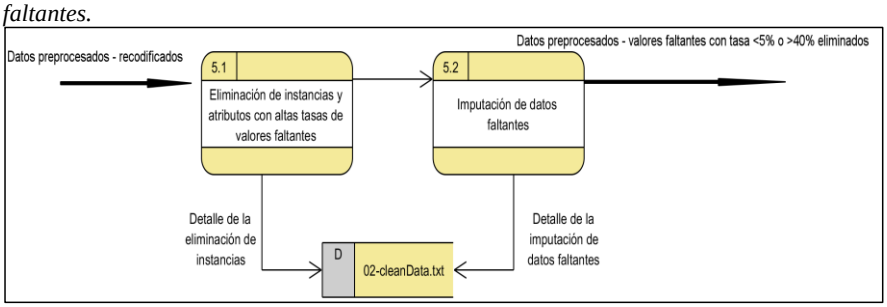
Ilustración 13: Diagrama de flujo de datos nivel 2 del proceso de recodificación de valores.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Tratamiento de valores faltantes: Mediante la función *“plot.missings.cols”* se genera el gráfico de los porcentajes de valores faltantes en los atributos. También en este proceso se eliminarán las instancias que tienen más del 40% de valores faltantes en sus atributos y los atributos con más del 40% de valores faltantes [18], [40, p. 2]. Mediante la función *“mlr::impute”* se imputarán aplicando métodos simples, la moda para atributos nominales y la media para los enteros, a los atributos con una tasa de valores faltantes menores a 5% [40, p. 1]. Este proceso se encuentra implementado en el script “02-dataClean.R” y el core del código se muestra en el Anexo 7.

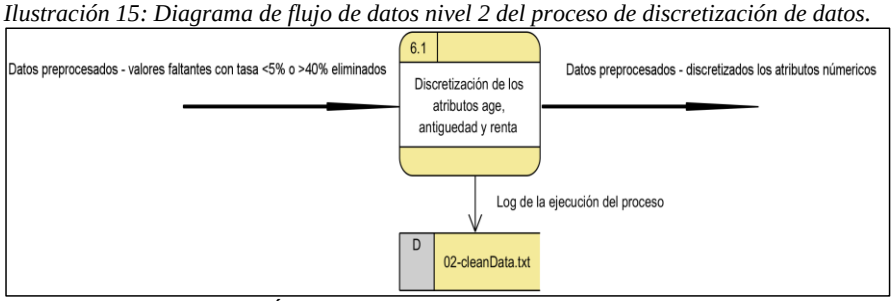
Ilustración 14: Diagrama de flujo de datos nivel 2 del proceso de tratamiento de valores faltantes.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Discretización de datos: La discretización de los atributos numéricos se la realiza utilizando el algoritmo de agrupamiento de datos (clusters) K-Means [52] mediante la función "*arules::discretize"*, especificando la cantidad de 20 categorías (clusters), para tratar de no perder tanta información en el proceso de discretización. Este proceso se encuentra implementado en el script "02-dataClean.R" y el core del código se muestra en el Anexo 8.

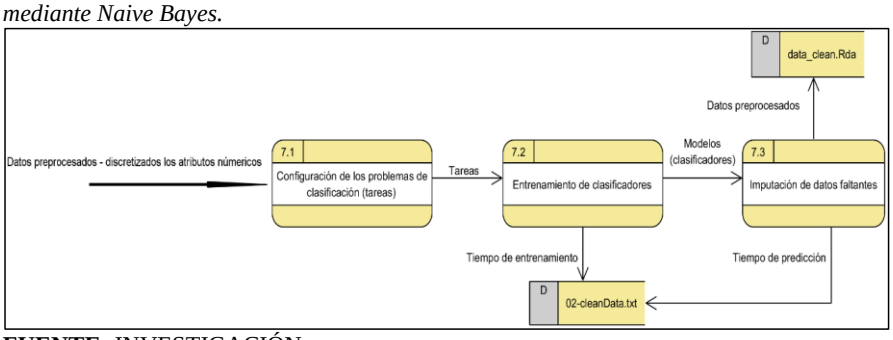
Ilustración 15: Diagrama de flujo de datos nivel 2 del proceso de discretización de datos.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Imputación de datos mediante Naive Bayes: Se empleó la imputación mediante técnicas avanzadas como las que utilizan algoritmos de machine learning, estas técnicas se las aplica a los atributos con una tasa de valores faltantes superior al 5% tal como lo recomiendan en [40, p. 1], en este caso el atributo *renta* se imputo mediante el algoritmo de clasificación de NB ya que en [41] se demuestra su eficiencia y efectividad para el tratamiento de valores faltantes. Para ello se crea la tarea (problema de clasificación) especificando el atributo clase (atributo a imputar) para entrenar el clasificador, se procede a predecir el atributo clase (atributo a imputar) y finalmente se guarda el conjunto de datos preprocesados en el archivo "data_clean.Rda". Este proceso se encuentra implementado en el script "02-dataClean.R" y el core del código se muestra en el Anexo 9.

Ilustración 16: Diagrama de flujo de datos nivel 2 del proceso de imputación de datos mediante Naive Bayes.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5.4. Procesos de la tercera fase (modelado primera iteración).

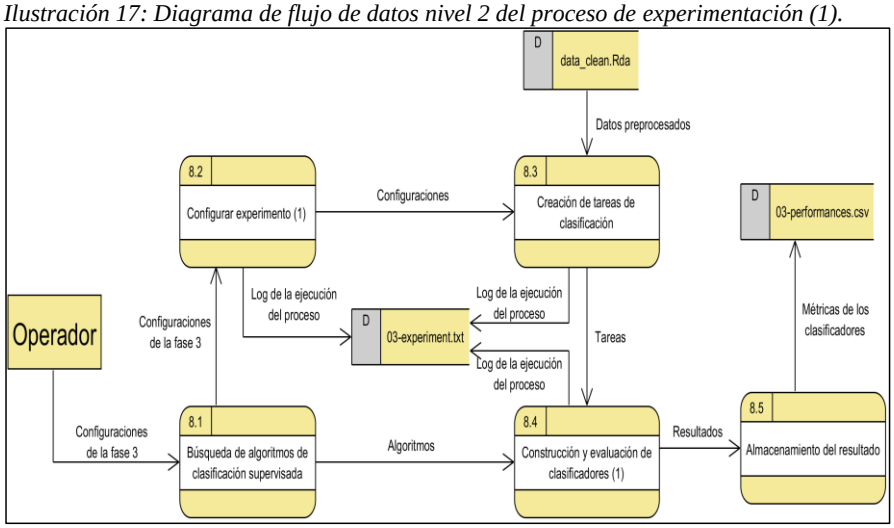
Experimentación (1): Se realiza la selección de los algoritmos de clasificación que conformaran el conjunto de algoritmos candidatos que cumplan las siguientes condiciones: (i) que puedan trabajar con atributos predictores categóricos, (ii) que se hayan utilizado o que hayan obtenido un buen desempeño en investigaciones referentes a esta o en estudios comparativos de técnicas de clasificación entre los años 2009 al año 2017, (iii) que se encuentren implementados en algún paquete del software R y (iv) que su salida sea un valor de probabilidad para calcular la AUC.

Se configura la experimentación mediante parámetros como la cantidad de muestras a generar, que atributos clase (targets) se van a clasificar, cuantos procesadores se van a utilizar, etc, y se prueba la aplicabilidad en los datos preprocesados al conjunto de algoritmos candidatos realizando la construcción y evaluación de sus clasificadores; esto mediante la interacción de los algoritmos y muestras del conjunto de datos. Estas muestras se generan mediante un muestreo estadístico aleatorio simple de la cantidad de instancias disponibles de los datos pre-procesados en base a los productos de banco y a la fecha del dato; y siguiendo las siguientes condiciones para controlar las altas tasas de desbalanceo de clases que podría causar la selección aleatoria de las instancias: (i) es que al generar las muestras se cumpla la tasa de desbalanceo de clases de 1/7 y en caso de no cumplir tal condición, se realice una nueva interacción en busca de una nueva muestra y (ii) que solo se realicen

máximo 100 iteraciones. De cada muestra se procede a generar una tarea de clasificación independiente, este proceso se lo realiza mediante la función *"create.task.main"*.

La construcción y evaluación de los clasificadores se realiza mediante la función "*mlr::benchmark"*. Este proceso se encuentra implementado en el script "03-experiment.R" y el core del código se muestra en el Anexo 10.

Ilustración 17: Diagrama de flujo de datos nivel 2 del proceso de experimentación (1).

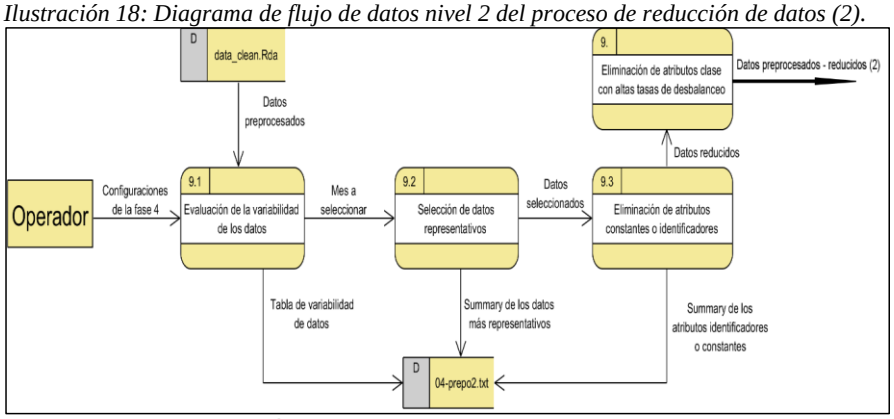


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5.5. Procesos de la cuarta fase (preprocesamiento segunda iteración).

Reducción de datos (2): Se realiza la carga de los datos desde el archivo "data_clean.Rda" mediante la función *"load"*. Después se realiza la reducción de instancias descartando las que no correspondan al valor del último mes en el atributo *fecha_dato*; es decir las instancias que sean de la fecha "2016-05-28", esto se lo realiza porque era el mes con mayor cantidad de clientes, y la cantidad contrataciones de productos del banco por parte de los clientes no tenía gran diferencia con las cantidades de contrataciones de los meses anteriores. Y por último se elimina los atributos clase con altas tasas de desbalanceo. Este proceso se encuentra implementado en el script "04-prepro2.R" y el core del código muestra en el Anexo 11.

Ilustración 18: Diagrama de flujo de datos nivel 2 del proceso de reducción de datos (2).



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Reducción de categorías: Se realiza la lectura de los archivos "autonoma_provincia.csv" y "continente_pais.csv", los cuales contienen la información de las provincias de cada ciudad autónoma y los países de cada continente, y esta información es almacenada en el archivo "utils.Rda". Después a los atributos *fecha_alta*, *canal_entrada*, *pais_residencia* y *cod_prov* se les realizará una combinación de categorías basada en la lógica del negocio, tal como lo indican en [53]. A continuación se establece el procedimiento para cada atributo:

- ✓ Al atributo *fecha_alta* se le reducirá las categorías mediante la transformación de la fecha a la cantidad de meses que han pasado hasta el preprocesamiento, esto se lo realizará con la función "*fechas2meses*".

- ✓ Al atributo *canal_entrada* se le reducirá las categorías mediante la transformación del nombre del canal de entrada a su abreviación, esto se lo realizará con la función *"reduce.canales"*.

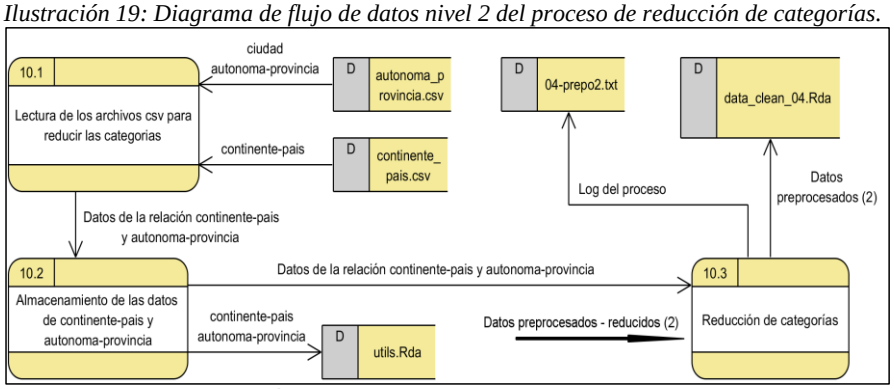
- ✓ Al atributo *pais_residencia* se le reducirá sus categorías mediante la asignación del continente a partir del país de residencia, esta información fue basada en la tabla de países y continentes de [54], esto se lo realizará con la función *"pais2continente"*.

- ✓ Al atributo *cod_prov* se le reducirá sus categorías mediante la asignación de la ciudad autónoma a la que corresponde cada código de provincia, basándose en la forma

como está dividida España según los datos de [55], esto se lo realizará con la función *"prov2autonoma"*.

Finalmente los datos preprocesados se almacenan en el archivo "data_clean_04.Rda". Este proceso se encuentra implementado en el script "04-prepro2.R" y el core del código se muestra en el Anexo 12.

Ilustración 19: Diagrama de flujo de datos nivel 2 del proceso de reducción de categorías.

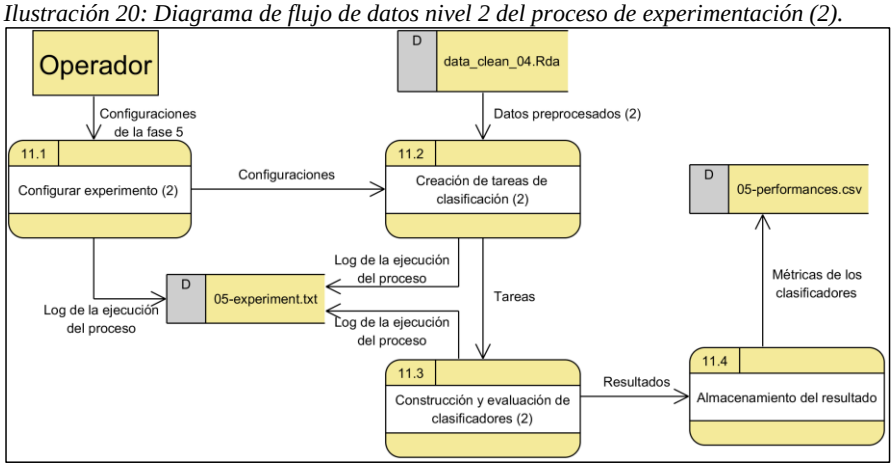


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5.6. Procesos de la quinta fase (modelado segunda iteración).

Experimentación (2): Con la finalidad de seleccionar cuatro algoritmos de clasificación que sean los más apropiados para realizar la comparación en la siguiente fase, se aplica el conjunto de algoritmos candidatos sobre la totalidad de los datos preprocesados que están almacenados en el archivo "data_clean_04.Rda". La configuración de la fase para la experimentación será la misma que en la tercera fase, pero cambiando el valor del parámetro de *cores.parallel* a un valor de "*1"* y el parámetro lista *num.reps* a un valor de *"5 y 10"*. Estos cambios se los realizan para evitar un desborde de memoria y que el proceso no requiera un alto tiempo computacional en la construcción y evaluación de los clasificadores que se realizará mediante la función "*mlr::benchmark"*. Este proceso se encuentra implementado en el script "05-experiment.R" y el core del código se muestra en el Anexo 13.

Ilustración 20: Diagrama de flujo de datos nivel 2 del proceso de experimentación (2).

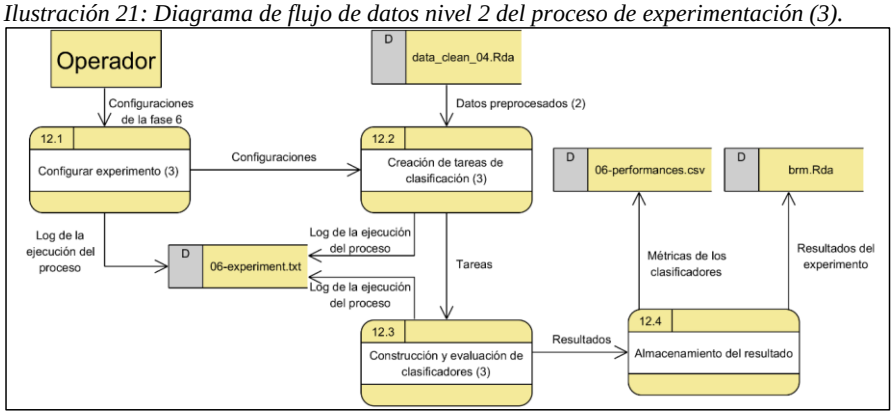


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.5.7. Procesos de la sexta fase (modelado tercera iteración).

Experimentación (3): Con los cuatro algoritmos de clasificación seleccionados, se realiza la construcción y evaluación sucesiva de los clasificadores a partir de subconjuntos de datos con cantidades de instancias crecientes (1000, 5000, 10000, 20000, 50000, 100000 y 200000 instancias) en los cuales, de ser posible, van a tener una tasa de desbalanceo de 1. Se almacena en el archivo "06-performances.csv" los valores de las métricas de AUC, tiempo de entrenamiento y tiempo de prueba, y en el archivo "brm.Rda" se almacenan todos los resultados del experimento. Este proceso se encuentra implementado en el script "06-experiment.R" y el core del código se muestra en el Anexo 14.

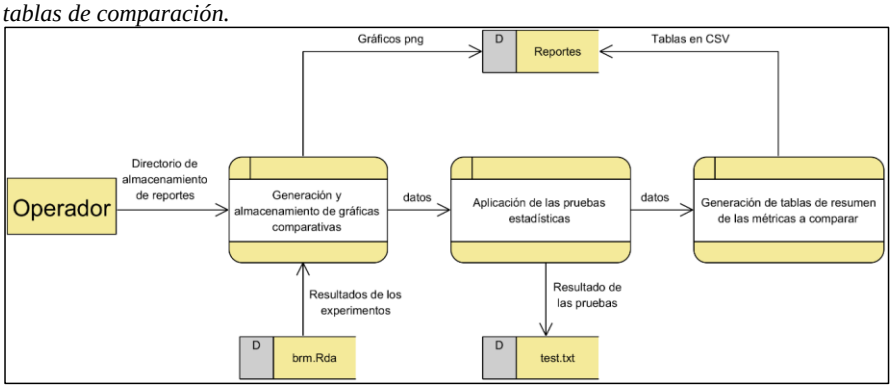
Ilustración 21: Diagrama de flujo de datos nivel 2 del proceso de experimentación (3).



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Generación de gráficas y tablas de comparación: Se realiza la carga de todos los siete archivos brm.Rda de cada ejecución con las diferentes cantidades de datos y se generan tablas, gráficas y un archivo "test.txt" en el cual estará los resultados de las pruebas estadísticas. Los resultados de este proceso servirán para comparar el desempeño de los clasificadores construidos por los cuatro algoritmos seleccionados. Este proceso se encuentra implementado en el script "reports.R" y el core del código se muestra en el Anexo 15.

Ilustración 22: Diagrama de flujo de datos nivel 2 del proceso de generación de gráficas y tablas de comparación.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.6. Tratamiento de datos.

Mediante la utilización del software de R y Excel se realizaron: tablas y gráficas a partir de los resultados de las métricas de evaluación de los clasificadores, en las cuales se expuso el comportamiento de estas con respecto la cantidad de registros y al producto bancario (atributo clase). La prueba estadística utilizada fue la prueba de significación estadística, para la verificación de la hipótesis de mejora del desempeño entre clasificadores. Por lo que fueron utilizadas las pruebas no paramétricas de Friedman y post hoc Friedman-Nemnyi.

3.7. Recursos humanos y materiales.

A continuación de detalla los recursos humanos, de hardware y de software que fueron utilizados en la investigación.

3.7.1. Recursos humanos.

Tabla 3: Recursos humanos

Nombre	Cargo
Sr. Ricardo Villarroel	Autor del presente proyecto
Ing. Iván Jaramillo	Docente Auspiciante del proyecto

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.7.2. Recursos materiales.

3.7.2.1.Hardware.

Tabla 4: Recursos materiales de hardware

Equipo	Características
Servidor HP ProLift	4 procesadores de 2.5 GHz (10 núcleos c/u) 32 GB de RAM 2 TB de disco duro

Notebook Sony Vaio	Procesador Intel(R) Pentium (R) CPU 2117U @ 1.8GHz 8 GB de RAM 750 GB disco duro

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

3.7.2.2.Software.

Tabla 5: Recursos materiales de software

Nombre	**Características**
Sistema operativo (laptop Sony)	Windows 8.1 64 bits
Sistema operativo (servidor HP)	CentOS 6.4.0
Microsoft R Open	Versión 3.3.3
Paquete Machine Learnning en R (MLR)	Versión 2.11.0
Microsoft Office Professional Plus 2013	Versión 15
Visual Paradigm for UML	Versión 10.0

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

Este capítulo muestra los resultados y la discusión del proceso de minería de datos que se ha realizado en la presente investigación, estos resultados se dividen en seis fases, las cuales fueron desarrolladas en base al diseño de la investigación. Para la ejecución de las siguientes fases se utilizó la distribución de R llamada Microsoft R Open v3.3.3 en el Servidor HP ProLift; y cuando fue necesario se utilizó funciones del paquete de R llamado MLR v2.11 [56].

4.1. Primera fase (comprensión de los datos).

4.1.1. Carga de los datos a R.

La lectura de los datos desde el archivo "train_ver2.csv" requirió un tiempo de 3 minutos con 30 segundos. En la *Tabla 6* se detalla el resumen de cantidad de atributos por tipo de datos a la que automáticamente asigno R en la lectura del csv. Después de la correcta asignación del tipo de dato a los atributos se obtiene la *Tabla 7*, con lo que se procede realizar el almacenamiento de los datos en un archivo en formato de R llamado "data.Rda".

Tabla 6: Cantidad de atributos por tipo de dato.

Tipo	Cantidad
factores	15
enteros	32
numéricos	1

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Tabla 7: Cantidad de atributos por tipo de dato después de la corrección.

Tipo	Cantidad
factores	45
enteros	2
numéricos	1

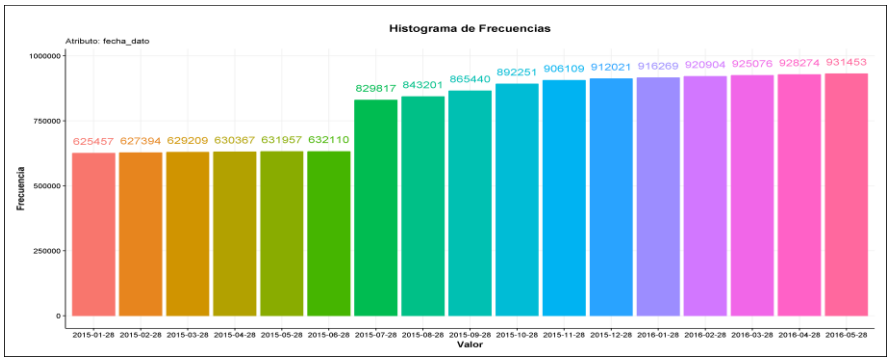
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.1.2. Análisis descriptivo de los datos.

Los datos se podrían dividir en dos grupos: datos de los clientes (24 atributos) y de los productos del banco que el cliente utiliza (24 atributos). Sus dimensiones son 48 atributos y 13’647.309 instancias. Los atributos de los datos cargados desde el archivo “data.Rda” presentan diferentes características, como la tener un 99.99% de valores faltantes en el atributo *conyuemp* entre otras. A continuación se describirá a cada uno de los atributos en detalle:

El atributo *fecha_dato* representa la fecha de captura de la instancia la cual está representada por ejemplo como 2015-01-28, el tipo de dato es de escala proporcional (continuo) y muestra que los datos han sido capturados en el día 28 de cada mes, durante un periodo de 17 meses que comprende desde el mes de enero del 2015 al mes de mayo del 2016.

Ilustración 23: Histograma de frecuencias del atributo "fecha_dato".



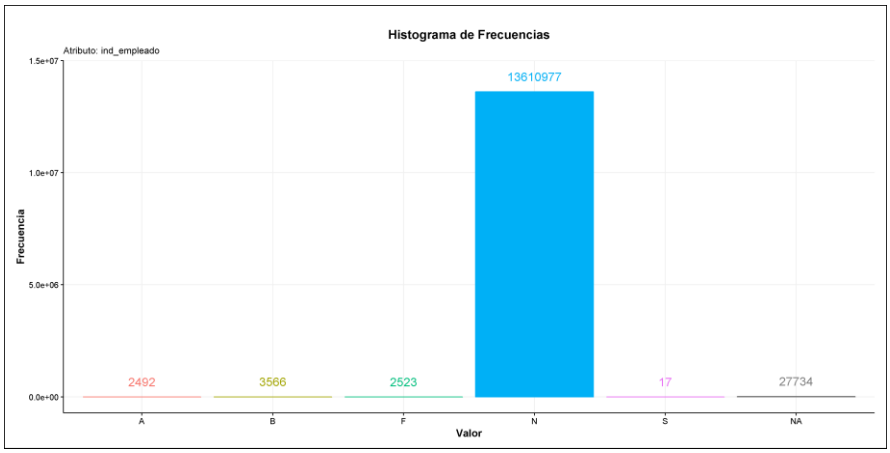
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *ncodpers* representa el código único de identificación del cliente y está representado por ejemplo como 29.071, el tipo de dato es nominal (categórico) y comprende un rango desde el código 15.889 al 1’553.689, existiendo un total de 956.645 clientes únicos.

El atributo *ind_empleado* representa el tipo de relación laboral del cliente con el banco y está representado por ejemplo como “A”, el tipo de dato es nominal (categórico) y comprende un rango de valores como A= Activo, B= Ex Empleado, F=Filial, N= No Empleado y

S=Pasivo. Existen 2.492 instancias que son empleados, 3.566 que son ex empleados, 2.523 son filiales, 13’610.977 instancias que no son empleados, 17 son pasivos y 27.734 instancias presentan un valor en blanco del atributo.

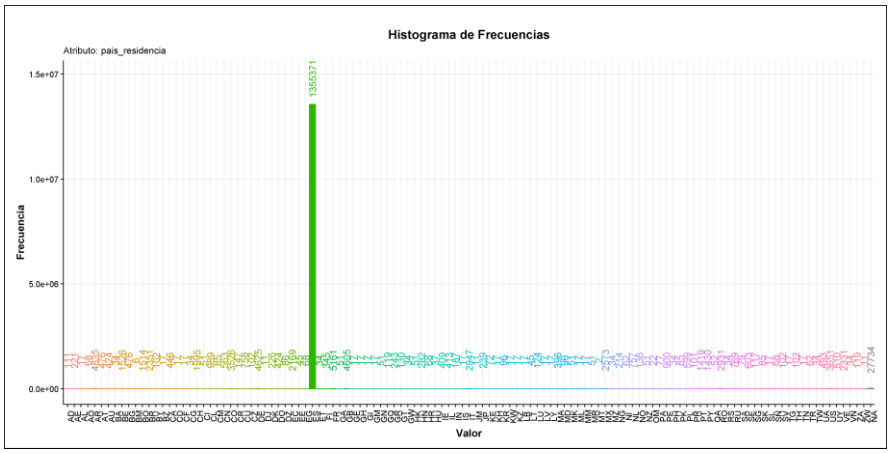
Ilustración 24: Histograma de frecuencias del atributo "fecha_ind_empleado".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *pais_residencia* representa el país de residencia del cliente, se encuentra representado por ejemplo como “ES”, el tipo de dato es nominal (categórico). Existen 118 países distintos en que los clientes residen, ejemplo de España (ES) son 13’553.710 clientes, también se encuentran clientes con residencia en México (MX) 2.573, Brasil (BR) 2.351, Ecuador (EC) 2.169, en blanco 27.734 instancias, etc.

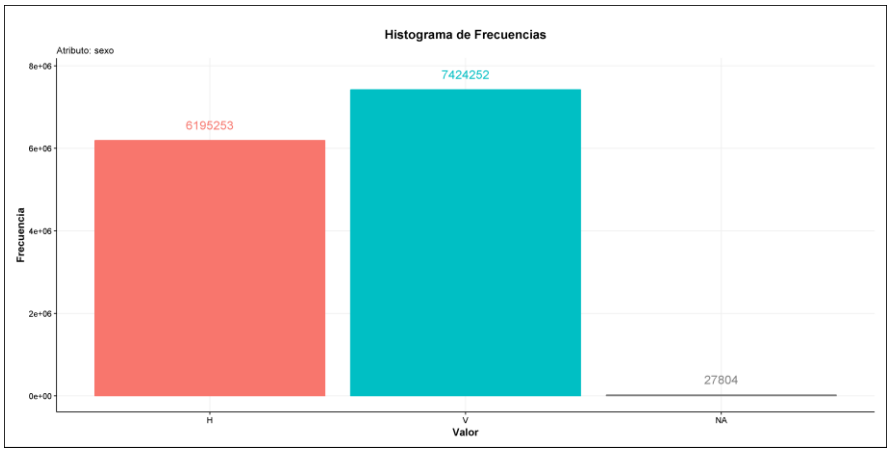
Ilustración 25: Histograma de frecuencias del atributo "pais_residencia".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *sexo* representa el sexo del cliente, se encuentra representado por ejemplo como "H" y "V". El tipo de dato es nominal (categórico) y contiene 27804 valores en blanco, 6'195.253 valores en "H" y 7'424.252 valores en "V".

Ilustración 26: Histograma de frecuencias del atributo "sexo".



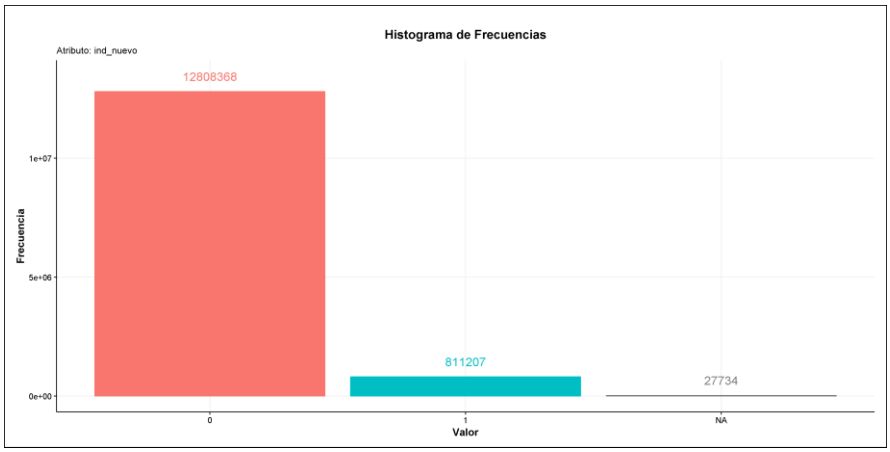
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *age* representa la edad del cliente, es de tipo entero (continuo) y presenta un valor mínimo de 2 años, una media de 40.18 años y máximo de 164 años.

El atributo *fecha_alta* representa la fecha en que el cliente firmo el primer contrato con el banco, se encuentra representado por ejemplo como “2013-08-03”. El tipo de dato es de escala proporcional (continuo) y está compuesto por 27.734 valores en blanco y 6.756 fechas distintas.

El atributo *ind_nuevo* representa si el cliente se ha registrado en los últimos 6 meses, se encuentra representado por ejemplo como “1”, el tipo de dato es binario (categórico) y contiene 27.734 valores en blanco, 12’808.368 valores en “0” y 811.207 valores en “1”.

Ilustración 27: Histograma de frecuencias del atributo "ind_nuevo".

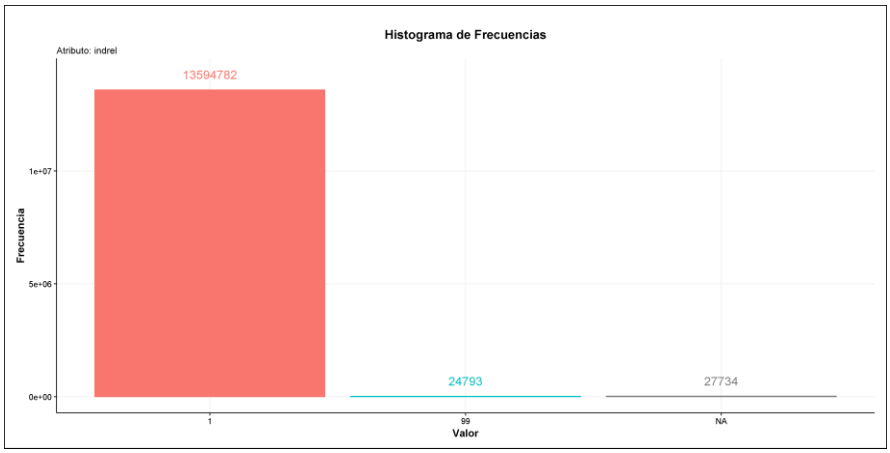


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *antiguedad* representa la antigüedad del cliente en el banco en meses, es de tipo entero (continuo) y presenta un valor mínimo de 1 mes, una media de 76,59 meses y máximo de 256 meses.

El atributo *indrel* representa el índice de relación del cliente con el banco, si es “1” es un cliente principal al principio y final del mes; y “99” representa que es cliente principal al principio del mes pero no al final del mes, es un atributo de tipo binario (categórico).

Ilustración 28: Histograma de frecuencias del atributo "indrel".

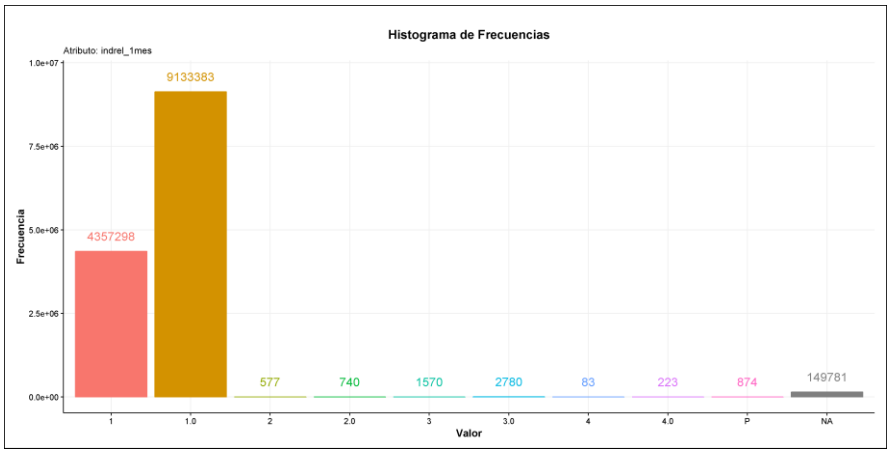


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *ult_fec_cli_1t* representa la última fecha como cliente es principal, se encuentra representado por ejemplo como “2015-10-19”. El tipo de dato es de escala proporcional (continuo), además contienen 223 diferentes fechas y 13’622.516 valores en blanco.

El atributo *indrel_1mes* representa el tipo de cliente al inicio del mes, se encuentra representado por ejemplo como “1” (cliente principal), “2” (cliente secundario), “P” (potencial), “3” (ex cliente principal), “4” (ex cliente secundario). El tipo de dato es nominal (categórico) y contiene 149.781 valores en blanco, 4’357.298 con valor de “1”, 9’133.383 con valor de “1.0”, 577 con valor de “2”, 740 con valor de “2.0”, 1.570 con valor de “3”, 2.780 con valor de “3.0”, 83 con valor de “4”, 223 con valor de “4.0” y 874 con valor de “P”.

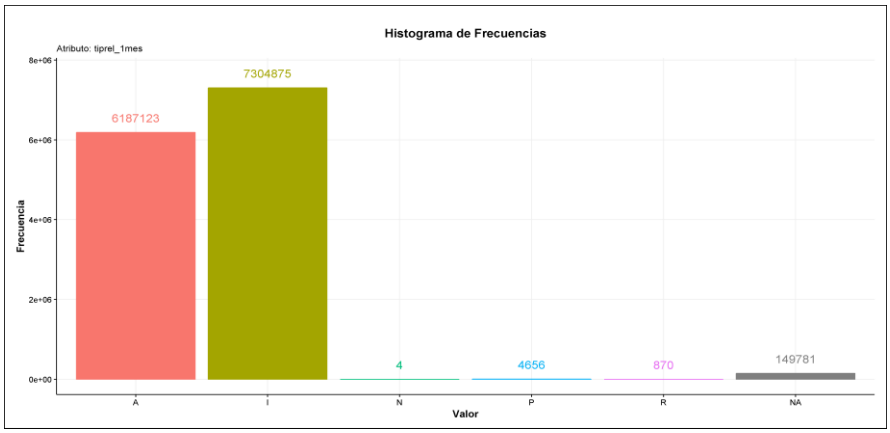
Ilustración 29: Histograma de frecuencias del atributo "indrel_1mes".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *tiprel_1mes* representa el tipo de relación del cliente con el banco al inicio del mes, se encuentra representado por ejemplo como “A” (activo), “I” (inactivo), “P” (ex cliente) y “R” (potencial). El tipo de dato es nominal (categórico) y contiene 149.781 valores en blanco, 6’187.123 con valor de “A”, 7’304.875 con valor de “I”, 4 con valor de “N”, 4.656 con valor de “P” y 870 con valor de “R”.

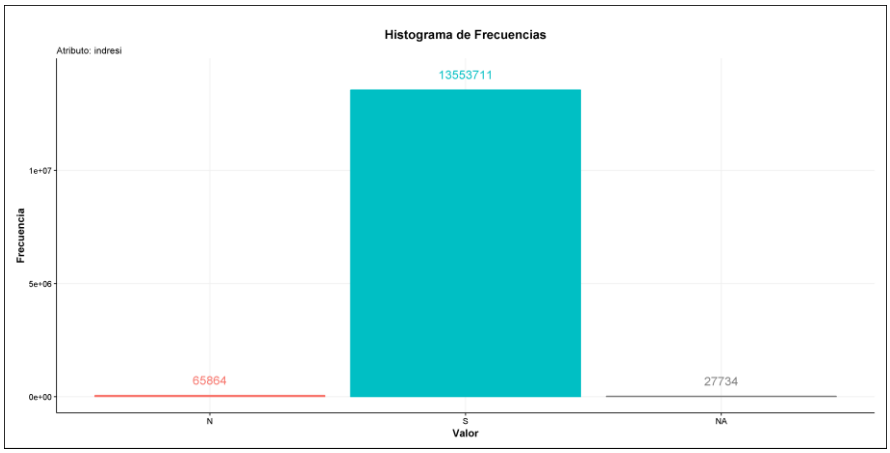
Ilustración 30: Histograma de frecuencias del atributo "tiprel_1mes".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *indresi* representa la residencia del cliente, con valores de (S (Sí) o N (No) si el país de residencia es el mismo que el país del banco, el tipo de dato es binario (categórico) y contiene 27.734 valores en blanco, 65.864 con valor “N” y 13’553.711 con valor “S”.

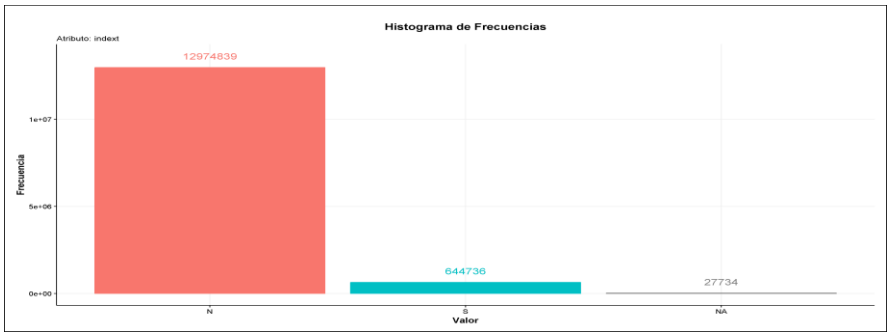
Ilustración 31: Histograma de frecuencias del atributo "indresi".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *indext* representa si el país de nacimiento del cliente es igual al del banco, con valores de (S (Sí) o N (No), el tipo de dato es binario (categórico) y contiene 27.734 valores en blanco, 12’974.839 con valor “N” y 644.736 con valor “S”.

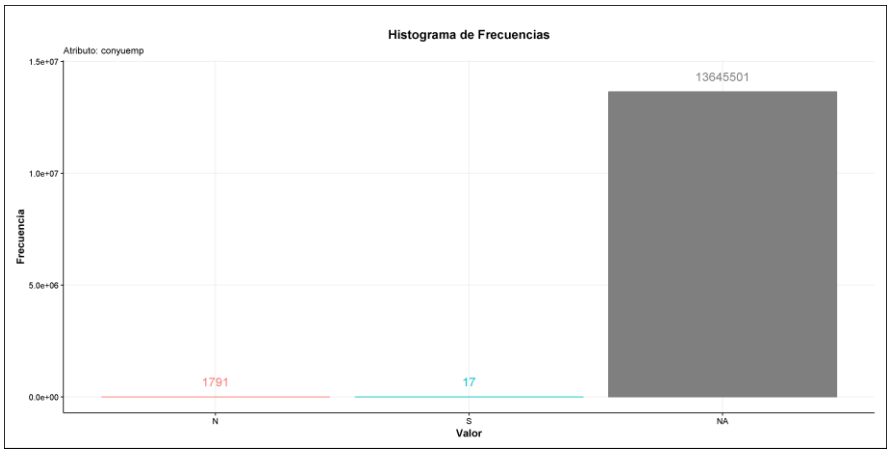
Ilustración 32: Histograma de frecuencias del atributo "indext".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *conyuemp* representa si el cliente es cónyuge de algún empleado del banco, con valores de “1” o “0”, el tipo de dato es binario (categórico) y contiene 13’645.501 valores en blanco, 1.791 con valor “N” y 17 con valor “S”.

Ilustración 33: Histograma de frecuencias del atributo "conyuemp".

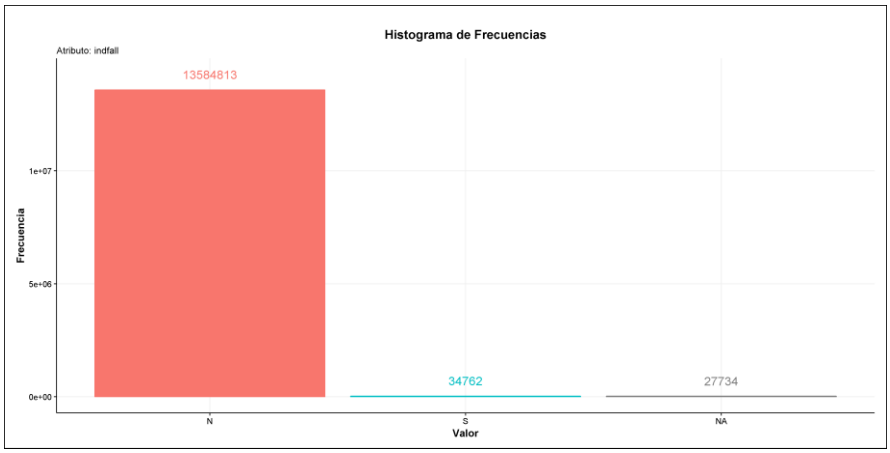


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *canal_entrada* representa el método de captación del cliente por el que se unió al banco. El tipo de dato es nominal (categórico) y contienen 162 diferentes métodos de captación y 186.126 valores en blanco.

El atributo *indfall* representa si el cliente ha fallecido, con valores de “S” o “N”, el tipo de dato es binario (categórico) y contiene 27.734 valores en blanco, 13’584.813 con valor “N” y 34.762 con valor “S”.

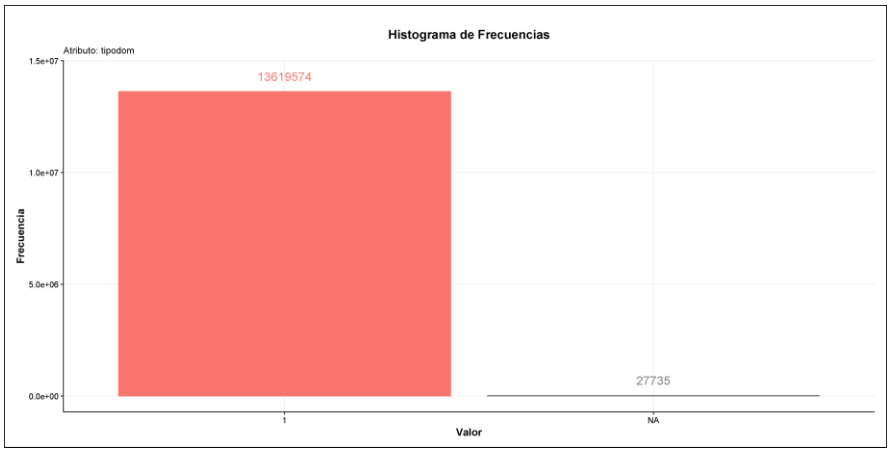
Ilustración 34: Histograma de frecuencias del atributo "indfall".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *tipodom* representa si se tiene la dirección principal del cliente, el tipo de dato es binario (categórico) y contiene 13’619.574 valores en “1” y 27.735 con valores faltantes “NA”.

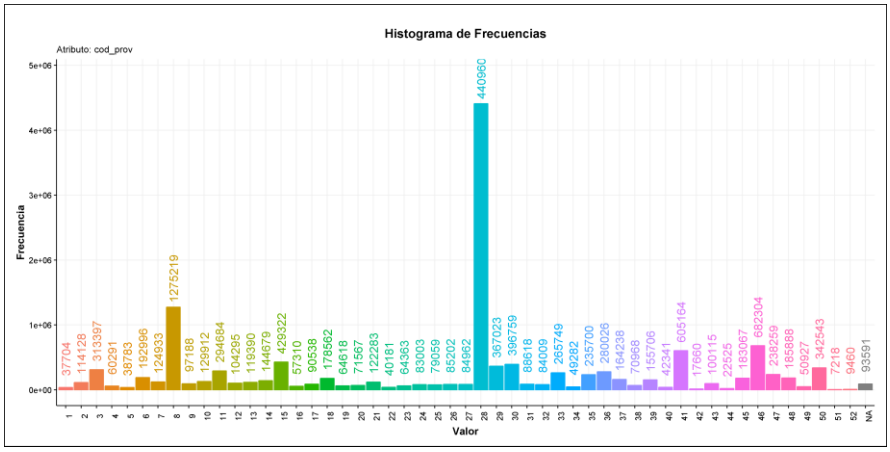
Ilustración 35: Histograma de frecuencias del atributo "tipodom".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *cod_prov* representa el código de la provincia de la dirección del cliente, el tipo de dato es nominal (categórico).

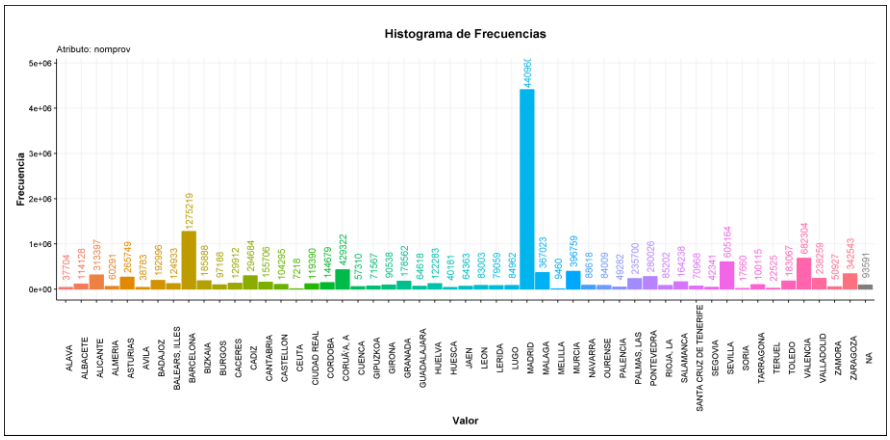
Ilustración 36: Histograma de frecuencias del atributo "cod_prov".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *nomprov* representa el nombre de la provincia de la dirección del cliente, el tipo de dato es nominal (categórico).

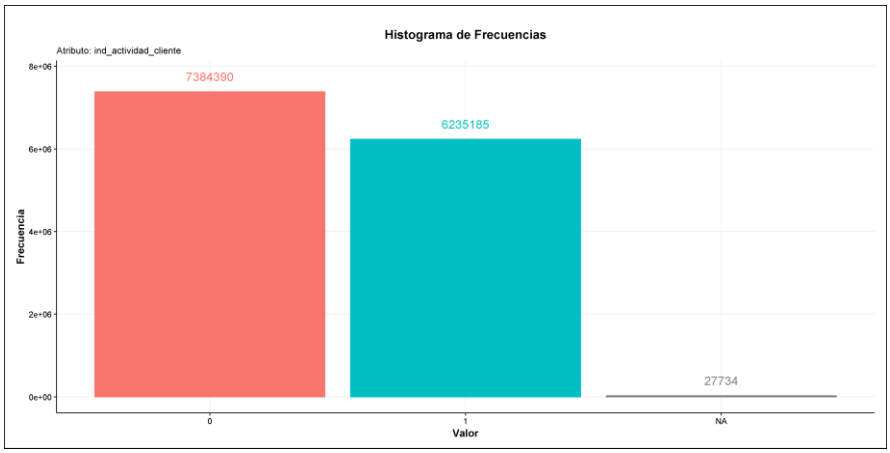
Ilustración 37: Histograma de frecuencias del atributo "nomprov".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *ind_actividad_cliente* representa si el cliente es activo “1” o inactivo “0”, el tipo de dato es binario (categórico) y contiene 7’384.390 instancias con valor “0”, 6’235.185 con valor “1” y 27.734 con valores faltantes “NA”.

Ilustración 38: Histograma de frecuencias del atributo "ind_actividad_cliente".

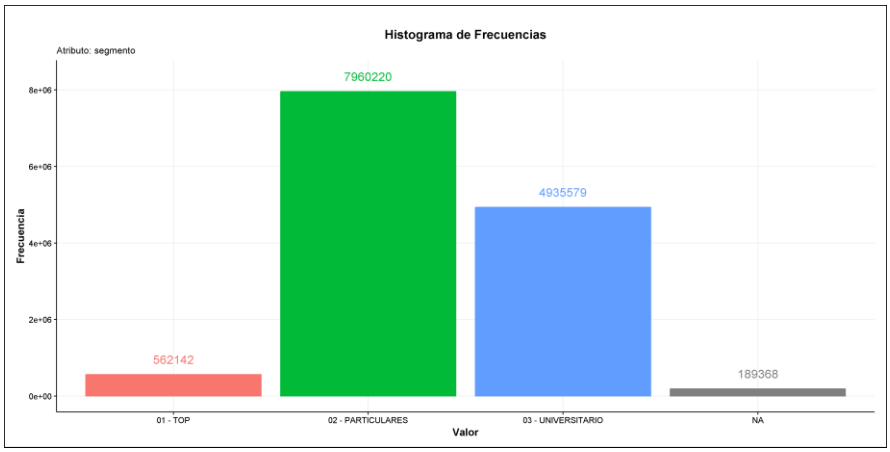


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El atributo *renta* representa los ingresos brutos del cliente, es de tipo de escala proporcional (continuo) y presenta un valor mínimo de 1.203, una media de 134.300 y una máxima de 28’890.000.

El atributo *segmento* representa el segmento al que pertenece el cliente en el banco, el tipo de dato es ordinal (categórico) y contiene 189.368 valores en blanco, 562.142 con valor de “01 - TOP”, 7’960.220 con valor de “02 - PARTICULARES” y 4’935.579 con valor de “03 - UNIVERSITARIO”.

Ilustración 39: Histograma de frecuencias del atributo "segmento".



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El resto de los 24 atributos finales representan los productos a los que han accedido los clientes. El tipo de dato de estos atributos es binario (categórico) y si el valor del atributo es "1" indica que el cliente posee tal producto del banco. Estos atributos se representan en la *Tabla 8.*

Tabla 8: Descripción de los atributos de los productos que han accedido los clientes del banco.

#	Atributo	Definición del atributo (Producto)	Clases negativas	Clases positivas	NA
25	ind_ahor_fin_ult1	Cuenta de ahorro.	13'645.913	1.396	0
26	ind_aval_fin_ult1	Garantías.	13'646.993	316	0
27	ind_cco_fin_ult1	Cuenta corriente.	4'701.721	8'945.588	0
28	ind_cder_fin_ult1	Cuenta derivada.	13'641.933	5.376	0
29	ind_cno_fin_ult1	Cuenta de nómina.	12'543.689	1'103.620	0
30	ind_ctju_fin_ult1	Cuenta junior.	13'518.012	129.297	0
31	ind_ctma_fin_ult1	Más cuenta particular	13'514.567	132.742	0
32	ind_ctop_fin_ult1	Cuenta particular.	11'886.693	1'760.616	0
33	ind_ctpp_fin_ult1	Cuenta particular plus.	13'056.301	591.008	0

34	ind_deco_fin_ult1	Depósitos a corto plazo.	13’623.034	24.275	0
35	ind_deme_fin_ult1	Depósitos de mediano plazo.	13’624.641	22.668	0
36	ind_dela_fin_ult1	Depósitos a largo plazo.	13’060.928	586.381	0
37	ind_ecue_fin_ult1	Cuenta electrónica.	12’518.082	1’129.227	0
38	ind_fond_fin_ult1	Fondos.	13’395.025	252.284	0
39	ind_hip_fin_ult1	Hipoteca.	13’566.973	80.336	0
40	ind_plan_fin_ult1	Pensiones.	13’522.150	125.159	0
41	ind_pres_fin_ult1	Préstamos.	13’611.452	35.857	0
42	ind_reca_fin_ult1	Impuestos.	12’930.329	716.980	0
43	ind_tjcr_fin_ult1	Tarjeta de crédito.	13’041.523	605.786	0
44	ind_valo_fin_ult1	Valores.	13’297.834	349.475	0
45	ind_viv_fin_ult1	Cuenta familiar.	13’594.798	52.511	0
46	ind_nomina_ult1	Nómina de sueldos.	12’885.285	745.961	16.063
47	ind_nom_pens_ult1	Pensiones.	12’821.161	810.085	16.063
48	ind_recibo_ult1	Débito directo.	11’901.597	1’745.712	0

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.2. Segunda fase (preprocesamiento primera iteración).

4.2.1. Reducción de datos.

Se eliminaron los atributos *conyuemp* y *tipodom*. El atributo *conyuemp* se eliminó porque al existir 17 casos positivos, 1.791 casos negativos y los restantes 13´645.501 eran valores faltantes, este atributo se comportaba más como una constante. El atributo *tipodom* también fue eliminado por su comportamiento como constante debido a que 27.735 instancias eran valores faltantes y el restante de 13´619.574 eran puros casos positivos.

Se eliminó el atributo *nomprov*, al ya estar representada su información en el atributo *cod_prov*, así se intentó reducir la dimensionalidad del problema eliminando atributos redundantes.

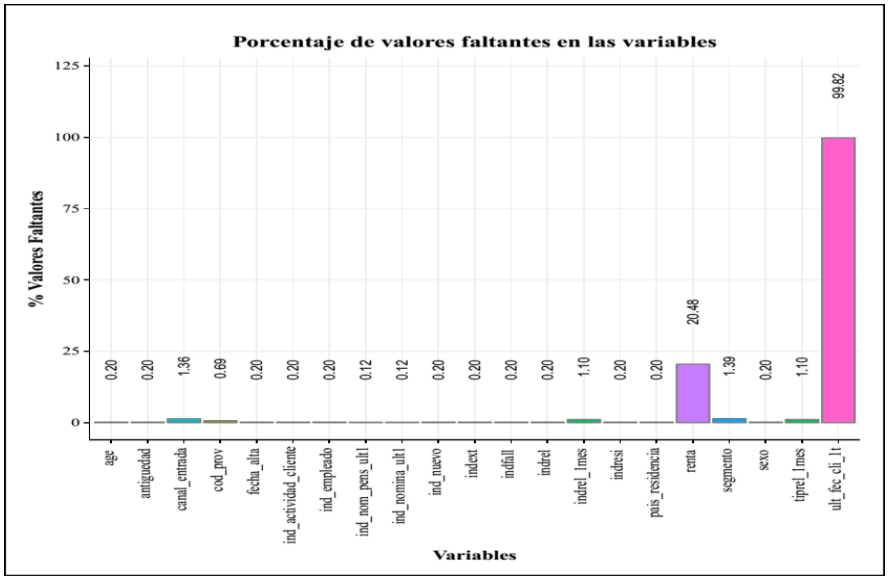
4.2.2. Recodificación de valores erróneos.

El atributo *indrel_1mes* se recodificó ya que existían los valores erróneos: 1, 1.0, 2, 2.0, 3, 3.0, 4, 4.0 y P; y se recodifico a: 1, 2, 3, 4 y P respectivamente.

4.2.3. Tratamiento de valores faltantes.

Tomando en cuenta que los valores en blanco también son considerados como valores faltantes se encontró que el atributo *ult_fec_cli_1t* (última fecha como cliente principal) presentaba el porcentaje de valores faltantes más alto, específicamente 99.82%; seguido del atributo *renta* (ingreso neto del hogar del cliente) con un 20.48%. Como resumen se obtuvo la *Ilustración 40* del análisis de valores faltantes.

Ilustración 40: Porcentaje de valores faltantes por atributo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Al tratar la existencia de valores faltantes se eliminó un total de 27.734 instancias y se eliminó el atributo *ult_fec_cli_1t*. El proceso de imputación mediante la función de *"mlr::impute"* se realizó en un tiempo de 36 segundos.

4.2.4. Discretización de datos.

El proceso de discretización de los atributos *age*, *antiguedad y renta* requirió un tiempo de procesamiento de 11, 12 y 31 segundos respectivamente. Esto permitió al siguiente proceso imputar mediante el algoritmo de NB, ya que NB necesita que todos los atributos del conjunto de datos a trabajar sean categóricos.

4.2.5. Imputación de datos mediante Naive Bayes.

La imputación por medio del algoritmo de clasificación NB se demoró un tiempo de 1 hora con 7 minutos. Una vez culminado todos los procesos de esta segunda fase se obtiene un conjunto de datos preprocesados con dimensiones de 44 atributos y 13´619.575 instancias, los cuales fueron almacenados en el archivo "data_clean.Rda".

4.3. Tercera fase (modelado primera iteración).

4.3.1. Experimentación (1).

4.3.1.1.Búsqueda de algoritmos de clasificación supervisada.

Para la selección de los algoritmos se tuvo en cuenta las condiciones que se establecieron en el diseño de la investigación. En la *Tabla 9* se detalla los algoritmos que participaron en la selección para su aplicación en el presente tipo de problema.

Tabla 9: Algoritmos de clasificación a seleccionar.

Investigación	Técnicas utilizadas	Tipos de datos	Algoritmo con mejor desempeño
[49]	NB, Árboles de decisión , SVM	Categórico, Numérico	SVM
[14]	MLPNN y C5.0	Categórico, Numérico	C5.0
[15]	MLPNN, Regresión logística, NB y C5.0	Categórico, Numérico	C5.0
[8]	J48, NB, K-NN y NNET	Categórico	J48
[57]	LDA , NNET y SVM	Numérico	NNET
[16]	C5.0	Categórico, Numérico	C5.0
[58]	C5.0, EvTree, CART y Árbol oblicuo	Categórico, Numérico	C5.0
[37]	Regresión logística, C4.5, NNET, Gradient boosting, LDA, QDA, LS-SVM, K-NN y Random Forests,	Categórico, Numérico	Random Forests

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Después de analizar las conclusiones y resultados de las investigaciones de la *Tabla 9* se realizó la selección del siguiente conjunto de algoritmos de clasificación candidatos. A continuación se presenta la fundamentación de la selección y el paquete de R donde se encuentre implementado el algoritmo:

- ✓ **SVM:** Perteneciente a los algoritmos clasificadores basados en Máquinas de Vectores de Soporte, es un algoritmo robusto que se encuentra implementado en

varios paquetes de R y ha obtenido un alto rendimiento predictivo sobre un conjunto de datos de un banco de Portugal sobre marketing directo [49]. El paquete de R a utilizar es el paquete llamado *e1071* y que mediante la binarización que es una técnica de transformación de datos nominales a numéricos [18, Ch. 3.5.8] le permite trabajar con datos categóricos.

- ✓ **NB:** Perteneciente a los algoritmos clasificadores estadístico y aunque es un algoritmo simple que utiliza la probabilidad, cuando es aplicado a grandes cantidades de datos obtiene un alto rendimiento y en algunos estudios han encontrado que en algunos dominios obtienen desempeños comparables con los de NNET y de Árboles de decisión [3]. Además su principal requerimiento es que los atributos predictores sean categóricos, el paquete de R a utilizar es el *e1071*.

- ✓ **NNET:** Es un algoritmo de redes neuronales que cuenta con las ventajas de ser fácil de implementar y que tiene gran aplicabilidad en varios problemas donde ha obtenido un buen desempeño comparado con los algoritmos de SVM y LDA [57]. Mediante la binarización de datos nominales puede trabajar con datos categóricos en la implementación del paquete *nnet*.

- ✓ **C5.0:** Es un algoritmo basado en árboles de decisión a partir del algoritmo C4.5. Este guarda las funciones del C4.5 pero agregando nuevas tecnologías como lo es Boosting, lo cual lo utiliza para poder reducir el sesgo y la variancia. Otra es la posibilidad de construcción de un árbol sensible a los costos [16]. Este algoritmo en la mayoría de aplicaciones presenta un buen desempeño en la clasificación [14]–[16]. Se utilizará la implementación del algoritmo del paquete *C50*.

- ✓ **Random Forests:** Forma parte de un grupo de algoritmos de árboles de clasificación o de regresión sin poda. Este se basa en el entrenamiento bootstrap de los datos de entrenamiento usando la selección aleatoria de las variables en el proceso de generación del árbol [37]. Se utilizara el paquete *randomForest*.

- ✓ **K-NN:** Es un algoritmo perteneciente a los métodos basados en los vecinos cercanos, basado en memoria, que realiza el cálculo de los k vecinos más cercanos de los objetos de entrenamiento los cuales son utilizados para caracterizar la clase de un

punto de datos de la muestra [9]. La implementación de este algoritmo a utilizar será la del paquete *kknn*.

- ✓ **EvTree:** Es una función del paquete evtree que sigue los principios descritos por los algoritmos genéticos. Esta función recibe dos grupos de parámetros, donde el primer grupo parametriza el proceso evolutivo y el segundo restringe el árbol de decisión [58]. La implementación de este algoritmo a utilizar será la del paquete *evtree.*

- ✓ **MARS:** Es el algoritmo perteneciente a métodos estadísticos, el cual se propone para realizar la comparación en base a las siguientes fuentes [26], [29], [59], [60], ya que proporciona un modelo aditivo que son modelos que se consideran muy interpretables [26]. La implementación de este algoritmo a utilizar será la del paquete *earth.*

4.3.1.2.Creación de tareas de clasificación.

Después de la selección de los algoritmos de clasificación de SVM, NB, NNET, C5.0, Random Forests, K-NN, EvTree y MARS; se procedió a realizar un primer encuentro de los datos y estos algoritmos para tener una breve perspectiva de cuáles serían los más adecuados para este problema de clasificación. Para ello se realizó un muestreo estadístico aleatorio simple de la cantidad de instancias disponibles de los datos preprocesados del archivo "data_clean.Rda", con una población de 13'647.309 dando como resultado muestras de 384.

Para la generación de las muestras se tuvo en cuenta que los datos fueron capturados en periodos de 17 meses diferentes y que cada uno de los 24 productos es un problema de clasificación individual. Por lo al especificar la obtención de 5 muestras por producto por cada mes da un total de 2.040 muestras. Para generar estas 2.040 muestras se utilizó las condiciones de muestreo establecidas en la parte de la generación de muestras del diseño de la investigación.

4.3.1.3.Construcción y evaluación de clasificadores.

Mediante el uso de la función "*mlr::benchmark"* se realizó la construcción y evaluación de los clasificadores mediante una serie de 7 pruebas, para ello se configuró ciertos parámetros necesarios para la ejecución de las pruebas que en la *Tabla 10* se detallan.

Tabla 10: Parámetros de configuración para pruebas de la tercera fase.

Nombre	**Descripción**	**Tipo de dato**	**Valor**
parallel.enable	Activación del procesamiento en paralelo	booleano	verdadero
cores.parallel	Cantidad de procesadores	entero	20
seed.parallel	Semilla para la generación de números aleatorios	entero	1234
level.parallel	Nivel de paralelización	carácter	mlr.resample
measures	Lista de métricas para evaluar a los clasificadores	lista	AUC máximo, AUC mínimo, AUC media, Tiempo de entrenamiento y tiempo de prueba
lrns	Lista de algoritmos	lista	SVM, NB, NNET, C5.0, Random Forests, K-NN, EvTree y MARS
num.reps	Lista de las cantidades de iteraciones del proceso de remuestreo.	lista	5, 10 y 50
num.folds	Valor de k para el método de remuestreo K-fold Cross Validation	entero	2
save.models	Si se van a guardar los clasificadores construidos	booleano	Falso

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Teniendo en cuenta las dos condiciones en la generación de muestras de los datos se obtuvieron en promedio una cantidad de 323 muestras por prueba. Como resultado de las 7 pruebas se encontró que:

- ✓ NNET arrojó un error por el exceso de pesos en el modelo lo cual fue causado por la creación del gran número de atributos artificiales resultado del proceso de binarización de los atributos con muchas categorías, como lo son los atributos *fecha_alta*, *canal_entrada*, *pais_residencia* y *cod_prov*.

- ✓ Random Forests no podía ejecutarse debido a la limitación de cantidades de categorías que deben tener los atributos categóricos, ya que la implementación del algoritmo en el paquete *randomForest* limita la cantidad de categorías a 53 categorías como máximo.

- ✓ MARS tampoco terminó todo el proceso ya que terminaban sin encontrar un modelo aditivo que convergiera después de las 25 iteraciones que realizaba, esto pudo ser causado porque MARS necesita más instancias para poder construir su clasificador.

- ✓ Los algoritmos de EvTree, NB, C5.0, K-NN y de SVM sin embargo terminaron sin ninguna novedad ambos procesos, tanto en el entrenamiento del clasificador como en su evaluación mediante las métricas seleccionadas.

4.4. Cuarta fase (preprocesamiento segunda iteración).

De acuerdo a los resultados de la tercera fase, se procede a realizar nuevamente una fase de preprocesamiento de los datos almacenados en "data_clean.Rda". A continuación se detallan los procesos realizados.

4.4.1. Reducción de datos (2).

Después de descartar las instancias que en el atributo *fecha_dato* no tengan la fecha de "2016-05-28", el atributo *fecha_dato* se comportaba como una constante y *ncodpers* como

un atributo identificador. Por lo que se eliminaron ya que no proporcionaban ninguna información útil al proceso de clasificación.

Los atributos clase *ind_ahor_fin_ult1*, *ind_aval_fin_ult1*, *ind_cder_fin_ult1* y *ind_deco_fin_ult1*, presentan pocos casos positivos, siendo 78, 16, 316 y 317 casos positivos correspondientemente, por lo que ocasiona que tengan altas tasas de desbalanceo de clases y se decida eliminar estos atributos.

4.4.2. Reducción de categorías.

Se procedió a reducir el número de categorías de los atributos *fecha_alta*, *canal_entrada*, *pais_residencia* y *cod_prov*; ya que estos atributos causaron que no se ejecuten correctamente los algoritmos de NNET y de Random forests. Resultado de esto se logró reducir al atributo *fecha_alta* a la cantidad de 257 meses distintos, al atributo *canal_entrada* a 14 canales de entrada distintos, al atributo *pais_residencia* a 6 posibles continentes de residencia y al atributo *cod_prov* a 19 ciudades autónomas posibles. Como resultado de este y el anterior proceso se obtuvo un conjunto de datos que se almaceno en el archivo "data_clean_04.Rda".

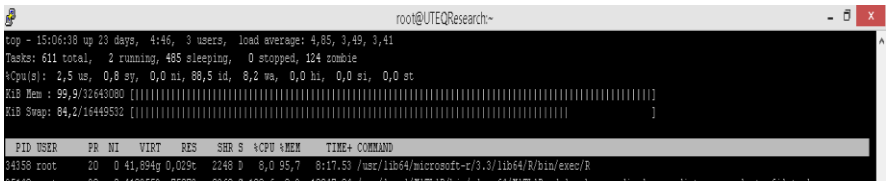
4.5. Quinta fase (modelado segunda iteración).

4.5.1. Experimentación (2).

Los resultados de la construcción y evaluación de los clasificadores fueron los siguientes:

Los algoritmos C5.0, Random forests, EvTree y K-NN; resultaron no ser idóneos para este problema de clasificación ya que todos causaban un desborde de memoria en la construcción del clasificador debido a la alta dimensionalidad de los datos y las limitaciones de recursos del servidor.

Ilustración 41: Ejecución de Random Forest causando desborde de memoria.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

En cambio los algoritmos SVM, NB, NNET y MARS, realizaron el proceso de construcción y evaluación sin problemas, por lo que se procede a seleccionar a estos cuatro algoritmos para la comparación.

4.6. Sexta fase (modelado tercera iteración).

Esta fase consiste en la construcción y evaluación de los clasificadores de los algoritmos de SVM, NB, NNET y MARS. Los datos a usar son los que están almacenados en “data_clean_04.Rda" y se configuraron los mismos parámetros de la *Tabla 10* pero realizando cambios de valores a los parámetros que se muestran en la *Tabla 11*.

Tabla 11: Parámetros de configuración para pruebas de la sexta fase.

Nombre	Descripción	Tipo de dato	Valor
cores.parallel	Cantidad de procesadores	entero	10
level.parallel	Nivel de paralelización	carácter	mlr.resample
lrns	Lista de algoritmos	lista	SVM, NB, NNET y MARS
num.reps	Lista de las cantidades de iteraciones del proceso de remuestreo.	lista	5

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.1. Experimentación (3).

Se construyeron subconjuntos de datos realizando la manipulación de la cantidad de instancias y la tasa de desbalanceo de clases mediante la función de R "*create.task"* y "*create.task.main2"*.

Después de obtener los subconjuntos de datos se procede a realizar la experimentación con los algoritmos del parámetro *lrns* de la *Tabla 11*, este proceso se lo realizó en siete pruebas independientes enviando como parámetro la cantidad de 1000, 5000, 10000, 20000, 50000, 100000 y 200000 instancias respectivamente. Por lo que el proceso de construcción y evaluación de los clasificadores se realizó un total de 2800 veces: (20 productos) * (4 algoritmos) * (5 iteraciones en el remuestreo) * (7 pruebas) = 2800. Todos los resultados de las métricas de evaluación de las 7 pruebas independientes se guardaron en un archivo csv llamado "06-resumen-final.csv", que en la *Tabla 13* se detalla los atributos que contiene y que serán utilizados de aquí en adelante para realizar la comparación de los clasificadores construidos por los algoritmos de clasificación.

Tabla 12: Atributos del archivo de los resultados de las métricas.

variables	descripcion
instancias	Cantidad de instancias en la prueba.
task.id	Producto del banco.
learner.id	Algoritmo.
auc.test.mean	Media del valor de AUC.
auc.test.min	Valor mínimo de AUC.
auc.test.max	Valor máximo de AUC.
timetrain.test.mean	Media del tiempo de entrenamiento (s).
timepredict.test.mean	Media del tiempo de prueba (s)

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.2. Generación de gráficas y tablas de comparación.

Para la comparación de los algoritmos de clasificación se generaron tablas y gráficas de las medidas de desempeño de los clasificadores para establecer la superioridad de un algoritmo sobre los otros. También se aplicó pruebas estadísticas para detectar la existencia de diferencias significativas entre las medidas desempeño, estas fueron las pruebas estadísticas no paramétricas de Friedman y post hoc Friedman-Nemenyi.

4.6.3. Comparación de los algoritmos de clasificación.

4.6.3.1.Tabulación de los resultados de la experimentación.

El resultado de las métricas de evaluación de los clasificadores que se muestran en la *Tabla 13* son en base a cálculos de grupos formados en función de la cantidad de instancias (registros) y algoritmos donde: *AUC*, *Tiempo de entrenamiento* y *Tiempo de prueba* tienen un valor promediado del grupo; *AUC Min* es el mínimo valor de AUC del grupo y *AUC Max* es el máximo valor de AUC del grupo. Se indica con **negritas** el valor más alto de AUC en cada grupo y en ***negrita y cursiva*** para el valor de AUC más alto de todos las grupos; en este caso el valor más alto de AUC de todos los grupos fue del algoritmo **MARS** con 20000 instancias que obtuvo un valor de **0.94674.**

Tabla 13: Resultados de las métricas de evaluación en la experimentación por cantidad de instancias (registros) y algoritmo.

# Registros	***Algoritmo***	***AUC***	***AUC Min***	***AUC Max***	***Tiempo de entrenamiento (s)***	***Tiempo de prueba (s)***
1000	MARS	0.92960	0.79361	1.00000	0.26497	0.02815
	NB	0.92593	0.72058	1.00000	0.03070	0.53804
	NNET	0.82127	0.44818	0.99864	0.20522	0.03355
	SVM	**0.92979**	0.74331	0.99992	0.16664	0.02989
5000	MARS	**0.94512**	0.85081	0.99986	0.84141	0.05788
	NB	0.92808	0.75749	0.99968	0.05693	2.69002
	NNET	0.80837	0.45758	0.99950	0.71390	0.05396
	SVM	0.94059	0.82051	0.99987	2.23562	0.29622
10000	MARS	**0.94616**	0.84919	0.99996	1.69314	0.06822
	NB	0.92811	0.76494	0.99987	0.08998	5.50035

	NNET	0.76591	0.44697	0.99945	1.30306	0.08876
	SVM	0.94249	0.83194	0.99994	7.40691	0.92423
20000	***MARS***	***0.94674****	***0.85107****	***0.99973****	***4.82346****	***0.12202****
	NB	0.92782	0.76256	0.99967	0.15181	14.55225
	NNET	0.84488	0.45295	0.99930	3.52614	0.15427
	SVM	0.94366	0.84038	0.99972	26.73856	3.11420
50000	MARS	**0.94654**	0.85150	0.99930	223.41219	0.25693
	NB	0.92810	0.76930	0.99972	0.33900	44.89699
	NNET	0.77992	0.46588	0.99968	7.88749	0.38769
	SVM	0.94291	0.85352	0.99967	163.40858	13.87696
100000	MARS	**0.94559**	0.85449	0.99914	370.89620	0.60603
	NB	0.92817	0.77247	0.99966	0.76163	90.40042
	NNET	0.80511	0.44500	0.99952	17.69233	0.90155
	SVM	0.94322	0.85645	0.99970	746.45117	56.35608
200000	MARS	**0.94252**	0.81805	0.99923	267.33014	0.91002
	NB	0.92827	0.77197	0.99964	1.41923	170.55952
	NNET	0.79632	0.41546	0.99942	35.83422	1.90502
	SVM	0.94023	0.84771	0.99958	3027.11345	197.52814

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Una perspectiva más resumida se proporciona en la *Tabla 14*, donde se muestra las métricas de *AUC*, *Tiempo de entrenamiento* y *Tiempo de prueba*, donde se agrupan en función de los algoritmos y calculando el valor promedio de las métricas *AUC*, *Tiempo de entrenamiento* y *Tiempo de prueba*; obteniendo el valor de AUC mínimo del grupo *AUC Min* y el valor de AUC máximo *AUC Max*. Se indica con **negritas** el valor más alto de AUC, en este caso el obtenido por el algoritmo de MARS que fue de **0.94318.**

Tabla 14: Resultados de las métricas de evaluación en la experimentación por algoritmo.

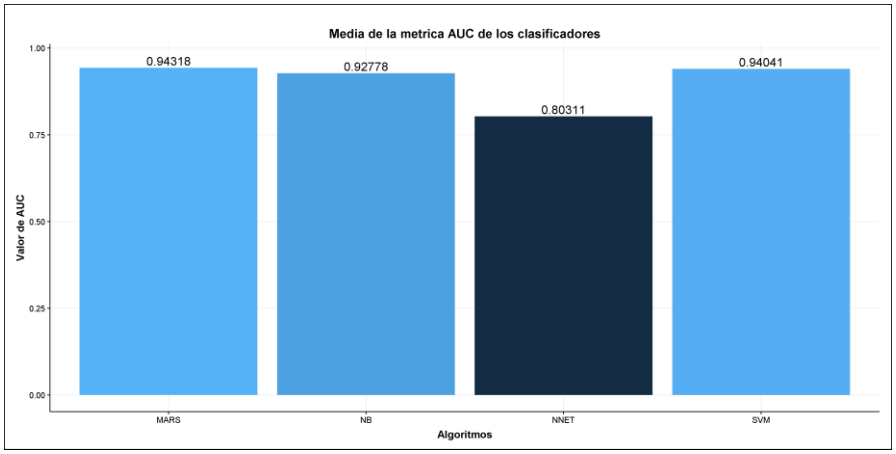
Algoritmo	***AUC***	***AUC Min***	***AUC Max***	***Tiempo de entrenamiento (s)***	***Tiempo de prueba (s)***
MARS	**0.94318**	**0.79361**	**1.00000**	**124.18022**	**0.29275**
NB	0.92778	0.72058	1.00000	0.40704	47.01965
NNET	0.80311	0.41546	0.99968	9.59462	0.50354
SVM	0.94041	0.74331	0.99994	567.64585	38.87510

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.3.2. Visualización de los resultados de la experimentación.

A partir de las métricas de AUC, tiempo de entrenamiento y tiempo de prueba de la *Tabla 14* se realizaron las siguientes ilustraciones:

Ilustración 42: Gráfica de barras de AUC por algoritmo.

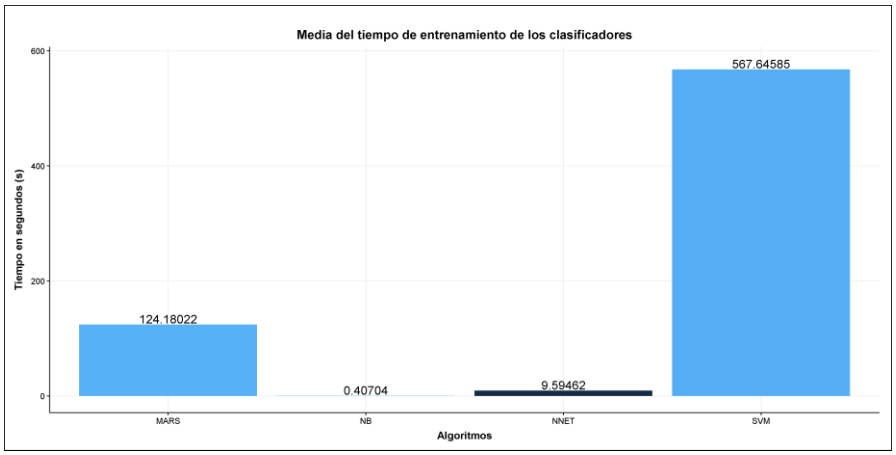


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

En la *Ilustración 42Ilustración 42* se puede apreciar la superioridad del valor de la métrica de AUC obtenida por los clasificadores del algoritmo MARS con una media de ***0.94318***, seguido por los clasificadores de SVM, NB y NNET.

El tiempo promedio empleado en el entrenamiento de los clasificadores se muestra en la *Ilustración 43*, donde el algoritmo NB es el que presenta el valor más óptimo con un tiempo de ***0.40704*** segundos (407 milisegundos). Siendo SVM el que mayor tiempo necesita, exactamente ***567.64585*** segundos (9.45 minutos).

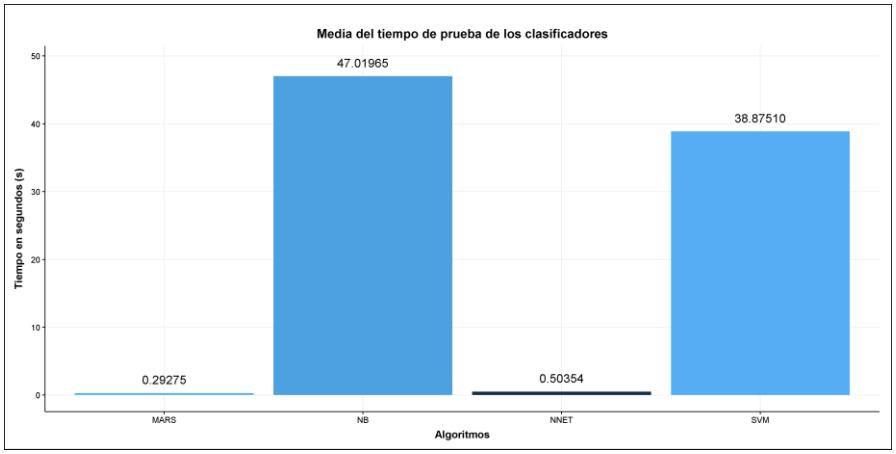
Ilustración 43: Gráfica de barras del tiempo de entrenamiento por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

El tiempo promedio empleado en las pruebas de los clasificadores se muestra en la gráfica *Ilustración 44*, donde MARS es el que presenta el más óptimo con un tiempo de ***0.29275*** segundos (293 milisegundos). Siendo NB el que mayor tiempo necesita, exactamente ***47.01965*** segundos.

Ilustración 44: Gráfica de barras del tiempo de prueba por algoritmo.

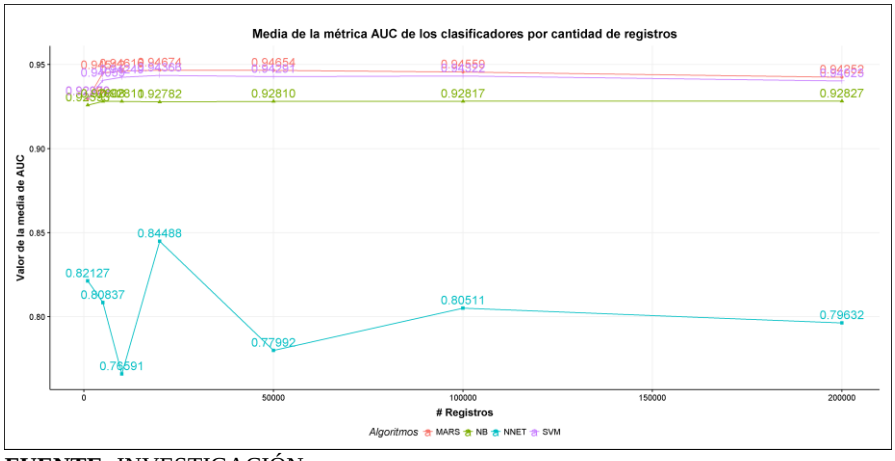


FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

A partir de la *Tabla 13* se realizaron las siguientes ilustraciones que muestran las diferencias en los valores de AUC, tiempo de entrenamiento y tiempo de prueba de los clasificadores cuando se aumentan la cantidad de instancias y la tasa de desbalanceo de las clases. Esa variación de instancias y la tasa de balanceo se encuentran en el Anexo 1.

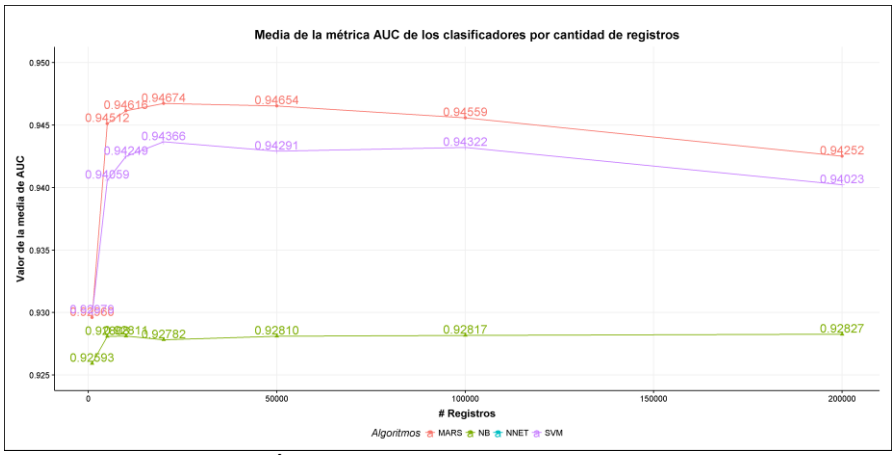
La *Ilustración 45* muestra que NNET presenta los valores más bajos de AUC comparados con los demás algoritmos y también se presenta una diferencia poco significativa de los valores AUC entre MARS y SVM. En la *Ilustración 46* se realiza filtrado de los valores, permitiendo tener una gráfica más detallada de los algoritmos MARS, SVM y NB.

Ilustración 45: Gráfica de líneas de AUC por cantidad de instancias y por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

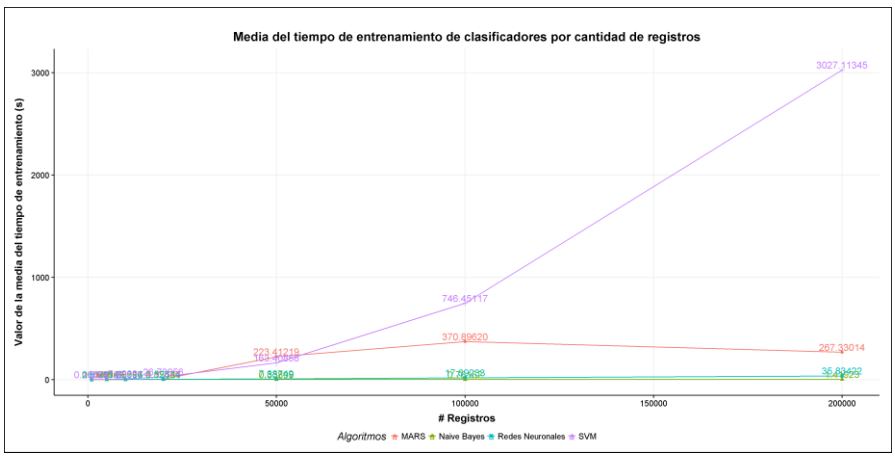
Ilustración 46: Gráfica de líneas filtrada de AUC por cantidad de instancias y algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

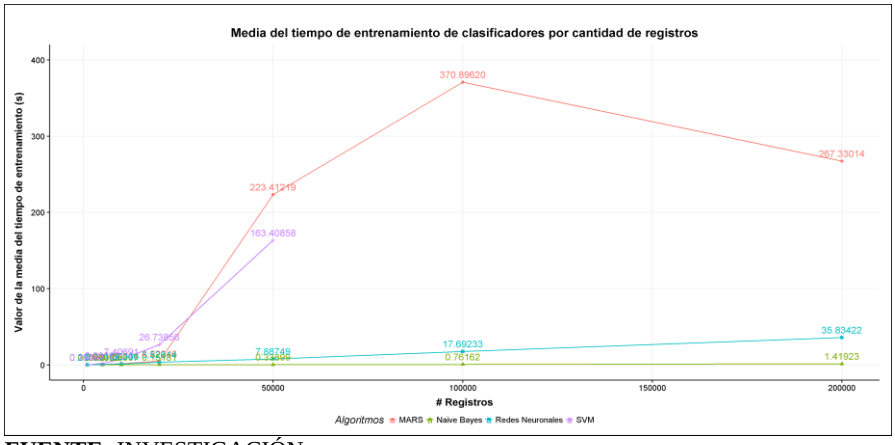
La *Ilustración 47* muestra el tiempo de entrenamiento de los clasificadores, donde el tiempo de entrenamiento que los clasificadores de SVM requieren es proporcional a la cantidad de instancias. En la *Ilustración 48* se realiza un filtrado de los algoritmos que mejor desempeño obtuvieron en esta métrica, en este caso fueron tres: NB, NNET y por ultimo MARS.

Ilustración 47: Gráfica de líneas del tiempo de entrenamiento por cantidad de instancias y por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

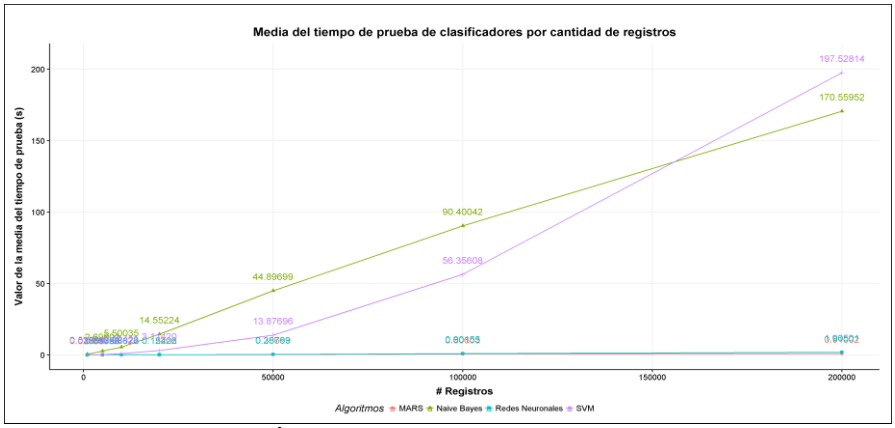
Ilustración 48: Gráfica de líneas filtrada del tiempo de entrenamiento por cantidad de instancias y por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

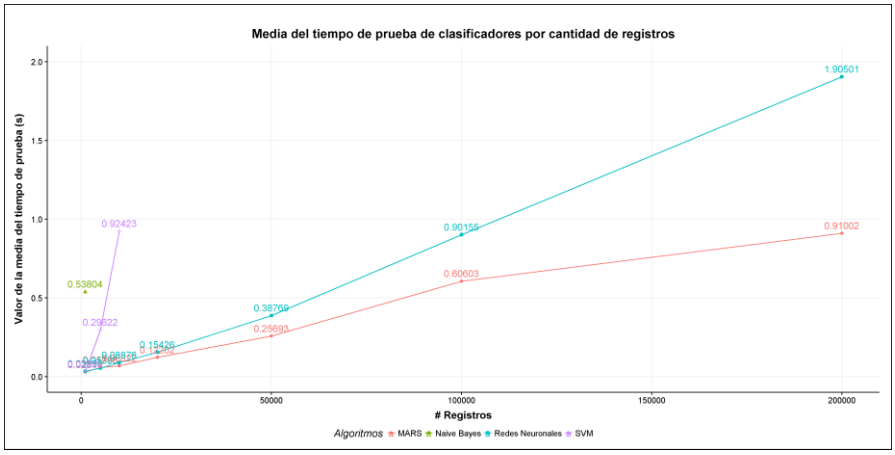
La *Ilustración 49* muestra el tiempo de prueba de los clasificadores, donde NB y SVM toman valores altos en cuanto a esta métrica. En la *Ilustración 50* se realiza un filtrado a los algoritmos que mejor desempeño obtuvieron en esta métrica, donde se destaca MARS por su bajo costo en tiempo computacional que necesita para realizar las predicciones, seguido por NNET.

Ilustración 49: Gráfica de líneas del tiempo de prueba por cantidad de instancias y por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

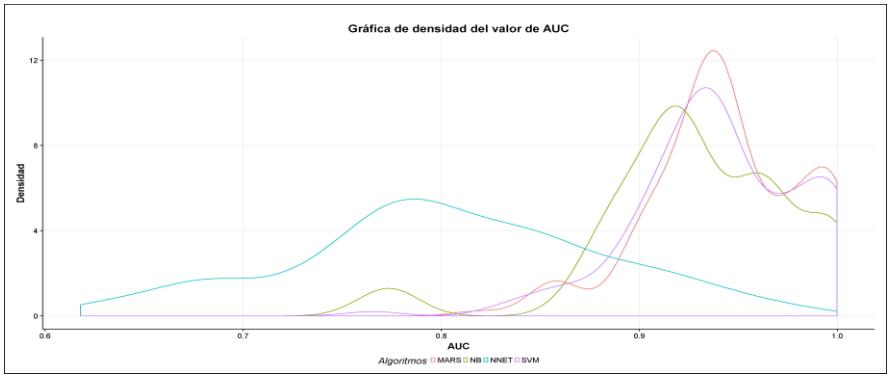
Ilustración 50: Gráfica de líneas filtrada del tiempo de prueba por cantidad de instancias y por algoritmo.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

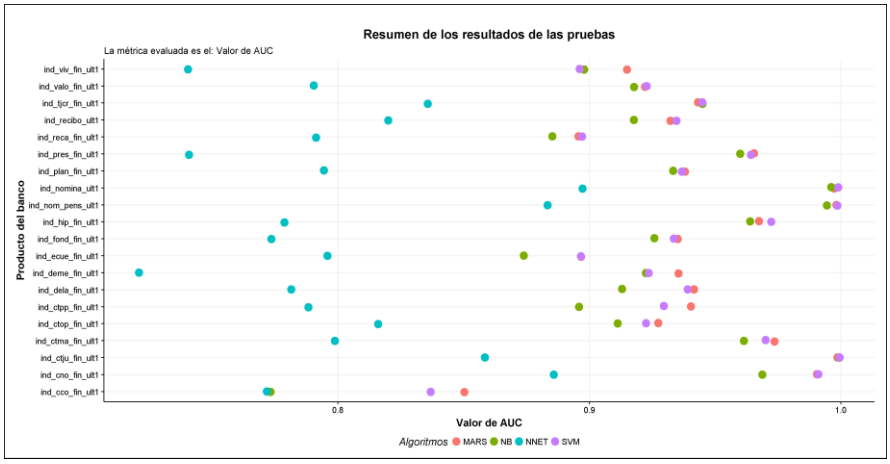
A partir de la tabla del Anexo 2 se realizaron las siguientes gráficas comparativas de las métricas de AUC, tiempo de entrenamiento y tiempo de prueba de los clasificadores construidos por los algoritmos.

Ilustración 51: Gráfica de densidad del valor de AUC en función de los algoritmos.



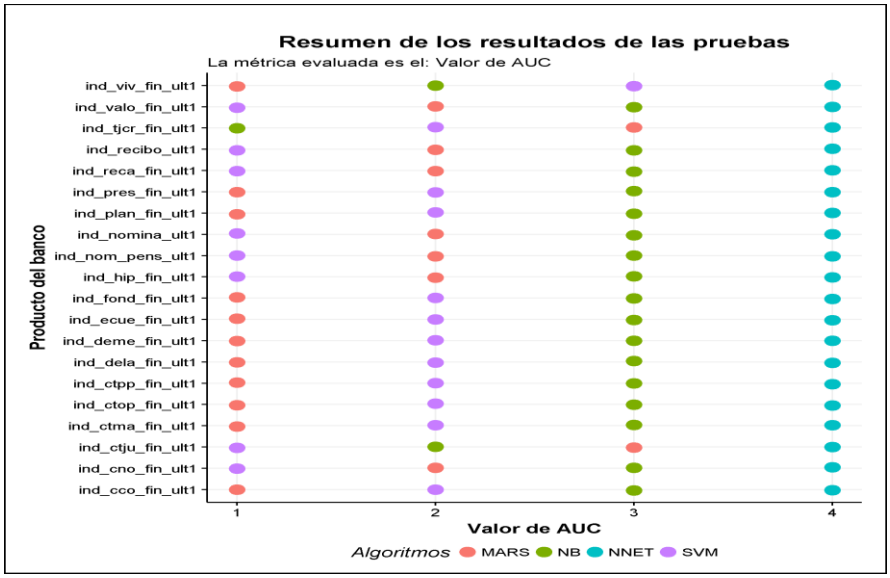
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 52: Gráfica del valor medio de AUC en función de los algoritmos y los productos del banco.



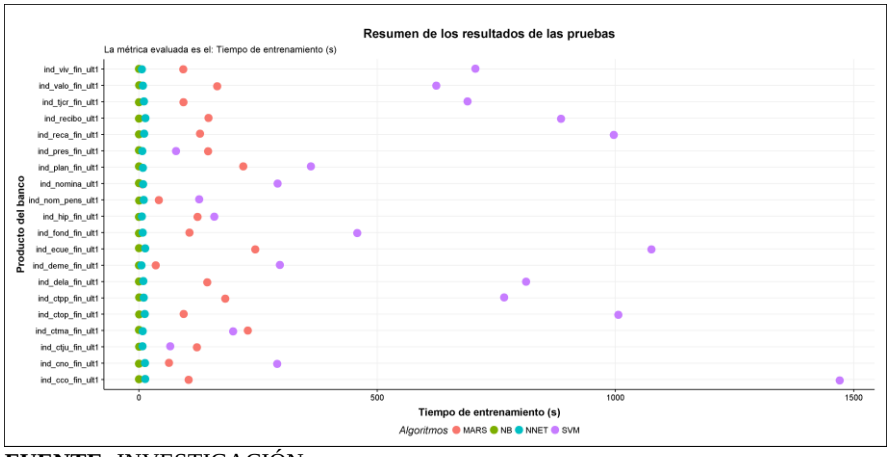
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 53: Gráfica de posiciones del valor medio de AUC en función de los algoritmos y los productos del banco.



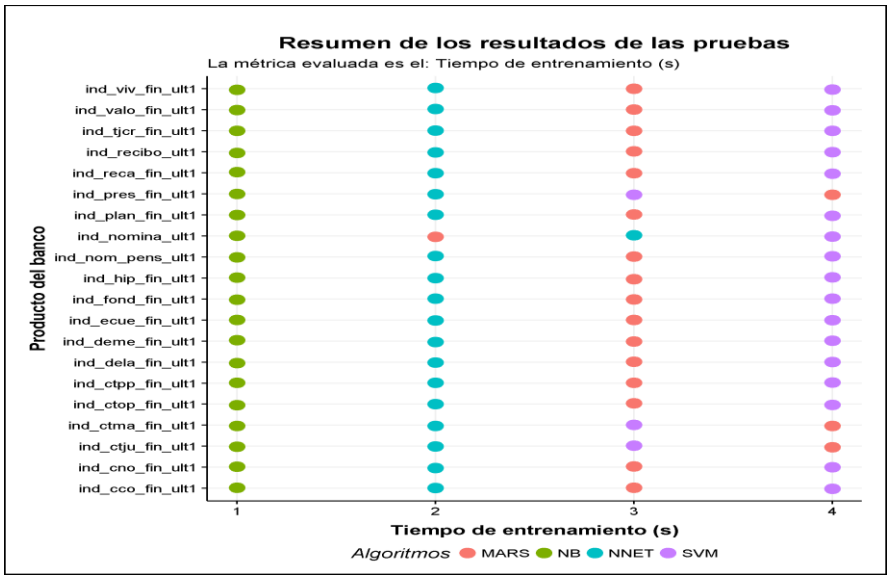
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 54: Gráfica del tiempo de entrenamiento en función de los algoritmos y los productos del banco.



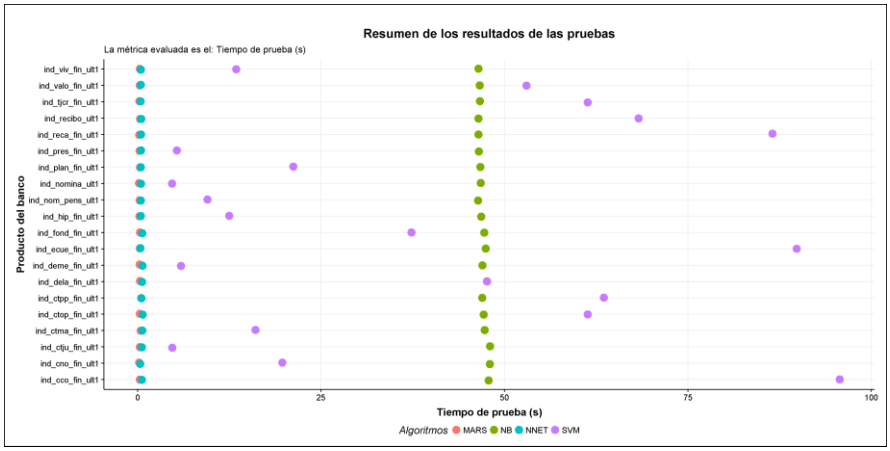
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 55: Gráfica de posiciones del tiempo de entrenamiento en función de los algoritmos y los productos del banco.



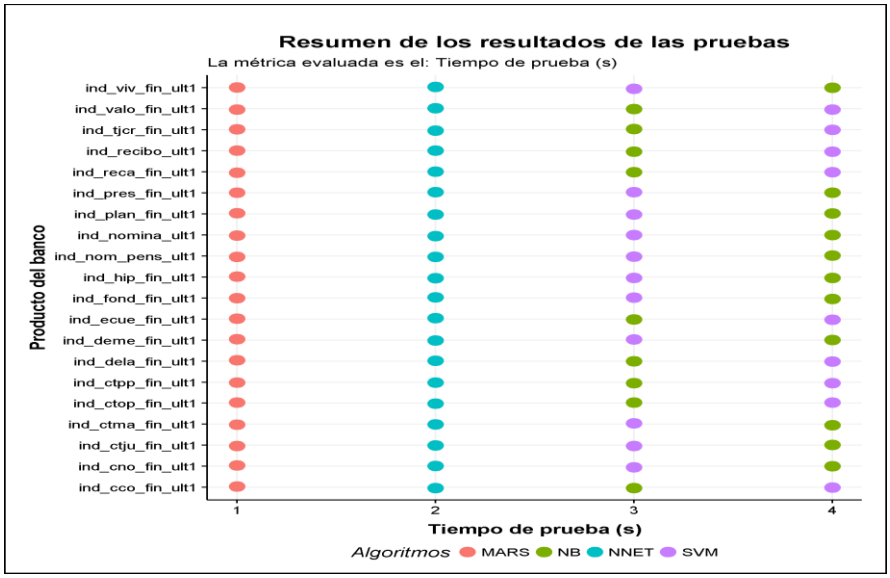
FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 56: Gráfica del tiempo de prueba en función de los algoritmos y los productos del banco.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 57: Gráfica de posiciones del tiempo de prueba en función de los algoritmos y los productos del banco.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.3.3.Comparación estadística de los clasificadores.

En busca de detectar diferencias estadísticamente significativas entre el comportamiento de los algoritmos se utilizó las pruebas de hipótesis de Friedman y post hoc Friedman-Nemenyi mediante las funciones *friedman.test* y *PMCMR::posthoc.friedman.nemenyi.test* respectivamente. La prueba de Friedman se aplicó a los datos contenidos en la tabla del Anexo 2 y de ser necesario se aplicaría la prueba post hoc Friedman-Nemenyi a estos mismos datos para detectar pares de algoritmos con diferencias significativas. A continuación, los resultados de las pruebas con respecto a los valores de AUC, tiempo de entrenamiento y tiempo de prueba:

4.6.3.3.1. Comparación de AUC.

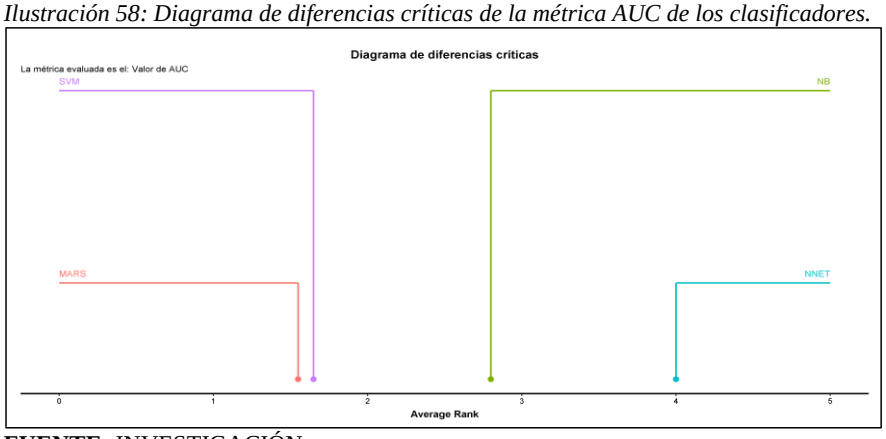
Se encontró que las posiciones promedio de con respecto al valor de la métrica **AUC** de los clasificadores son estadísticamente significativos, con un valor de $X_F^2 = 47.58$, un nivel de significación de 0.05 y un valor de $p = 2.616\text{E} - 10$. Por lo que procedió a aplicar la prueba post hoc Friedman-Nemenyi obteniendo un valor de diferencia crítica de 1.048803 y los valores individuales de p para los pares de clasificadores, se muestra en la *Tabla 15*. Tabla que indica que con un nivel de significancia de 0.01 los clasificadores de NNET presentan altas diferencias significativas con los clasificadores de MARS y SVM; en cambio los clasificadores de NB con un nivel de significancia de 0.05 difieren significativamente de MARS y SVM; y por último los clasificadores de SVM y MARS no presentan diferencias significativas. En la *Ilustración 58* se encuentra el diagrama de diferencias críticas entre valores de AUC de los clasificadores.

Tabla 15: Valores de "p" entre clasificadores individuales (AUC).

	MARS	***NB***	***NNET***
NB	0.012	-	-
NNET	1.2E-08	0.017	-
SVM	0.995	0.025	5.20E-08

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 58: Diagrama de diferencias críticas de la métrica AUC de los clasificadores.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.3.3.2. Comparación del tiempo de entrenamiento.

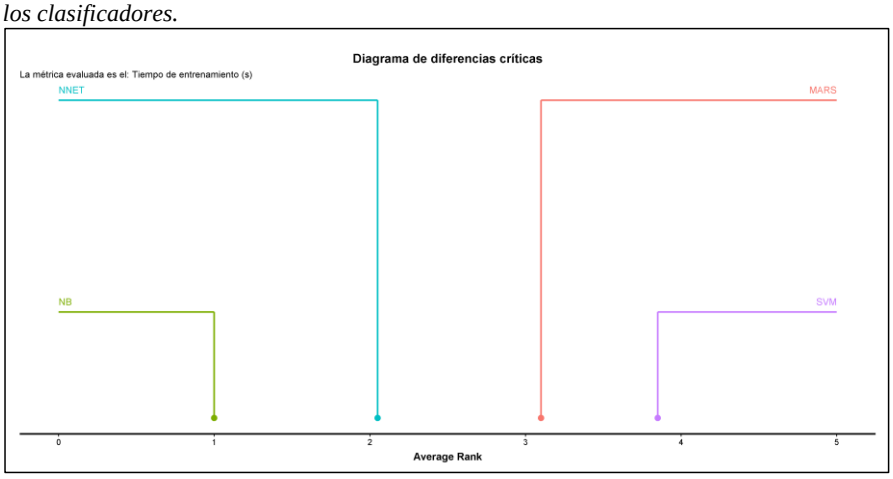
Se encontró que las posiciones promedio de con respecto al valor del **tiempo de entrenamiento (s)** de los clasificadores son estadísticamente significativos, con un valor de $X_F^2 = 55.62$, un nivel de significación de 0.05 y un valor de $p = 5.063\text{E} - 12$. Por lo que procedió a aplicar la prueba post hoc Friedman-Nemenyi obteniendo un valor de diferencia crítica de 1.048803 y los valores individuales de p para los pares de clasificadores, se muestra en la *Tabla 16*. Tabla donde muestra que con un nivel de significancia de 0.01 los clasificadores de NB presentan altas diferencias significativas con los clasificadores de MARS y SVM; al igual que los clasificadores de NNET presentan altas diferencias significativas los clasificadores de SVM; en cambio los clasificadores de NNET con un nivel de significancia de 0.05 difieren significativamente de MARS y NB; y por último los clasificadores de SVM y MARS no presentan diferencias significativas. En la *Ilustración 59* se encuentra el diagrama de diferencias críticas entre valores del tiempo de entrenamiento de los clasificadores.

Tabla 16: Valores de "p" entre clasificadores individuales (tiempo de entrenamiento).

	MARS	***NB***	***NNET***
NB	1.6E-06	-	-
NNET	0.05	0.05	-
SVM	0.26	1.8E-11	6.1E-05

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 59: Diagrama de diferencias críticas de la métrica tiempo de entrenamiento de los clasificadores.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.6.3.3.3. Comparación del tiempo de prueba.

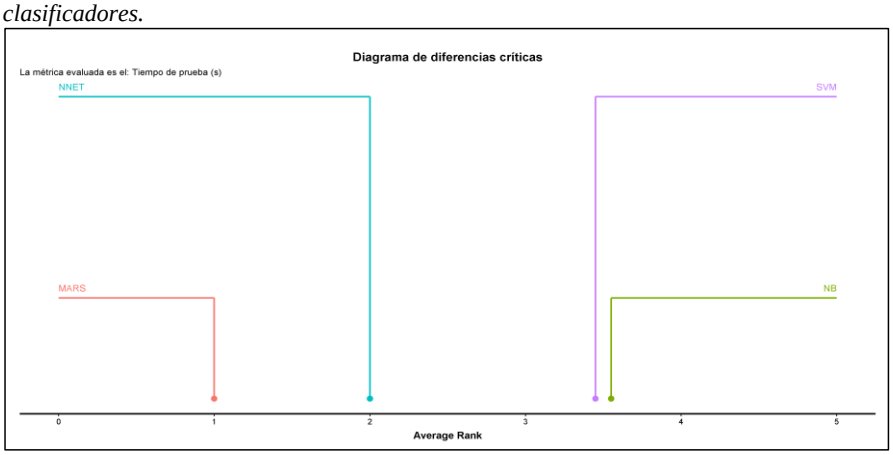
Se encontró que las posiciones promedio de con respecto al valor del **tiempo de prueba (s)** de los clasificadores son estadísticamente significativos, con un valor de $X_F^2 = 54.06$, un nivel de significación de 0.05 y un valor de $p = 1.089E - 11$. Por lo que procedió a aplicar la prueba post hoc Friedman-Nemenyi obteniendo un valor de diferencia crítica de 1.048803 y los valores individuales de p para los pares de clasificadores, se muestra en la *Tabla 17*. Tabla donde muestra que con un nivel de significancia de 0.01 los clasificadores de MARS presentan altas diferencias significativas con los clasificadores de NB y SVM; al igual que los clasificadores de NNET presentan altas diferencias significativas los clasificadores de SVM y NB; y por último no se presentan diferencias significativas entre los clasificadores de SVM y MARS, y entre los clasificadores de SVM y NB. En la *Ilustración 60* se encuentra el diagrama de diferencias críticas entre valores del tiempo de entrenamiento de los clasificadores.

Tabla 17: Valores de "p" entre clasificadores individuales (tiempo de prueba).

	MARS	***NB***	***NNET***
NB	2.5E-09	-	-
NNET	0.06812	0.00084	-
SVM	1.2E-08	0.99484	0.00217

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Ilustración 60: Diagrama de diferencias críticas de la métrica tiempo de prueba de los clasificadores.



FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

4.7. Discusión.

Las características del servidor utilizado permitieron disminuir el tiempo de prueba debido a su arquitectura multiprocesador, la cual fue aprovechada eficientemente en las funciones del paquete de MLR ya que permitió ejecutar en paralelo varios procesos. También es destacable que aunque la distribución de R llamada Microsoft R Open tiene una interfaz por línea de comandos (CLI), esta y la distribución oficial de R permiten la ejecución de scripts lo que permitió hacer reproducible el proceso de minería de datos de la presente investigación.

En el marco referencial se presentan algunas investigaciones que utilizan la clasificación para generar recomendaciones [12] [50] [25]. Por ejemplo en el artículo [25] comparan el desempeño de los clasificadores utilizando la métrica de recuerdo (Recall), la cual es un medio de comparación menos efectivo que el uso de la métrica de AUC o las pruebas estadísticas de mejoras de desempeño como la prueba de Friedman y post hoc Friedman-Nemenyi las cuales son utilizadas en esta investigación.

Es importante resaltar que dos investigaciones para el marketing directo de un banco se realizaron a partir de un mismo conjunto de datos [13], [49]; las cuales mediante la métrica de AUC se encontró que los mejores resultados fueron de los clasificadores proporcionados por los algoritmos de clasificación de SVM y NNET; respectivamente. En contraste con la presente investigación, donde se encontró que MARS con un valor medio de AUC de **0.94318** tiene mejor desempeño que los valores obtenidos por SVM y NNET, cuyos valores de AUC fueron de **0.94041** y **0.80311** respectivamente.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

La presente investigación se realizó para determinar un algoritmo de clasificación supervisada que sea apropiado para generar recomendaciones de productos bancarios. Al finalizar esta investigación se concluye lo siguiente:

- ✓ La aplicación de las técnicas preprocesamiento permitió reducir las dimensionalidad del conjunto datos de 48 a 38 atributos y de 13’647.309 a 931.453 instancias. Esta reducción se realizó mediante la eliminación de atributos con información duplicada y atributos clase altamente desbalanceados. Además que la reducción de categorías de los atributos *fecha_alta*, *canal_entrada*, *pais_residencia* y *cod_prov*, permitió disminuir la complejidad de los clasificadores construidos a partir del conjunto datos preprocesado.

- ✓ Aunque, en investigaciones similares a esta, los algoritmos basados en árboles de decisión fueron los que mejor desempeño obtuvieron, en esta investigación realizada no fue así. Y fueron cuatro los algoritmos que se seleccionaron ya que se comprobó su aplicabilidad en el conjunto de datos, los cuales fueron *NNET, NB, SVM y MARS*.

- ✓ Los clasificadores construidos por *MARS* mostraron una superioridad en los valores de AUC comparados con los clasificadores de *NNET*, *NB* y *SVM*, aunque esta superioridad es pequeña con respecto a *SVM*, esta aumenta con respecto a *NB* y es aún más notoria con respecto a *NNET*.

- ✓ Al comparar los valores de las métricas del tiempo de entrenamiento y de prueba se encontró, que SVM y NB en la construcción de sus clasificadores necesitan más tiempo; y que MARS es el que menor tiempo de prueba necesita. Por lo que, en cuanto a estas dos métricas los clasificadores de mejor desempeño fueron los construidos por el algoritmo de MARS y los de menor desempeño por el algoritmo de SVM.

- ✓ Aplicando las pruebas estadísticas no paramétricas de Friedman y post hoc Friedman-Nemenyi a los valores de la métrica de AUC, del tiempo de entrenamiento

y de prueba se encontró que: con respecto a AUC existen diferencias estadísticamente significativas, entre todos los pares de algoritmos con excepción del par conformado por *SVM* y *MARS*; en cuanto con respecto a la métrica del tiempo de entrenamiento se encontró que difieren significativamente todos los pares de algoritmos con excepción del par conformado por *SVM* y *MARS*; y finalmente con respecto a la métrica del tiempo de prueba se encontró que difieren significativamente todos los pares de algoritmos con excepción de los pares (*SVM* y *NB*) y (*MARS* y *NNET*).

Por lo que finalmente se concluye que el algoritmo de *MARS* por la característica del tipo de clasificador que proporciona (modelo aditivo de funciones lineales del tipo polinomial), por su desempeño en cuanto a las métricas evaluadas (AUC, tiempo de entrenamiento y de prueba) y por su aplicabilidad en este conjunto de datos de alta dimensionalidad; es el más apropiado para generar recomendaciones de productos bancarios mediante sus clasificadores.

5.2. Recomendaciones.

La presente investigación contiene información suficiente para poder llevar a cabo una implementación en un ámbito real de algún banco de la región, así como realizar otras investigaciones que estudien el efecto de las técnicas de preprocesamiento lo cual podría complementarla a esta presente investigación, pero como es de conocimiento siempre existen requerimientos específicos por lo que se recomienda:

- De acuerdo al tipo de enfoque con que el problema de clasificación se quiera solucionar, se necesitará evaluar que métrica de desempeño será la más adecuada.

- De ser posible realizar una optimización de los hiperparámetros (parámetros de configuración) de los algoritmos mediante técnicas de tuning.

- Utilizar reportes generados por los paquetes de R llamados knitr y pandoc.

- En caso de realizar un proyecto de minería de datos a partir de esta investigación, se recomienda hacerlo reproducible y llevarlo a cabo mediante metodología de CRISP-DM.

CAPITULO VI

BIBLIOGRAFÍA

6.1. Bibliografía

[1] M. Rojas and J. . Rojas, "Comportamiento Estratégico en la Industria Bancaria," *Inversión y Finanz.*, vol. 3, no. 2, 1995.

[2] X. Hu, "A data mining approach for retailing bank customer attrition analysis," *Appl. Intell.*, vol. 22, no. 1, pp. 47–60, 2005.

[3] H. Jiawei, M. Kamber, J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. 2012.

[4] F. Ricci, L. Rokach, B. Shapira, and K. Paul B., *Recommender Systems Handbook*, vol. 532. 2011.

[5] M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: A fast scalable classifier for data mining," in *Advances in Database Technology --- EDBT '96: 5th International Conference on Extending Database Technology Avignon, France, March 25--29, 1996 Proceedings*, P. Apers, M. Bouzeghoub, and G. Gardarin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 18–32.

[6] J. Adams, "Georgia Bank Uses Sophisticated Algorithm for Product Recommendations," 2012. [Online]. Available: https://www.americanbanker.com/news/georgia-bank-uses-sophisticated-algorithm-for-product-recommendations?feed=00000158-bab1-dda9-adfa-fef7649f0000. [Accessed: 07-Jul-2017].

[7] S. R. Dash and S. Dehuri, "Comparative Study of Different Classification Techniques for Post Operative Patient Dataset," *Int. J. Innov. Res. Comput. Communucation Eng.*, vol. 1, no. 5, pp. 1101–1108, 2013.

[8] Sakshi and P. S. Khare, "A Comparative Analysis of Classification Techniques on Categorical Data in Data Mining," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 3, no. 8, pp. 5142–5147, 2015.

[9] S. S. Nikam, "A Comparative Study of Classification Techniques in Data Mining Algorithms," *Orient. J. Comput. Sci. Technol.*, vol. 8, no. 1, pp. 13–19, 2015.

[10] T. Smitha and V. Sundaram, "Comparative Study of Data Mining Algorithms for High Dimensional Data Analysis," *Int. J. Adv. Eng. Technol.*, vol. 4, no. 2, pp. 173–178, 2012.

[11] C. C. Aggarwal, *Data mining : the textbook*, 1st ed. Cham: Springer International Publishing, 2015.

[12] C. Basu, H. Hirsh, and W. Cohen, "Recommendation As Classification: Using Social and Content-based Information in Recommendation," in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, 1998, pp. 714–720.

[13] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decis. Support Syst.*, vol. 62, pp. 22–31, 2014.

[14] H. A. Elsalamony and A. M. Elsayad, "Bank Direct Marketing Based on Neural Network," *Int. J. Eng. Adv. Technol.*, vol. 2, no. 6, pp. 392–400, 2013.

[15] H. A. Elsalamony, "Bank Direct Marketing Analysis of Data Mining Techniques," *Int. J. Comput. Appl.*, vol. 85, no. 7, pp. 12–22, 2014.

[16] S. PANG and J. GONG, "C5.0 Classification Algorithm and Application on Individual Credit Evaluation of Banks," *Syst. Eng. - Theory Pract.*, vol. 29, no. 12, pp. 94–104, 2009.

[17] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 261–273, 2015.

[18] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, 1st ed., vol. 72. Springer International Publishing, 2015.
[19] M. Bramer, *Principles of Data Mining*, 3rd ed. Springer-Verlag London, 2016.
[20] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.
[21] Two Crows Corporation, *Introduction to Data mining and Knowledge discovery*, 3rd ed. Two Crows Corporation, 2006.
[22] R. Wirth and J. Hipp, "CRISP-DM : Towards a Standard Process Model for Data Mining," *Proc. Fourth Int. Conf. Pract. Appl. Knowl. Discov. Data Min.*, no. 24959, pp. 29–39, 2000.
[23] F. Herrera, F. Charte, A. J. Rivera, and M. J. Del Jesus, *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer International Publishing, 2016.
[24] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.
[25] L. Schmidt-Thieme, "Compound classification models for recommender systems," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005, p. 8 pp.-pp.
[26] L. Torgo, *Data Mining with R: Learning with Case Studies, Second Edition*, Second edi. Taylor & Francis;Chapman and Hall/CRC, 2017.
[27] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.
[28] J. Vanegas and F. Vásquez, "Multivariate Adaptative Regression Splines (MARS), una alternativa para el análisis de series de tiempo," *Gac. Sanit.*, vol. 31, no. 3, pp. 235–237, 2017.
[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.*, 2nd ed. Springer-Verlag New York, 2009.
[30] T. Borovicka, M. J. Jr., P. Kordik, and M. Jirina, "Selecting Representative Data Sets," in *Advances in Data Mining Knowledge Discovery and Applications*, A. Karahoca, Ed. Rijeka: InTech, 2012.
[31] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Comput.*, vol. 10, pp. 1895–1923, 1998.
[32] A. Eniafe Festus and A. B. Adeyemo, "A Data Mining-Based Response Model for Target Selection in Direct Marketing," *Int. J. Inf. Technol. Comput. Sci.*, vol. 4, no. 1, pp. 9–18, 2012.
[33] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.
[34] "Curva ROC," 2016. [Online]. Available: https://es.wikipedia.org/wiki/Curva_ROC. [Accessed: 01-Dec-2016].
[35] C. Sibona and J. Brickey, "A Statistical Comparison of Classification Algorithms on a Single Data Set," *AMCIS 2012 Proc.*, vol. 2, 2012.
[36] M. A. Maloof, *Machine Learning and Data Mining for Computer Security*, First. Springer-Verlag London, 2006.
[37] I. Brown and C. Mues, "An experimental comparison of classification algorithms for imbalanced credit scoring data sets," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3446–3453, 2012.
[38] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
[39] A. Farhangfar, L. A. Kurgan, and W. Pedrycz, "A Novel Framework for Imputation of Missing Values in Databases," *Trans. Sys. Man Cyber. Part A*, vol. 37, no. 5, pp.

692–709, 2007.
[40] E. Acuña and C. Rodriguez, “The Treatment of Missing Values and its Effect on Classifier Accuracy,” in *Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15--18 July 2004*, D. Banks, F. R. McMorris, P. Arabie, and W. Gaul, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 639–647.
[41] P. Liu and L. Lei, “Missing Data Treatment Methods and NBI Model,” in *Sixth International Conference on Intelligent Systems Design and Applications*, 2006, vol. 1, pp. 633–638.
[42] C. Sammut and W. Geoffrey I, Eds., *Encyclopedia of Machine Learning*. Springer US, 2010.
[43] C. X. Ling and C. Li, “Data Mining for Direct Marketing: Problems and Solutions,” in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 73–79.
[44] A. Jaramilloa and H. Paz-Ariasb, “Aplicación de Técnicas de Minería de Datos para Determinar las Interacciones de los Estudiantes en un Entorno Virtual de Aprendizaje,” *Rev. Tecnológica ESPOL*, vol. 28, no. 1, pp. 64–90, Aug. 2015.
[45] P. Chapman *et al.*, “Crisp-dm 1.0: Step-by-step data mining guide.” SPSS, p. 78, 2000.
[46] R. Qamar Parvez and K. Parminder, “Comparison of Various Tools for Data Mining,” *Int. J. Eng. Res. Technol.*, vol. 3, no. 10, pp. 393–397, 2014.
[47] “Microsoft R Application Network,” 2017. [Online]. Available: https://mran.microsoft.com/rro/. [Accessed: 19-Apr-2017].
[48] H. Wu and M. Guan, “Segmentation of the Bank Client Value Based on Fuzzy Data Mining,” in *Proceedings of 2012 3rd International Asia Conference on Industrial Engineering and Management Innovation (IEMI2012)*, R. Dou, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 569–582.
[49] S. Moro, R. M. S. Laureano, and P. Cortez, “Using Data Mining for Bank Direct Marketing : An Application of the CRISP-DM Methodology,” 2011.
[50] H. Tan and H. Wu Ye, “A collaborative filtering recommendation algorithm based on item classification,” *Proc. 2009 Pacific-Asia Conf. Circuits, Commun. Syst. PACCS 2009*, pp. 694–697, 2009.
[51] Kaggle Inc, “Kaggle - Your Home for Data Science,” 2016. [Online]. Available: https://www.kaggle.com/. [Accessed: 22-Nov-2016].
[52] A. Gupta, K. G. Mehrotra, and C. Mohan, “A clustering-based discretization for supervised learning,” *Stat. Probab. Lett.*, vol. 80, no. 9–10, pp. 816–824, 2010.
[53] S. Ray, “Simple Methods to deal with Categorical Variables in Predictive Modeling,” 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/. [Accessed: 03-Apr-2017].
[54] “Países del Mundo.” [Online]. Available: http://country-code.cl/es/index.htm. [Accessed: 04-Apr-2017].
[55] Wikipedia, “Provincias y ciudades autónomas de España,” 2017. [Online]. Available: https://es.wikipedia.org/wiki/Anexo:Provincias_y_ciudades_autónomas_de_España. [Accessed: 04-Apr-2017].
[56] B. Bischl *et al.*, “mlr: Machine Learning in R,” *J. Mach. Learn. Res.*, vol. 17, no. 170, pp. 1–5, 2016.
[57] R.-M. Tefan, “Emerging Markets Queries in Finance and Business A Comparison of Data Classification Methods,” *Procedia - Soc. Behav. Sci.*, vol. 3, no. 12, pp. 420–425, 2012.

[58] R. Muñoz-Mas, S. Fukuda, P. Vezza, and F. Martínez-Capel, “Comparing four methods for decision-tree induction: A case study on the invasive Iberian gudgeon (Gobio lozanoi; Doadrio and Madeira, 2004),” *Ecol. Inform.*, vol. 34, pp. 22–34, 2016.
[59] L. Torgo, *Data Mining with R: Learning with Case Studies*, 1st ed. Chapman & Hall/CRC, 2010.
[60] J. H. Friedman, “Multivariate Adaptive Regression Splines,” *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.
[61] “Studentized range,” 2017. [Online]. Available: https://en.wikipedia.org/wiki/Studentized_range. [Accessed: 12-Jun-2017].

CAPITULO VII

ANEXOS

Anexo 1: Tabla del porcentaje de instancias con clases positivas (P) y negativas (N) dentro de los conjuntos de datos usados en la sexta fase.

Atributo	#1000		#5000		#10000		#20000		#50000		#100000		#200000	
	P	N	P	N	P	N	P	N	P	N	P	N	P	N
ind_cco_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
ind_cno_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	36.53	63.47
ind_ctju_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	37.91	62.10	15.16	84.84	7.58	92.42	3.79	96.21
ind_ctma_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	40.49	59.52	16.19	83.81	8.10	91.90	4.05	95.95
ind_ctop_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
ind_ctpp_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	33.27	66.73	16.64	83.36
ind_deme_fin_ult1	50.00	50.00	20.42	79.58	10.21	89.79	5.11	94.90	2.04	97.96	1.02	98.98	0.51	99.49
ind_dela_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	31.22	68.78	15.61	84.39
ind_ecue_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	37.92	62.08
ind_fond_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	29.37	70.63	14.69	85.31	7.34	92.66
ind_hip_fin_ult1	50.00	50.00	50.00	50.00	45.28	54.72	22.64	77.36	9.06	90.94	4.53	95.47	2.26	97.74
ind_plan_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	36.80	63.21	14.72	85.28	7.36	92.64	3.68	96.32
ind_pres_fin_ult1	50.00	50.00	39.82	60.18	19.91	80.09	9.96	90.05	3.98	96.02	1.99	98.01	1.00	99.00
ind_reca_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	45.54	54.47	22.77	77.23
ind_tjcr_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	34.82	65.18	17.41	82.59
ind_valo_fin_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	42.85	57.15	21.43	78.57	10.71	89.29
ind_viv_fin_ult1	50.00	50.00	50.00	50.00	29.59	70.41	14.80	85.21	5.92	94.08	2.96	97.04	1.48	98.52
ind_nomina_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	48.31	51.69	24.15	75.85
ind_nom_pens_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	26.42	73.58
ind_recibo_ult1	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Anexo 2: Tabla de los resultados del proceso de experimentación de la sexta fase.

task.id	*learner.id*	*auc*	*auc.min*	*auc.max*	*timetrain*	*timepredict*
ind_cco_fin_ult1	MARS	0.85028	0.79361	0.86244	104.596	0.296
ind_cco_fin_ult1	NB	0.77315	0.72058	0.80516	0.365	47.824
ind_cco_fin_ult1	NNET	0.77164	0.47517	0.87575	13.142	0.567
ind_cco_fin_ult1	SVM	0.83685	0.74331	0.86399	1470.317	95.684
ind_cno_fin_ult1	MARS	0.99045	0.97282	0.99457	63.227	0.211
ind_cno_fin_ult1	NB	0.96876	0.95892	0.97581	0.429	48.012
ind_cno_fin_ult1	NNET	0.88577	0.47555	0.99256	12.947	0.355
ind_cno_fin_ult1	SVM	0.99103	0.98133	0.99484	290.357	19.727
ind_ctju_fin_ult1	MARS	0.99861	0.99117	0.99996	121.888	0.297
ind_ctju_fin_ult1	NB	0.99941	0.99835	1.00000	0.505	48.040
ind_ctju_fin_ult1	NNET	0.85838	0.46783	0.99968	7.755	0.576
ind_ctju_fin_ult1	SVM	0.99951	0.99885	0.99994	65.731	4.709
ind_ctma_fin_ult1	MARS	0.97360	0.93510	0.98290	228.549	0.407
ind_ctma_fin_ult1	NB	0.96137	0.94314	0.97080	0.423	47.303
ind_ctma_fin_ult1	NNET	0.79875	0.41546	0.97870	7.925	0.630
ind_ctma_fin_ult1	SVM	0.97005	0.93614	0.97672	197.771	16.069
ind_ctop_fin_ult1	MARS	0.92736	0.87486	0.93718	93.857	0.283
ind_ctop_fin_ult1	NB	0.91119	0.89099	0.92558	0.509	47.165
ind_ctop_fin_ult1	NNET	0.81596	0.44697	0.93770	12.545	0.688
ind_ctop_fin_ult1	SVM	0.92252	0.89690	0.93292	1005.723	61.329
ind_ctpp_fin_ult1	MARS	0.94031	0.90971	0.94806	181.069	0.486
ind_ctpp_fin_ult1	NB	0.89581	0.87996	0.93099	0.518	46.955
ind_ctpp_fin_ult1	NNET	0.78825	0.45180	0.94181	10.291	0.513
ind_ctpp_fin_ult1	SVM	0.92956	0.87247	0.93861	766.217	63.534
ind_dela_fin_ult1	MARS	0.94158	0.89806	0.94965	143.426	0.315
ind_dela_fin_ult1	NB	0.91297	0.89465	0.92221	0.401	47.597
ind_dela_fin_ult1	NNET	0.78139	0.46539	0.95425	9.424	0.595
ind_dela_fin_ult1	SVM	0.93903	0.90292	0.95268	812.197	47.604
ind_deme_fin_ult1	MARS	0.93538	0.81805	0.95327	35.180	0.276
ind_deme_fin_ult1	NB	0.92237	0.88347	0.94149	0.308	46.991
ind_deme_fin_ult1	NNET	0.72085	0.43756	0.94745	5.479	0.656
ind_deme_fin_ult1	SVM	0.92353	0.89142	0.94876	295.665	5.905
ind_ecue_fin_ult1	MARS	0.89665	0.84739	0.90770	243.920	0.298
ind_ecue_fin_ult1	NB	0.87388	0.85226	0.88934	0.382	47.431
ind_ecue_fin_ult1	NNET	0.79577	0.47367	0.91697	13.359	0.373
ind_ecue_fin_ult1	SVM	0.89657	0.86394	0.91518	1075.207	89.816
ind_fond_fin_ult1	MARS	0.93511	0.90631	0.95068	106.037	0.323
ind_fond_fin_ult1	NB	0.92576	0.91537	0.95082	0.392	47.241
ind_fond_fin_ult1	NNET	0.77351	0.44904	0.94125	8.060	0.630
ind_fond_fin_ult1	SVM	0.93351	0.92060	0.95634	458.249	37.317

ind_hip_fin_ult1	MARS	0.96743	0.93270	0.97683	123.199	0.253
ind_hip_fin_ult1	NB	0.96390	0.94870	0.97229	0.411	46.820
ind_hip_fin_ult1	NNET	0.77869	0.45798	0.97845	6.324	0.448
ind_hip_fin_ult1	SVM	0.97227	0.95378	0.97702	158.321	12.480
ind_nom_pens_ult1	MARS	0.99818	0.98760	0.99979	42.034	0.268
ind_nom_pens_ult1	NB	0.99439	0.99077	0.99677	0.467	46.383
ind_nom_pens_ult1	NNET	0.88335	0.47367	0.99912	10.208	0.437
ind_nom_pens_ult1	SVM	0.99858	0.99699	0.99925	126.551	9.505
ind_nomina_ult1	MARS	0.99738	0.99393	1.00000	7.398	0.189
ind_nomina_ult1	NB	0.99610	0.99438	0.99851	0.336	46.775
ind_nomina_ult1	NNET	0.89726	0.46882	0.99952	8.951	0.458
ind_nomina_ult1	SVM	0.99889	0.99698	0.99992	290.833	4.689
ind_plan_fin_ult1	MARS	0.93793	0.89637	0.94642	218.924	0.311
ind_plan_fin_ult1	NB	0.93331	0.91320	0.95312	0.376	46.708
ind_plan_fin_ult1	NNET	0.79436	0.46030	0.94675	8.379	0.437
ind_plan_fin_ult1	SVM	0.93677	0.91167	0.95136	360.806	21.220
ind_pres_fin_ult1	MARS	0.96548	0.94010	0.97674	145.143	0.265
ind_pres_fin_ult1	NB	0.95989	0.94815	0.96659	0.401	46.502
ind_pres_fin_ult1	NNET	0.74073	0.45375	0.97376	7.001	0.460
ind_pres_fin_ult1	SVM	0.96422	0.93560	0.97783	77.823	5.342
ind_reca_fin_ult1	MARS	0.89555	0.85646	0.91107	128.397	0.224
ind_reca_fin_ult1	NB	0.88519	0.86263	0.89595	0.371	46.457
ind_reca_fin_ult1	NNET	0.79129	0.45908	0.91120	11.037	0.470
ind_reca_fin_ult1	SVM	0.89709	0.86588	0.90711	996.359	86.504
ind_recibo_ult1	MARS	0.93223	0.88323	0.94497	145.931	0.313
ind_recibo_ult1	NB	0.91761	0.89054	0.92471	0.333	46.455
ind_recibo_ult1	NNET	0.81994	0.47465	0.94284	13.481	0.482
ind_recibo_ult1	SVM	0.93456	0.89890	0.94188	885.844	68.289
ind_tjcr_fin_ult1	MARS	0.94313	0.92228	0.94880	93.465	0.242
ind_tjcr_fin_ult1	NB	0.94493	0.93366	0.95336	0.327	46.654
ind_tjcr_fin_ult1	NNET	0.83571	0.45421	0.95059	10.679	0.404
ind_tjcr_fin_ult1	SVM	0.94487	0.93371	0.95404	689.306	61.329
ind_valo_fin_ult1	MARS	0.92207	0.88316	0.93246	164.134	0.303
ind_valo_fin_ult1	NB	0.91773	0.89166	0.93267	0.476	46.619
ind_valo_fin_ult1	NNET	0.79036	0.45901	0.93124	8.585	0.450
ind_valo_fin_ult1	SVM	0.92274	0.89669	0.93185	623.713	53.004
ind_viv_fin_ult1	MARS	0.91492	0.88902	0.94333	93.231	0.294
ind_viv_fin_ult1	NB	0.89793	0.88659	0.93342	0.411	46.460
ind_viv_fin_ult1	NNET	0.74029	0.45378	0.92592	6.321	0.441
ind_viv_fin_ult1	SVM	0.89609	0.84771	0.93158	705.928	13.446

FUENTE: INVESTIGACIÓN.
ELABORADO POR: AUTOR.

Anexo 3: Core de la implementación del proceso de carga de datos.

```
config.sink("01-loadData.txt")

c.title("Carga de datos desde el archivo CSV.")

c.subtitle("Tiempo de Carga de los datos desde archivo inicial CSV")
time1 <- out_(print.time.return(data <- read.table(paste(dir.data, "train_ver2.csv", sep = dir.sep), sep = ",", na.strings = c("NA", ""), header = T, as.is = F, strip.white = T)))

divider()

## Visualización de las clases de los atributos de los datos.
c.subtitle("Clases de los atributos de los datos iniciales")
print(sapply(data, class))

c.subtitle("Cantidad de cada tipo de variables en el dataset")
clases <- summary(factor(sapply(data, class)))
df <- data.frame(type = names(clases), cant = unname(clases))
print(df)

# Segun la descripción de los datos donde se obtuvo los datos se deben cambiar el tipo de dato a ciertas variables
var_to_factor <- c(2, 8, 10, 19, 20, 22, 25:48)
data <- asfactorCols(data, var_to_factor)

c.subtitle("Cantidad de cada tipo de variables en el dataset despues de la correcta asignacion de tipo de varible")
clases2 <- summary(factor(sapply(data, class)))
df2 <- data.frame(type = names(clases2), cant = unname(clases2))
print(df2)
divider()
# Vista inicial de los datos para conocer su estructura.
c.subtitle("Tiempo de summary")
print.time(summarize <- summary_data(data))

c.subtitle("Summary de los datos iniciales en bruto")
print(summarize)
# Guardado de los datos en archivo de R en formato Rda.
c.subtitle("Tiempo de guardado de datos en formato Rda")
time2 <- out_(print.time.return(save(data, file = paste(dir.project, "data.Rda", sep = dir.sep))))
```

Anexo 4: Core de la implementación del proceso de análisis descriptivo de datos.

```
c.subtitle("Porcentaje de valores faltantes por columna")
per.col.faltantes <- sum.cols.faltantes[sum.cols.faltantes > 0]/dim(data)[1]*100
print(sapply(per.col.faltantes, function(x){ sprintf('%.2f', x)}))
c.subtitle("Tiempo de cálculo de verificación de registros con valores faltantes")
print.time(sum.row.faltantes <- rowSums(is.na(data)))
c.subtitle("Cantidad de registros con valores faltantes")
print(length(sum.row.faltantes))
c.subtitle("Porcentaje máximo y mínimo en los registros con valores faltantes")
rows.faltantes <- sum.row.faltantes[sum.row.faltantes > 0]/dim(data)[2]*100
print(paste("Maximo: ", sprintf("%0.003f%%", max(rows.faltantes)),", mínimo: ",
sprintf("%0.003f%%", min(rows.faltantes)), sep=""))
c.subtitle("Tiempo de cálculo de verificación de datos con valores faltantes")
print.time(datos.faltantes <- which(is.na(data) != 0))
c.subtitle("Porcentaje de datos con valores faltantes")
per.datos.faltantes <- length(datos.faltantes)/(dim(data)[1]*dim(data)[2])*100
print(per.datos.faltantes)

## Configurar la salida a la consola
sink()
sink()

# generacion de las graficas para conocer las caracteristicas de los datos
indexs <- c(1, 3:5, 8, 10, 12:22, 24:48)
for (index in indexs) {
  data.table <- as.data.frame(table(data[, index], useNA = "ifany"))
  name <- names(data)[index]
  angle.this <- 0
  factor <- 0.05
  size.this <- 5
  if(dim(data.table)[1]>20){
    angle.this <- 90
    factor <- 0.1
    size.this <- 4
  }
  graph <- ggplot(data.table, aes(x= Var1, y = Freq, fill=Var1, colour=Var1)) +
  geom_col() +
  geom_text(aes(label = Freq), size= size.this, position = position_nudge(y =
(max(data.table$Freq)*factor)), angle = angle.this) +
  labs(title = "Histograma de Frecuencias", x = "Valor", y = "Frecuencia", subtitle =
paste0("Atributo: ", name)) +
  guides(fill="none", colour= "none") +
  theme(axis.text.x = element_text(angle = angle.this))

#  ggsave(paste(name, " histogram", ".pdf", sep = ""))
  ggsave(paste(dir.plots, dir.sep, name, " histogram", ".png", sep = ""))
}
```

Anexo 5: Core de la implementación del proceso de reducción de datos.

```
# Archivo donde se mostrara los resultados
config.sink("02-cleanData.txt")

c.title("Eliminación de información no relevante para la clasificación")

## Configuración de semilla para obtención de valores aleatorios para hacer el proceso reproducible
set.seed(seed.parallel, "L'Ecuyer")
print(sprintf("Se configuro la semilla con el valor de: %s", seed.parallel))

# Carga de los datos preparados en la fase 1
c.subtitle("Tiempo de carga de los datos desde el archivo data.Rda")
time1 <- out_(print.time.return(load(paste(dir.project, "data.Rda", sep = dir.sep))))

c.subtitle("Memoria RAM usada por los datos sin preprocesar")
mem_1 <- object_size(data)
print.bytes(mem_1)

#
# ELIMINACIÓN DE ATRIBUTOS ATRIBUTOS CONSTANTES
# ------------------------------------------------------------------
c.subtitle("Summary de los atributos sin preprocesar")
summary1 <- summary(data)
print(summary1)


c.subtitle("Summary de los atributos constantes")
print(summary(data %>% select(conyuemp, tipodom)))


c.subtitle("Se eliminaran los atributos que son constantes")
names <- names(data %>% select(conyuemp, tipodom))
print(names)

data_clean <- data %>% select(-conyuemp, -tipodom)

#
# ELIMINACIÓN DE ATRIBUTOS CON INFORMACIÓN DUPLICADA
# ------------------------------------------------------------------
c.subtitle("Se eliminara el atributo nomprov, al ya estar representada su informacion en el atributo cod_prov")
data_clean <- data_clean %>% select(-nomprov)

# Liberación de memoria usada por el objeto data
rm.custom2(c("data"))
```

Anexo 6: Core de la implementación del proceso de recodificación de datos.

```
#
# RECODIFICANDO VALORES
# ------------------------------------------------------------------
c.title("Limpieza de atributos con valores erroneos")

c.subtitle("Summary del atributo indrel_1mes sin recodificar")
summary_indrel_mes <- summary(data_clean$indrel_1mes)
print(summary_indrel_mes)
#limpieza de valores en la columna indrel_1mes
data_clean <-
  data_clean %>%
  mutate(indrel_1mes = gsub("^1.0$",
                    1,
                    indrel_1mes)) %>%
  mutate(indrel_1mes = gsub("^2.0$",
                    2,
                    indrel_1mes)) %>%
  mutate(indrel_1mes = gsub("^3.0$",
                    3,
                    indrel_1mes)) %>%
  mutate(indrel_1mes = gsub("^4.0$",
                    4,
                    indrel_1mes))
# Recodificación de la variable de caracter a factor
data_clean$indrel_1mes <- as.factor(data_clean$indrel_1mes)


c.subtitle("Summary del atributo indrel_1mes despues de recodificar su valores")
summary_indrel_mes2 <- summary(data_clean$indrel_1mes)
print(summary_indrel_mes2)
```

Anexo 7: Core de la implementación del proceso tratamiento de valores faltantes.

```
#
# IMPUTACIÓN DE ATRIBUTOS CON VALORES FALTANTES
# ------------------------------------------------------------------------

# Se probo el algoritmo de imputacion proporcionado por el package RKEEL
# causando que tome un tiempo de horas por lo que se opto por buscar otros metodos
# algorithm <- RKEEL::MostCommon_MV(data_train, data_test)

temp.sum.cols.faltantes <- colSums(is.na(data_clean))
temp.index.cols.faltantes <- which(temp.sum.cols.faltantes != 0)
c.subtitle("Atributos con valores faltantes")
print(temp.index.cols.faltantes)

c.subtitle("Resumen de los atributos con valores faltantes")
summary4 <- summary(data_clean[,temp.index.cols.faltantes])
print(summary4)

c.subtitle("Atributos con tasa de valores faltantes menores a 5%")
temp.rate.missing <- temp.sum.cols.faltantes/dim(data_clean)[1]*100

index.cols.faltantes.manejables <- which(temp.rate.missing > 0 & temp.rate.missing < 5)
c.subtitle("Atributos con tasas de valores faltantes manejables")
print(index.cols.faltantes.manejables)

c.subtitle("Imputación de atributos con valores faltantes menores a una tasa de 5%")
time2 <- out_(print.time.return(temp.imp <- mlr::impute(data_clean, classes = list(integer =
mlr::imputeMean(), factor = mlr::imputeMode()))))
print(paste("Tiempo que demoro el proceso de imputación: ", time2, sep=""))

data_clean <- temp.imp$data

# Eliminación de memoria de objetos temporales
rm.t.vars()
```

Anexo 8: Core de la implementación del proceso discretización de datos.

```
#
# DISCRETIZACIÓ DE VARIABLES NUMERICAS PARA POSTERIOR IMPUTACIÓN
# ------------------------------------------------------------------------

# Se probaron las siguientes funciones de discretizacion sin obtener un buen desempeño ya que causaban
# errores de memoria dprep::disc.mentr, dprep::discretevar y dprep::disc.ew.

# Los discretizadores por frecuencia y por ancho fijo creavan intervalos muy grandes.

c.subtitle("Tiempo que demoro el proceso de discretización del atributo age")
time2.1 <- out_(print.time.return(temp.1 <- arules::discretize(data_clean$age, "cluster", categories=20)))
data_clean$age <- temp.1

c.subtitle("Tiempo que demoro el proceso de discretización del atributo antiguedad")
time2.2 <- out_(print.time.return(temp.2 <- arules::discretize(data_clean$antiguedad, "cluster", categories=20)))
data_clean$antiguedad <- temp.2

c.subtitle("Tiempo que demoro el proceso de discretización del atributo renta")
time2.3 <- out_(print.time.return(temp.3 <- arules::discretize(data_clean$renta, "cluster", categories=10)))
data_clean$renta <- temp.3

# Eliminación de memoria de objetos temporales
rm.t.vars()
```

Anexo 9: Core de la implementación del proceso imputación de datos mediante Naive Bayes.

```
time3 <- Sys.time()
print(time3)

temp.train.set <- which(!is.na(data_clean$renta))

temp.classif.task <- mlr::makeClassifTask(id = "imputedTask", data = data_clean[temp.train.set, ], target = "renta", fixup.data = "no", check.data = FALSE)
temp.classif.task.test <- mlr::makeClassifTask(id = "imputed.testTask", data = data_clean[-temp.train.set, ], target = "renta", fixup.data = "no", check.data = FALSE)

temp.lrn <- mlr::makeLearner("classif.naiveBayes")
time3.1 <- out_(print.time.return(temp.mod <- mlr::train(temp.lrn, temp.classif.task)))
time3.2 <- out_(print.time.return(temp.task.pred <- predict(temp.mod, task = temp.classif.task.test))) # mlr Predict S3 Object

c.subtitle("Tiempo de finalización del proceso de imputación por el algoritmo Naive Bayes")
time3.3 <- Sys.time()
print(time3.3)

c.subtitle(paste0("El tiempo que tomo construir el clasificador por el algoritmo Naive Bayes: ", time3.1))

c.subtitle(paste0("El tiempo que tomo predecir los valores faltantes con el clasificador: ", time3.2))

# Unión de los valores imputados en el atributo renta del data set original.
temp.index.imputed.set <- as.integer(row.names(temp.task.pred$data))
data_clean$renta[temp.index.imputed.set] <- temp.task.pred$data[ ,3]

# Se guardan el modelo construido por Naive Bayes y también los valores imputados
save(temp.task.pred, temp.mod, file = paste(dir.project, "model_data_imputed.Rda", sep = "/"))

temp.sum.cols.faltantes.imp <- colSums(is.na(data_clean))
index.cols.faltantes.imp <- which(temp.sum.cols.faltantes.imp != 0)
c.subtitle("Atributos con valores faltantes despues de la imputación por medio de Naive Bayes")
print(index.cols.faltantes.imp)
# Guardado de los datos en archivo de R en formato Rda.
c.subtitle("Tiempo de guardado de datos en formato Rda despues del proceso de limmpieza de datos")
time5 <- out_(print.time.return(save(data_clean, file = paste(dir.project, "data_clean.Rda", sep = dir.sep))))
```

Anexo 10: Core de la implementación del proceso de experimentación (1).

```
# Iteracciones del remuestreo fuera del bucle
num.reps <- c(5, 10, 50)

cat(paste0("\nSe realizarán: ", (num.reps), " repeticiones"))

num.folds <- 2
cat(paste0("\nNúmero de k en K-fold Cross Validation: ", num.folds, "\n"))

predict.type <- "test"
cat(paste0("\nConjunto a evaluar: ", predict.type, "\n"))

resample.type <- "RepCV"
cat(paste0("\nMétodo de resampling: ", resample.type, "\n"))

save.models <- FALSE
cat(paste0("\nSe guardaran los modelos: ", save.models, "\n"))

save.predicciones <- FALSE
cat(paste0("\nSe guardaran las predicciones: ", save.predicciones, "\n"))

info.experiment <- TRUE
cat(paste0("\nSe presentara información del proceso de la expermentación: ", info.experiment, "\n"))


# Lista de resultados de la expermentación
bmrs.paths <- c()

index <- 1
while(index <= length(num.reps)) {

  # Configuración de la paralelización del experimento
  # Se configura por en nivel de paralelización proporcionados por MLR
  # https://mlr-org.github.io/mlr-tutorial/release/html/parallelization/index.html#parallelization-levels
  # levels= "mlr.benchmark, mlr.resample, mlr.selectFeatures, mlr.tuneParams, mlr.ensemble"
  if(parallel.enable) { parallelMap::parallelStartMulticore(cpus=cores.parallel, logging=log.parallel, storagedir= getwd(), level=level.parallel, show.info=info.parallel) }

  # Captura de hora y fecha de inicio del experimento
  timeInicial <- Sys.time()

  # Configuración de remuestreo fuera del bucle
  outer <- mlr::makeResampleDesc(resample.type, predict=predict.type, folds=num.folds, reps=num.reps[index])
  # Ejecución del experimento con las tareas, algoritmos, medidas de desempeño y la configuración de la función benchmark para conservar la memoria del computador
```

```
   bmr <- mlr::benchmark(lrns, tasks, outer, measures=measures, keep.pred=save.predicciones, models=save.models, show.info=info.experiment)

   # Captura de hora y fecha del fin del experimento
   timeFinal <- Sys.time()

   # Elementos que se almacenaran en la lista resultados
   performances <- mlr::getBMRPerformances(bmr, task.ids = NULL, learner.ids = NULL, as.df = TRUE)
   result <- list(label= paste("Experimento: ", index), timeTrainSum= sum(performances$timetrain), timePredictSum= sum(performances$timepredict),
      timeInicial= timeInicial, timeFin= timeFinal, bmr= bmr, repeticiones= num.reps[index])

   # Detención de la paralelización
   if(parallel.enable) { parallelMap::parallelStop() }
   if(show.results) print.list(result)
   print(paste("Se ah culminado el experimento: ", index, sep = ""))
   path.bmr <- paste("brm_", index, ".Rda", sep = "")
   path.bmr <- check.name(path.bmr)
   save(result, file = path.bmr)

   bmrs.paths <- c(bmrs.paths, path.bmr)
   index <- index + 1
}

print.list(bmrs.paths)
## Lista de resultados de la expermentación
bmrs <- list()
result.df <- data.frame()
index <- 1
while(index <= length(bmrs.paths)) {

   load(bmrs.paths[[index]])

   df <- cbind(label= result$label, repeticiones= result$repeticiones,
         mlr::getBMRAggrPerformances(result$bmr, task.ids= NULL, learner.ids= NULL, as.df= TRUE),
         timeTrainSum=result$timeTrainSum, timePredictSum= result$timePredictSum)

   result.df <- rbind(result.df, df)
   index <- index + 1
}

# Creación del archivo CSV de salida de los resultados de los experimentos
path.csv <- check.name("03-performances.csv")
write.table(result.df, file=path.csv, sep= ",", dec= ".", row.names= FALSE, na="", col.names=TRUE)
```

Anexo 11: Core de la implementación del proceso de reducción de datos (2).

```
#Obtención la variavilidad de los datos
fechas <- as.character(unique(data_clean$fecha_dato))
meses <- list()
for(i in fechas){
  temp.mes <- data_clean %>% filter(fecha_dato == i)
  meses <- c(meses, list(temp.mes))
}

cols <- c(21:44)
i <- meses[[1]]
df <- data.frame(col =names(i)[cols])

for(i in meses){
  df2 <- data.frame()
  for(j in cols){
    t <- table(i[, j])
    df2 <- rbind(df2, data.frame(yes= t[2], no=t[1]))
  }
  df <- cbind(df, df2)
}
row.names(df) <- c()
c.subtitle("Presentación de tabla de variavilidad de datos entre meses")
print(df)

#Seleccion de los datos mas representativos del conjunto de datos
data_clean <- data_clean %>% filter(fecha_dato == "2016-05-28")

c.subtitle("Summary de los datos mas representativos")
summary1 <- summary(data_clean)
print(summary1)

#
# ELIMINACIÓN DE ATRIBUTOS IDENTIFICADORES Y ATRIBUTOS CONSTANTES
# -----------------------------------------------------------------------
c.subtitle("Summary de los atributos identificadores o constantes")
print(summary(data_clean %>% select(fecha_dato, ncodpers)))

c.subtitle("Se eliminaran los atributos que son identificadoras o constantes")
names <- names(data_clean %>% select(fecha_dato, ncodpers))
print(names)

data_clean <- data_clean %>% select(-fecha_dato, -ncodpers)
```

Anexo 12: Core de la implementación del proceso de reducción de categorías.

```
fechas2meses <<- function(data.vector){
  library(lubridate)
  this_day <- lubridate::today()
  this_day
  dates <- ymd(data.vector)
  dates <- (year(this_day) * 12 + month(this_day)) - (year(dates) * 12 + month(dates))
  detach_package("lubridate")
  return(dates)
}


prov2autonoma <<- function(data.vector){
  load(paste(dir.project, "utils.Rda", sep = dir.sep))
  autonomas <- plyr::mapvalues(data.vector, from = provincias[,1], to = provincias[,4])
  rm.custom2(c("paises", "provincias"))
  return(autonomas)
}


pais2continente <<- function(data.vector){
  load(paste(dir.project, "utils.Rda", sep = dir.sep))
  contientes <- plyr::mapvalues(data.vector, from = paises[,4], to = paises[,1])
  rm.custom2(c("paises", "provincias"))
  return(contientes)
}

#
# REDUCCIÓN DE CATEGORIAS DE LOS ATRIBUTOS CATEGORICOS
# ------------------------------------------------------------------------

if(reload.utils){
  paises <- read.table(paste(dir.data, "continente_pais.csv", sep = dir.sep), sep = ",",
strip.white = TRUE, na.strings = c("?"), header = T, as.is = T, encoding="UTF-8")
  provincias <- read.table(paste(dir.data, "autonoma_provincia.csv", sep = dir.sep), sep = ",",
strip.white = TRUE, na.strings = c("?"), header = T, as.is = T, encoding="UTF-8")
  save(paises, provincias, file = paste(dir.project, "utilidades.Rda", sep = dir.sep))
}

data_clean$pais_residencia <- pais2continente(data_clean$pais_residencia)
data_clean$fecha_alta <- fechas2meses(data_clean$fecha_alta)
data_clean$canal_entrada <- reduce.canales(data_clean$canal_entrada)
data_clean$cod_prov <- prov2autonoma(data_clean$cod_prov)


c.subtitle("Summary de los datos despues de reducción de categorias")
summary2 <- summary(data_clean)
print(summary2)
```

Anexo 13: Core de la implementación del proceso de experimentación (2).

```
index <- 1
while(index <= length(num.reps)) {

   # Configuración de la paralelización del experimento
   # Se configura por en nivel de paralelización proporcionados por MLR
   if(parallel.enable) { parallelMap::parallelStartMulticore(cpus=cores.parallel, logging=log.parallel, storagedir= getwd(), level=level.parallel, show.info=info.parallel) }

   # Captura de hora y fecha de inicio del experimento
   timeInicial <- Sys.time()

   # Configuración de remuestreo fuera del bucle
   outer <- mlr::makeResampleDesc(resample.type, predict=predict.type, folds=num.folds, reps=num.reps[index])

   # Ejecución del experimento con las tareas, algoritmos, medidas de desempeño y la configuración de la función benchmark para conservar la memoria del computador
   bmr <- mlr::benchmark(lrns, tasks, outer, measures=measures, keep.pred=save.predicciones, models=save.models, show.info=info.experiment)

   # Captura de hora y fecha del fin del experimento
   timeFinal <- Sys.time()

   # Elementos que se almacenaran en la lista resultados
   performances <- mlr::getBMRPerformances(bmr, task.ids = NULL, learner.ids = NULL, as.df = TRUE)
   result <- list(label= paste("Experimento: ", index), timeTrainSum= sum(performances$timetrain), timePredictSum= sum(performances$timepredict),
      timeInicial= timeInicial, timeFin= timeFinal, bmr= bmr, repeticiones= num.reps[index])

   # Detención de la paralelización
   if(parallel.enable) { parallelMap::parallelStop() }

   if(show.results) print.list(result)

   print(paste("Se ah culminado el experimento: ", index, sep = ""))

   path.bmr <- paste("brm_", index, ".Rda", sep = "")
   path.bmr <- check.name(path.bmr)
   save(result, file = path.bmr)

   bmrs.paths <- c(bmrs.paths, path.bmr)
   index <- index + 1
}


print.list(bmrs.paths)
```

```
#
# PRESENTACIÓ DE RESULTADOS
# ------------------------------------------------------------------

## Lista de resultados de la expermentación
bmrs <- list()

result.df <- data.frame()
index <- 1
while(index <= length(bmrs.paths)) {

   load(bmrs.paths[[index]])

   df <- cbind(label= result$label, repeticiones= result$repeticiones,
            mlr::getBMRAggrPerformances(result$bmr, task.ids= NULL, learner.ids= NULL, as.df= TRUE),
            timeTrainSum=result$timeTrainSum, timePredictSum= result$timePredictSum)

   result.df <- rbind(result.df, df)
   index <- index + 1
}

# Creación del archivo CSV de salida de los resultados de los experimentos
path.csv <- check.name("05-performances.csv")
write.table(result.df, file=path.csv, sep= ",", dec= ".", row.names= FALSE, na="", col.names=TRUE)

# Configurar la salida a la consola
sink()
sink()
```

Anexo 14: Core de la implementación del proceso de experimentación (3).

```
# Configuración de los algoritmos a evaluar en el experimento
nnetLearner <- mlr::makeLearner("classif.nnet", predict.type = "prob")
naiveBayesLearner <- mlr::makeLearner("classif.naiveBayes", predict.type = "prob")
svmLearner <- mlr::makeLearner("classif.svm", predict.type = "prob")
earthLearner <- mlr::makeLearner("classif.earth", predict.type = "prob")

# Learners
lrns <- list(nnetLearner, naiveBayesLearner, svmLearner, earthLearner)

c.subtitle("Se configuraron los siguientes algoritmos:")
print.list(lrns)

#
# EXPERMIMENTO
# ------------------------------------------------------------------

# Iteracciones del remuestreo fuera del bucle
num.reps <- c(5)

cat(paste0("\nSe realizarán: ", (num.reps), " repeticiones"))

num.folds <- 2
cat(paste0("\nNúmero de k en K-fold Cross Validation: ", num.folds, "\n"))

predict.type <- "test"
cat(paste0("\nConjunto a evaluar: ", predict.type, "\n"))

resample.type <- "RepCV"
cat(paste0("\nMétodo de resampling: ", resample.type, "\n"))

save.models <- FALSE
cat(paste0("\nSe guardaran los modelos: ", save.models, "\n"))

save.predicciones <- FALSE
cat(paste0("\nSe guardaran las predicciones: ", save.predicciones, "\n"))

info.experiment <- TRUE
cat(paste0("\nSe presentara información del proceso de la expermentación: ",
info.experiment, "\n"))


# Lista de resultados de la expermentación
bmrs.paths <- c()

index <- 1
while(index <= length(num.reps)) {

   # Configuración de la paralelización del experimento
```

```
if(parallel.enable) { parallelMap::parallelStartMulticore(cpus=cores.parallel, logging=log.parallel, storagedir= getwd(), level=level.parallel, show.info=info.parallel) }

   # Captura de hora y fecha de inicio del experimento
   timeInicial <- Sys.time()

   # Configuración de remuestreo fuera del bucle
   outer <- mlr::makeResampleDesc(resample.type, predict=predict.type, folds=num.folds, reps=num.reps[index])
   #Se realiza un simple proceso de prediccion 75% para train y 25% para test
   #outer = mlr::makeResampleDesc("Holdout", split= 3/4)



   # Ejecución del experimento con las tareas, algoritmos, medidas de desempeño y la configuración de la función benchmark para conservar la memoria del computador
   bmr <- mlr::benchmark(lrns, tasks, outer, measures=measures, keep.pred=save.predicciones, models=save.models, show.info=info.experiment)

   # Captura de hora y fecha del fin del experimento
   timeFinal <- Sys.time()

   # Elementos que se almacenaran en la lista resultados
   performances <- mlr::getBMRPerformances(bmr, task.ids = NULL, learner.ids = NULL, as.df = TRUE)
   result <- list(label= paste("Experimento: ", index), timeTrainSum= sum(performances$timetrain), timePredictSum= sum(performances$timepredict),
      timeInicial= timeInicial, timeFin= timeFinal, bmr= bmr, repeticiones= num.reps[index])

   # Detención de la paralelización
   if(parallel.enable) { parallelMap::parallelStop() }

   if(show.results) print.list(result)

   print(paste("Se ah culminado el experimento: ", index, sep = ""))

   path.bmr <- paste("brm_", index, ".Rda", sep = "")
   path.bmr <- check.name(path.bmr)
   save(result, file = path.bmr)

   bmrs.paths <- c(bmrs.paths, path.bmr)
   index <- index + 1
}
sink()
sink()
```

Anexo 15: Core de la implementación del proceso de generación de gráficas y tablas de comparación.

```
init.theme()

table.filtrada <- make.tables(data, task.id, learner.id)
table.filtrada <- as.data.frame(table.filtrada)
path.csv <- check.name("learner-task resumen.csv")
write.table(table.filtrada, file=path.csv, sep= ",", dec= ".", row.names= FALSE, na="", col.names=TRUE)


measure.names <- c("auc.test.mean", "timetrain.test.mean", "timepredict.test.mean")
measures <- list(mlr::auc, mlr::timetrain, mlr::timepredict)
titles <- c("Valor de AUC", "Tiempo de entrenamiento (s)", "Tiempo de prueba (s)")

config.sink("test.txt")
index <- 1
for(index in c(1:length(measures))){
   measure.title <- titles[index]
   measure.now <- measures[[index]]
   measure.name <- measure.names[index]

   fr <- friedmanTestCustom(data, measure = measure.now)

   post <- friedmanPostHocTestCustom(data, measure = measure.now)


   p <- plotBMRBoxplotsCustom(data, measureName = measure.name)
   p <- p + geom_boxplot(aes(colour = learner.id)) + labs(title = paste0("Boxplot de los resultados de las pruebas"), subtitle=paste0("La métrica evaluada es el: ", measure.title), x = "Algoritmos", y = measure.title) +
      guides(colour= guide_legend("Algoritmos")) + theme(legend.position = "bottom")
   name <- check.name(paste0("Boxplot general ", measure.now$id,".png"))
   ggplot2::ggsave(name, plot=p)

   p <- plotBMRSummaryCustom(data, measure = measure.now, trafo = "none")
   p <- p + labs(title = paste0("Resumen de los resultados de las pruebas"), subtitle=paste0("La métrica evaluada es el: ", measure.title), y = "Producto del banco", x = measure.title) +
  guides(col= guide_legend("Algoritmos")) + theme(legend.position = "bottom")
   name <- check.name(paste0("Summary plot general ", measure.now$id,".png"))
  ggplot2::ggsave(name, plot=p)

  p <- plotBMRSummaryCustom(data, measure = measure.now, trafo = "rank")
   p <- p + labs(title = paste0("Resumen de los resultados de las pruebas"), subtitle=paste0("La métrica evaluada es el: ", measure.title), y = "Producto del banco", x = measure.title) +
  guides(col= guide_legend("Algoritmos")) + theme(legend.position = "bottom")
   name <- check.name(paste0("Summary rank plot general ", measure.now$id,".png"))
```

```
  ggplot2::ggsave(name, plot=p, width = 7)

    p <- plotDensityPlotsCustom(data, measureName = measure.name)
    p <- p + aes(colour = learner.id) + labs(title = paste0("Gráfica de densidad del valor de AUC"), y = "Densidad", x = "AUC") +
        guides(colour= guide_legend("Algoritmos")) + theme(legend.position = "bottom")
    name <- check.name(paste0("Density general ", measure.now$id,".png"))
    ggplot2::ggsave(name, plot=p)

    obj <- generateCritDifferencesDataCustom(data, measure = measure.now, p.value = 0.05)
    p <- plotCritDifferencesCustom(obj)
    p <- p + labs(title = paste0("Diagrama de diferencias críticas"), subtitle=paste0("La métrica evaluada es el: ", measure.title))
    name <- check.name(paste0("diferencias críticas ", measure.now$id,".png"))
    ggplot2::ggsave(name, plot=p)

    cat("\n")
    cat("\n")
    print(fr)
    cat("\n")
    cat("\n")
    print(post)
    cat("\n")
    cat("\n")
    print(obj)
}
sink()


table.filtrada <- make.tables(data, learner.id)

index.x <- 1
index.y <- 2
index.learn <- 2
Algoritmos <- table.filtrada[, index.learn]

graph <- ggplot(table.filtrada,
  aes(
    x=table.filtrada[ , index.x],
    y=table.filtrada[, index.y],
    fill= Algoritmos)) +
geom_col() +
labs(title = "Media de la metrica AUC de los clasificadores", x = "Algoritmos", y = "Valor de AUC") +
geom_text(aes(label = sprintf('%.5f', table.filtrada[, index.y])), size=5, position = position_nudge(y = +(0.02))) +
guides(fill="none")
ggsave("auc resumen.png")
```