Facultad 2

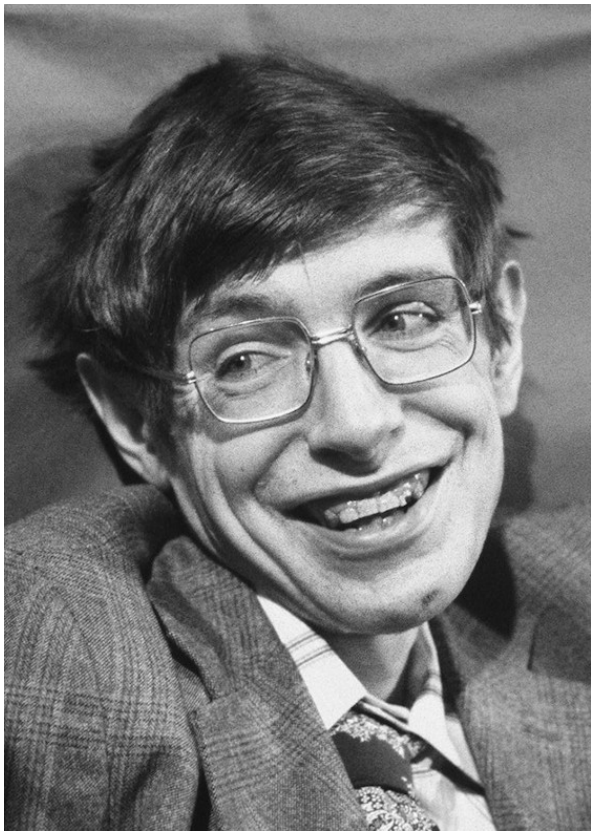
Método para la detección del fraude en transacciones bancarias con escenarios de Flujo de Datos.

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Alayn Lado Chaviano, Leonardo González Abreu
Tutores: Dr.C Héctor Raúl González Diez, Ing. Vladimir Milián Núñez
Co-tutor: Ing. Odeynis Valdés Suárez

La Habana, noviembre de 2022

“Año 64 de la Revolución”



(…) El éxito en la creación de IA sería el evento más grande en la historia de la humanidad.

Desafortunadamente, también podría ser el último, a menos que aprendamos a evitar los riesgos (…)

Stephen Hawking

DECLARACIÓN DE AUTORÍA

Los autores del trabajo de diploma con título ***"Método para la detección del fraude en transacciones bancarias con escenarios de Flujo de Datos"***, conceden a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declaran como únicos autores de su contenido. Para que así conste firman la presente a los 30 días del mes de noviembre del año 2022.

Alayn Lado Chaviano

_____
Firma del Autor

Leonardo González Abreu

_____
Firma del Autor

Dr.C Héctor Raúl González Diez

_____
Firma del Tutor

Ing. Vladimir Milián Núñez

_____
Firma del Tutor

Ing. Odeynis Valdés Suárez

_____
Firma del Co-tutor

DATOS DE CONTACTO

Profesor Titular Dr.C. Matemáticas Héctor Raúl González Diez (hglez@uci.cu), 7 837-2577.

Líneas de Investigación:

- ✓ Optimización matemática en el contexto del aprendizaje supervisado.
- ✓ Predicción con salidas múltiples con tratamiento de dependencia.
- ✓ Aprendizaje de funciones de distancia para problemas de predicción con salidas múltiples.

Profesor Auxiliar Ing. Vladimir Milián Núñez (vmilian@uci.cu), 7835-8134.

Líneas de investigación:

- ✓ Procesamiento de lenguaje natural y aprendizaje automático.
- ✓ Ingeniería de características para problemas de alta dimensionalidad.
- ✓ Big data.

Profesor Ing. Odeynis Valdés Suárez (ovaldes@uci.cu), 58058414.

Líneas de investigación:

- ✓ Procesamiento distribuido.
- ✓ Deep Learning.

AGRADECIMIENTOS

Infinita gratitud a nuestros tutores de tesis, Dr.C Héctor Raúl González Diez, Ing. Vladimir Milián Núñez e Ing. Odeynis Valdés Suárez, porque en cada sesión siempre aprendimos algo nuevo, por su confianza, porque sin sus enseñanzas, tenacidad y seguimiento no habríamos logrado este gran paso.

Gracias a todos los maestros que dedicaron su tiempo y compartieron sus conocimientos y experiencias con nosotros a lo largo de esta carrera, a la UCI por darnos la oportunidad de alcanzar este objetivo.

Nuestro agradecimiento a todos los que han creído en nosotros, a los que sin saberlo nos han ayudado a prepararnos profesional e individualmente, gracias por enfrentarnos a retos y reconocer logros y fracasos.

DEDICATORIA

Alayn:

Esta tesis está dedicada:

Con todo mi amor y cariño a mi padre Humberto y mi madre Kenia por estar siempre a mi lado impulsándome a continuar luchando por mis metas. A mi hermana Arlyn y mi novia Yesliany para ser su inspiración e impulsarlas a cumplir sus metas. En especial a mi novia por lograr guiarme en la adversidad apoyándome siempre.

A todos los que confiaron en mi cuando estaba perdido y me dieron la segunda oportunidad. A todas esas personas que nos apoyaron y han hecho que este trabajo se realice con éxito, en especial, a aquellos que nos confiaron esta importante tarea, abrieron sus puertas y compartieron sus conocimientos. A todos esos profesores que brindaron sus brazos para construir la escalera para alcanzar esta meta.

A todos muchas gracias.

Leonardo:

Mi tesis la dedico con todo mi amor y cariño a mi amada esposa Yaneisy, siempre por su paciencia, sacrificio y esfuerzo. Por estar siempre a mi lado apoyándome, por darme una carrera para nuestro futuro y por creer en mi capacidad. Aunque hemos pasado momentos difíciles siempre ha estado brindándome su comprensión, cariño y amor.

A mi amada hija Ana Laura por ser mi fuente de motivación e inspiración para poder superarme cada día más y así poder luchar para que la vida nos depare un futuro mejor.

A mi familia especialmente a mis queridos padres Osvel y Lucía por no perder la fe en mí y no dejarme decaer para que siguiera adelante y cumpliera con mis ideales.

A mis compañeros y amigos presentes y pasados, quienes sin esperar nada a cambio compartieron su conocimiento, alegrías y tristezas y a todas aquellas personas que durante esta carrera de ingeniería estuvieron a mi lado apoyándome y lograron que este sueño se haga realidad.

Gracias a todos.

RESUMEN

En este trabajo de diploma se desarrollará un sistema de detección de fraude en pagos con transacciones bancarias en tiempo real utilizando tecnologías de procesamiento distribuido. En la actualidad uno de los problemas ante los que se enfrentan las organizaciones con fines de crédito es el de administrar diariamente un volumen de operaciones excesivamente grande para su procesamiento, el desbalance del problema entre el número de operaciones fraudulentas y las transacciones normales, así como un elevado flujo de operaciones por unidad de tiempo. El objetivo del presente trabajo consiste en el desarrollo de un método de detección automatizada de anomalías en transacciones bancarias para la identificación del fraude. Para ello se realiza una fundamentación teórica previa de los principales conceptos y algoritmos relacionados con el campo de acción, así como las métricas, herramientas y tecnologías que se usan para su estudio y aplicación. Se selecciona como metodología KDD (Knowledge Discovery in Databases) para guiar el ciclo de vida del proyecto. Se obtienen cuáles son los algoritmos que mejores resultados ofrecen en la detección del fraude sobre las variantes propuestas, para escenarios de flujos de datos, así como un método para la detección automatizada de anomalías en tiempo real para transacciones bancarias.

Palabras clave: detección de fraude, procesamiento distribuido, método, flujos de datos, transacciones bancarias.

ABSTRACT

In this diploma work, a fraud detection system will be developed in payments with banking transactions in real time using distributed processing technologies. Currently, one of the problems faced by organizations for credit purposes is managing an excessively large volume of operations on a daily basis for processing, the imbalance of the problem between the number of fraudulent operations and normal transactions, as well as a high flow of operations per unit of time. The objective of this work consists in the development of an automated detection method of anomalies in banking transactions for the identification of fraud. For this, a previous theoretical foundation of the main concepts and algorithms related to the field of action is carried out, as well as the metrics, tools and technologies used for their study and application. Knowledge Discovery in Databases (KDD) methodology is selected to guide the project life cycle. The algorithms that offer the best results in the detection of fraud on the proposed variants are obtained, for data flow scenarios, as well as a method for the automated detection of anomalies in real time for banking transactions.

Keywords: fraud detection, distributed processing, method, data streams, banking transactions.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, con el uso y normalización de las tecnologías se está viviendo un cambio en el paradigma social, en el cual los datos masivos cobran vital importancia. Si se tienen en consideración los grandes volúmenes de datos generados sólo en los pasados años y los avances adquiridos en la esfera científica, es de suma relevancia notar que dichos datos sobrepasan claramente la capacidad de comprensión, almacenamiento y recolección de los mismos sin el uso de herramientas adecuadas. Limitando así las capacidades de detección del fraude en instituciones con fines de crédito.

Con la finalidad de solucionar dicho dilema, siendo la detección de anomalías una técnica de Minería de Datos (MD) con un vasto conjunto de aplicaciones de análisis de datos y transacciones bancarias (TB), se propone hacer uso de la misma como vía para extraer conocimiento a partir de los datos existentes. En este contexto, se puede adquirir dicho conocimiento a través de la búsqueda de patrones estadísticamente confiables, ideas o conceptos derivados de los datos originales.

Las instituciones bancarias son consideradas un eslabón esencial para la economía de un país, así como para la población que recibe sus servicios. En general, sus actividades de financiamiento conllevan varios tipos de préstamos, tales como: para viviendas, proyectos, financiamiento a corto plazo, pequeñas y medianas empresas, comercio y de otros tipos. También es posible que solo se concentren en transacciones específicas con clientes que cumplan ciertos requisitos y con ciertos sectores industriales. En un sentido amplio, se considera a estas instituciones el objetivo principal de aquellos individuos que son promotores del fraude bancario.

Encabezando la lista de fraudes, por las diversas formas secretas encontradas para acceder a la información de las cuentas de crédito se encuentra el fraude en transacciones bancarias. Se puede detectar por medio de métodos de inteligencia artificial como la detección de anomalías.

La detección de anomalías tiene muchísimos casos de uso. Desde ciberseguridad (intrusión) hasta medicina (procesamiento de imagen), pasando por mantenimiento industrial, detección de fraude e incluso el internet de las cosas (monitorización). Un buen sistema de detección de anomalías puede hacer decrecer costes provenientes de áreas del negocio de muchas industrias. Existen varias categorías de técnicas de detección de anomalías para Aprendizaje Automático (AA) y dependen del conjunto de datos de entrenamiento que se provee, este último clasificado en supervisados, semi-supervisados y no supervisados.

Se intenta hacer uso del aprendizaje supervisado: empleado en aplicaciones financieras, para puntuación crediticia, clasificación de bonos y negociación algorítmica. De forma que su objetivo será descubrir grupos de datos similares entre sí, reproducir una distribución de datos de entrada o identificar patrones en las muestras.

Cada titular de cuenta generalmente tiene ciertos patrones de depósito de dinero y compras realizadas. Si hay un valor atípico en estos patrones, el banco debe poder detectarlo y analizarlo. Actualmente, estas operaciones son realizadas manualmente por un especialista ante una solicitud emitida por un cliente consiente de una perpetración por lo que, requieren de un sistema de mecanismos de seguridad para la detección de anomalías

en caso de sustracción fraudulenta o lavado de dinero.

Tras una consideración sobre este tema L. Eduardo Domínguez alerta sobre la realización de fraude bancario en Cubadebate:

- Usuarios cubanos reportan suplantación de identidad de la web de ENZONA: La plataforma para la realización de operaciones financieras y negocios digitales de la empresa cubana XETID. El método de perpetración empleado, es conocido como phishing o suplantación de identidad.
- Alertan sobre posible estafa para hacerse con tu tarjeta de electrónica: Circula por grupos de redes sociales, una alerta sobre cómo un estafador puede hacerse con el control de tu tarjeta de banco mediante el uso coordinado de las aplicaciones de Transfermóvil y ENZONA. La aparición de capturas de pantalla con chats privados donde individuos se hacen pasar por instituciones cubanas exigiendo datos personales de los usuarios, incendió aún más las redes sociales, en especial Twitter, YouTube y Telegram.

Con el objetivo de contrarrestar estas situaciones, en el Programa Nacional de Ciencias: “Telecomunicaciones e Informatización de la Sociedad”, se lleva a cabo el proyecto: “Plataforma para el análisis de grandes volúmenes de datos y su aplicación a sectores estratégicos”, teniendo a la Universidad de las Ciencias Informáticas (UCI) como entidad ejecutora principal con la tarea de solucionar los problemas existentes.

Desde el punto de vista de la modelación del problema para la detección del fraude en operaciones bancarias mediante aprendizaje automático existen varios problemas computacionales a tener en cuenta:

1. El volumen de operaciones que ocurren diariamente es excesivamente grande para que sea procesado por una computadora por lo que el problema debe ser tratado en entornos distribuidos.
2. El número de operaciones fraudulentas es muchísimo menor (aproximadamente 99.9% vs 0.1%) que las transacciones normales por lo que hay un desbalance del problema.
3. El flujo de operaciones que ocurren por unidad de tiempo es elevado.

Problema de investigación:

¿Cómo automatizar la detección de anomalías en escenarios de flujo de datos bancarios para la identificación de fraudes?

Objetivo general: Desarrollar un método de detección automática de anomalías en transacciones bancarias para la identificación de fraudes.

Objeto de estudio: La detección de anomalías.

Campo de acción: Métodos de detección de anomalías para la identificación de fraude bancario.

Objetivos específicos:

1. Caracterizar el marco teórico-conceptual del problema de la detección de fraudes bancarios, los enfoques basados en detección de anomalías, sustentado en escenarios de Flujo de Datos (FD).
2. Desarrollar algoritmos basados en el aprendizaje supervisado y la computación distribuida para la detección de anomalías en operaciones bancarias.
3. Validar la solución implementada mediante el diseño de experimentos sobre conjuntos de datos de referencia, comparando los resultados con otros algoritmos del estado del arte.

Tareas científicas:

Las tareas que guiaran la investigación son:

- Identificar, conceptualizar y caracterizar la detección de fraudes mediante el enfoque de la detección de anomalías.
- Identificar y caracterizar los algoritmos usados para la detección de anomalías en transacciones bancarias.
- Buscar algoritmos basados en AA para la detección de anomalías en fraudes bancarios.
- Desarrollar un método haciendo uso de los algoritmos encontrados para la detección de anomalías.
- Validar la solución propuesta mediante experimentos con juegos de datos.
- Comparar los resultados obtenidos entre los algoritmos usados.

Como entidad responsable de velar por la seguridad de los datos y activos, de las entidades estatales o particulares que se encuentran en los bancos cubanos, para el Banco Central de Cuba S.A. (BCC S.A.) con el fin de automatizar el proceso de detección de fraudes en transacciones bancarias el desarrollo de una solución práctica permitiendo su detección en tiempo real cobra vital importancia.

La investigación científica, incluyendo la del campo educativo, supone el dominio y empleo consciente de los

métodos de investigación, lo que incluye tanto el uso de los métodos teóricos como empíricos de investigación. El objetivo del presente estudio consiste en el desarrollo de un método automatizado para la detección de anomalías concretamente fraudes en transacciones bancarias. Para ello se emplearon a su vez el método teórico (Histórico-lógico) y el método empírico (Observación).

Método teórico

Histórico-lógico: Determina las tendencias actuales de los sistemas de detección, de los modelos de desarrollo, las técnicas, lenguajes y herramientas a utilizar en la investigación.

Método empírico

Observación: Se lleva a cabo para el proceso de detección de fraude bancario para obtener información sobre las etapas, los actores, y requerimientos del sistema entre otras.

Los aportes prácticos y sociales de la investigación

Se espera desarrollar un método informático que use un algoritmo para la detección del fraude bancario mediante la detección de anomalías.

El ***"Método para la detección del fraude en transacciones bancarias con escenarios de Flujo de Datos"*** es el principal aporte realizado por este trabajo de diploma, el mismo automatizará la tarea de detección de fraudes en transacciones bancarias, haciendo que sea más factible y seguro el trabajo para los usuarios.

La estructura del documento se desglosa mediante capítulos:

Capítulo 1: Aborda todas las bases teóricas relacionadas para el desarrollo del programa, referenciando a todas las investigaciones de la cual se extrajo información.

Capítulo 2: Refleja todo lo relacionado con los primeros 7 pasos de la metodología que se aplicará para la minería de datos.

Capítulo 3: Describe las últimas 2 fases de la metodología, las cuales abordan la evaluación y resultados comparativos de los algoritmos teniendo en cuenta las métricas establecidas, además de la definición del método informático que se desarrolló.

Capítulo 1: Informatización de la detección del fraude en transacciones bancarias.

El presente capítulo contiene los fundamentos imprescindibles para la comprensión de las relaciones existentes entre los conceptos expuestos. Los pilares de esta investigación comprenden la detección de fraudes (anomalías) dentro de instituciones bancarias, así como una presentación de modelos de AA como mejor solución al problema de detección de fraude en TB.

1.1 La detección de anomalías.

Según [CITATION Iku22 \l 3082], una de las ventajas más grandes que ha traído el avance tecnológico tiene que ver con la ayuda que puede brindar a las tareas cotidianas de una empresa, especialmente en materia de seguridad. Si buscas optimizar al máximo el trabajo de seguridad en tu empresa, la detección de anomalías basada en AA puede ser una de las mejores y más innovadoras estrategias.

Existen varios autores que abordan sobre este tema, no obstante para los oficios de estos estudios, en base a la bibliografía consultada, fueron seleccionados por su nivel de especificidad los de los siguientes conceptos:

Anomalía: Un cambio dentro de un patrón de datos, un valor atípico o un evento que se encuentre fuera de una tendencia estándar; una desviación de algo esperado o algo que no se ajusta a las expectativas. Una anomalía, o un valor atípico en un patrón, pueden indicar algo que está fuera de la norma o algo que posiblemente no esté bien. Ellas pueden ser puntuales/globales, contextuales o colectivas [CITATION Ser22 \l 3082].

Anomalías puntuales/globales: Un único punto de datos que se ha identificado como demasiado lejos del resto [CITATION Ser22 \l 3082].

Anomalías contextuales: Anomalía anormal en el contexto de un conjunto de datos, pero normal en el contexto de otro conjunto de datos. Este es el tipo más común de anomalía contextual en datos de series temporales [CITATION Ser22 \l 3082].

Anomalías colectivas: Cuando un subconjunto completo de datos es anómalo al compararlo con un conjunto más amplio de datos; los puntos de datos individuales no se tienen en cuenta cuando se identifican anomalías colectivas [CITATION Ser22 \l 3082].

Detección de anomalías: La identificación de un valor atípico raro o un punto de datos fuera de las tendencias de un conjunto de datos. Las anomalías pueden indicar eventos sospechosos, fallos, defectos o fraude [CITATION Ser22 \l 3082].

Fraude electrónico: El fraude electrónico o delito informático es una actividad indebida basada en la manipulación fraudulenta de elementos informáticos y sistemas de comunicación, para obtener un beneficio no autorizado [CITATION Tél04 \l 3082].

Dato: Un dato es la representación de una variable que puede ser cuantitativa o cualitativa que indica un valor que se le asigna a las cosas y se representa a través de una secuencia de símbolos, números o letras [CITATION Enc22 \l 3082].

Dato en informática: Es la expresión general que describe aquellas características de la entidad sobre la que opera. Los programas y aplicaciones tienen como función el procesamiento de datos, ya que cada lenguaje de programación tiene un conjunto da datos a partir de los cuales trabaja. Toda la información que entra y sale de un ordenador lo hace en forma de datos. Dentro de los archivos existen datos que son paquetes más pequeños de otros datos llamados registros (reunidos por características iguales o similares) [CITATION Enc22 \l 3082].

Tipos de datos [CITATION Enc22 \l 3082]: En programación es indispensable determinar a qué tipo o categoría corresponden los datos con los que se trabaja. Cada conjunto de datos de un tipo específico se manipula de diferente manera para obtener los resultados deseados.

- **Numérico » Entero.** Tipo de dato formado por una variable numérica que no cuenta con parte decimal.
- **Numérico » Real.** Tipo de dato formado por una variable numérica que puede contar con parte decimal.
- **Texto » Carácter.** Tipo de dato formado por una unidad o símbolo que puede ser una letra, un número, una mayúscula o un signo de puntuación.
- **Texto » Cadena.** Tipo de dato formado por un conjunto de caracteres dispuestos de forma consecutiva que se representa entre comillas.

- **Lógico » Boolean.** Tipo de dato que puede representar dos valores: verdadero o falso.

Flujo de datos: El curso que siguen las entradas de datos o información a través de un sistema hasta que se produce la salida [CITATION Ven22 \l 3082].

Computación distribuida: Se utiliza en numerosos ámbitos, ya que es un modelo informático que usa un *software* que permite hacer grandes cálculos y funciona repartiendo la información entre miles de ordenadores, que envían sus resultados a un mismo servidor [CITATION SAP22 \l 3082].

Pipeline: Es un esquema que interpreta un flujo constante de trabajo de forma secuencial, dando como entrada de cada proceso la salida del anterior de forma concatenada. Es el proceso según el cual, mientras una instrucción es ejecutada, otra está siendo interpretada por el ordenador y una más está siendo leída. Se trata de un conjunto de etapas que conforman la secuencia de ejecución de código en una CPU. Un mayor número de etapas supone poder hacer más cosas a la vez, pero también **«**trocear**»** las instrucciones en tareas más simples que requieren a su vez una mayor velocidad de reloj [CITATION Glo22 \l 3082].

Entidad Financiera: Una entidad financiera es una agrupación cuyo giro es ofrecer servicios financieros en el área de la banca, valores y seguros. Su oferta considera desde la intermediación, comercialización de seguros, créditos y asesoramiento, entre otros [CITATION BBV22 \l 3082].

La detección y prevención del fraude bancario, está magnificada dentro de un espectro de características y limitaciones [CITATION Tél04 \l 3082].

- Primeramente, se debe tener sumo cuidado de no bloquear incorrectamente tarjetas o cuentas genuinas o con el procesamiento de demasiadas transacciones legítimas.
- En segundo lugar, las instituciones financieras procesan un volumen inmenso de transacciones, de las cuales sólo un pequeño porcentaje es fraudulento (cerca del 0,1%).
- En tercer lugar, el número de transacciones que pueden ser revisadas por los investigadores de fraudes es limitado, por lo existe la necesidad de automatizar el proceso de detección.

Tras ser identificados los riesgos a los que se enfrentan las entidades financieras, es de vital importancia contar con herramientas informáticas. Dichas herramientas deben permitir la identificación de patrones de comportamiento inusuales y/o que corresponden a actividades potencialmente fraudulentas dentro del gran número de registros de transacciones.

1.2 Aplicación del Aprendizaje Automático en sistemas bancarios.

AA es una disciplina científica que maneja sistemas inteligentes, es decir que aprenden automáticamente al identificar ciertos patrones presentes en los datos [CITATION Moh18 \l 3082]. Para este aprendizaje:

- Usa algoritmos que se encargan de revisar datos mediante ejemplos o instrucciones predefinidas para así predecir comportamientos futuros permitiendo además la incorporación de información adicional y reajustar el resultado.
- Maneja conocimiento inductivo obteniendo un enunciado general en base a enunciados que describen casos particulares.

Los algoritmos de AA se clasifican generalmente en:

- **Aprendizaje supervisado (AS).** La máquina aprende no sólo de los propios datos finales (inputs), sino que es posible darle modelos o datos adicionales ya categorizados (outputs) para que el aprendizaje sea mucho más fiable [CITATION Mur12 \l 3082].
- **Aprendizaje no supervisado (ANS).** Sólo se dan los datos finales (inputs) a la máquina para que encuentre patrones interesantes a partir de esos datos. A diferencia del aprendizaje supervisado, el no supervisado utiliza procedimientos inductivos, extrayendo conocimiento sólo de los datos, como en el caso del análisis de clusters para la clasificación [CITATION Mur12 \l 3082].
- **Aprendizaje por refuerzo (AR).** En el aprendizaje por refuerzo el agente no cuenta con los datos de entrada y la respuesta esperada. En éste caso el algoritmo intenta obtener la mayor recompensa posible ante un determinado estado y una acción tomada para dicho estado. El agente debe descubrir que acciones le brindan la mayor recompensa ante determinado estado, una medida numérica, un número alto representa un mayor nivel de recompensa [CITATION Mer22 \l 3082].

A continuación se analizará cómo el AA puede ayudar a mejorar el sistema bancario [CITATION Ram22 \l 3082]:

- **Detección de fraudes:** A través de él se pueden detectar posibles fraudes en tiempo real. Se extraen datos y patrones de comportamiento que permiten identificar comportamientos anómalos o acciones sospechosas, en cuyo caso existe la posibilidad de solicitar una identificación del usuario extra como medida de seguridad. En este caso, el mayor valor de estas técnicas reside en que se realizan a tiempo real, de modo que se pueden evitar los fraudes y no solo descubrirlos a posteriori.

- **Predicción del riesgo crediticio:** El AA facilita esta compleja tarea, calculando estas variables de forma rápida y con gran precisión a través de la inteligencia artificial genera modelos de riesgo de créditos basados en datos financieros, comportamiento crediticio y consumo de los clientes. De esta forma se puede identificar cuándo incrementar o reducir la línea de crédito de un cliente en base a la aversión del banco al riesgo.

- **Análisis y segmentación de clientes:** El estudio y análisis de los datos es la mejor forma de conocer el comportamiento de los clientes, su nivel de afinidad, relación y cuáles son sus patrones de conducta en determinados segmentos, lo que permite diseñar estrategias o recomendar productos específicos a través de campañas de publicidad personalizadas. Es una herramienta muy útil para detectar si un cliente se plantea cambiar de entidad, analizando cómo ha dejado de utilizar los servicios financieros en el último periodo, para tomar acción e inclinar la balanza de nuevo a favor.

Modelos de Aprendizaje Automático.

- **Artificial Neural Network (ANN):** Este es un tipo de método de aprendizaje automático modelado en el cerebro y el sistema nervioso. Utilizando información histórica, sus diseños de ANN pueden descubrir las tendencias y también pueden clasificar los datos entrantes. Se hace uso de este algoritmo con LR para la detección de fraude bancario en el artículo *"Detecting credit card fraud by ANN and logistic regression, 2011" [CITATION YSa11 \l 3082].*
- **Decision Trees (DT):** El árbol de clasificación marca, registra y asigna factores de clase separados y es construido por un proceso llamado particionado recursivo binario. El árbol de clasificación puede proporcionar una medida que tal vez vuelva la categoría precisa. Mencionado en el artículo *"Credit Card Fraud Detection Techniques. A Survey, 2020"* como la herramienta más popular y potente para la clasificación y predicción [CITATION NSh201 \l 3082].
- **Gradient Boosting (GBC):** Es un algoritmo prominente de aprendizaje automático, siempre tuvo que llevar a cabo actividades de clasificación y regresión. El modelo anterior consiste en una cantidad en diseños de conjuntos fundamentales, como árboles de decisión débiles. Tal que los árboles de decisión se combinan para crear un poderoso modelo especular que aumenta la reflexión. Se utiliza este algoritmo que posee buenos resultados en la detección de fraude bancario en el artículo *"An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine, 2020"[CITATION AAT20 \l 3082].*
- **Isolation Forest (IF):** Es especialmente diseñado para la detección de anomalías sin importar el tamaño del conjunto de datos, consiste en la creación de varios árboles donde se van particionado al mismo tiempo que se van clasificando. Se utiliza en un sistema junto a *Local Outlier Factor* (LOF) para la detección de fraude en el artículo *"Isolation Forest and Local Outlier Factor for Credit Card Fraud Detection System, 2020"* [CITATION VVi20 \l 3082].
- **K-Nearest Neighbors (KNN):** Un clasificador de K-vecinos más cercanos parece ser un aprendizaje automático supervisado directo y fácil de implementar. Algoritmo que podría usarse para abordar la clasificación respectivamente, así como dificultades de regresión. Centrados en la predicción de fraude de tarjetas de crédito, en el documen-

to *“Credit card fraud detection using Naïve Baiyes model based on KNN classifier, 2018”* [CITATION Kir18 \l 3082]es utilizado este algoritmo como clasificador junto a un modelo NBC. De manera similar se puede apreciar en el artículo *“Machine Learning Approach for Credit Card Fraud Detection (KNN and Naïve Baiyes)-(March 30, 2020)” [CITATION Kau20 \l 3082]*.

- **Logistic Regression (LR):** Es otro método que se decidió tomar prestado de la profesión de datos estadísticos por aprendizaje automático. También es el proceso de referencia de problemas relacionados con la categorización binaria (dificultades con más de dos valores morales de clase). La regresión logística se usa para modelar el resultado como aprobación real/rechazo total, positivo y constructivo/negativo o neutral y en los casos de detección de amenazas de fraude con tarjetas de crédito, entonces usamos clasificación de distribución de probabilidad como fraudulenta y no fraudulenta. En el artículo *“Relative Analysis of ML Algorithm QDA, LR and SVM for Credit Card Fraud Detection Dataset, 2020”* se usa este algoritmo para la extracción de características, junto a *Quadratic Discriminant Analysis* (QDA) y SVM [CITATION PNa20 \l 3082].
- **Naïve Baiyes Classifier (NBC):** Este es de hecho un proceso estadístico basado en teoría predictiva que selecciona su mejor decisión centrado en la probabilidad. Resultados no identificados de sistemas de valor reconocidos han sido estimados por verosimilitud bayesiana. Esto también permite la implementación de conocimientos previos, así como la lógica en afirmaciones impredecibles. Esa primera metodología parece tener un carácter jurídicamente vinculado a la presunción de independencia en torno a las características a lo largo de los datos. En los siguientes documentos *“Credit card fraud detection using machine learning techniques, 2017” [CITATION JOA17 \l 3082]* y *“Credit card fraud detection using Naïve Baiyes model based on KNN classifier, 2018” [CITATION Kir18 \l 3082]* demuestran la efectividad de este clasificador en la detección de fraude bancario. En el primero se realiza un análisis del uso de un clasificador binario basado en reglas de clasificación bayesiana y en el segundo se utiliza un modelo NBC con un clasificador K-Nearest Neighbors.
- **Random Forest (RF):** Esta es una de las metodologías de conjunto utilizado para mejorar la prosperidad y precisión en algoritmos de aprendizaje automático de inteligencia artificial. Sugerido por

el investigador Breiman, el método del bosque aleatorio también puede ayudar a identificar las variables independientes realmente apropiadas para que el sistema pueda elegir la funcionalidad. Además, muchos hallazgos ya demuestran su importancia en la selección de varias posibilidades para cada arbusto, pero en la investigación empírica, se descubre que esto también es óptimo con respecto a la precisión del pronóstico. Obteniendo cerca de un 90% de exactitud en el artículo *"Credit Card Fraud Detection Using Random Forest Algorithm, 2019"* se puede apreciar cómo se utiliza este algoritmo para la detección del fraude bancario [CITATION MSK19 \l 3082].

- **Support Vector Machine (SVM):** Es un modelo supervisado, usa algoritmos de clasificación de aprendizaje para clasificar problemas importantes incluso en dos grupos e individuales. Realmente pueden clasificar un nuevo documento desde que se le da el número del conjunto de datos etiquetados a cada clasificación en un sistema SVM. Haciendo uso de este modelo junto al algoritmo de DT en el artículo *"Fraudulent Detection in Credit Card System Using SVM and Decision Tree"* [CITATION Vij \l 3082] se usa en un sistema para la detección fraudulenta y también en el artículo *"Relative Analysis of ML Algorithm QDA, LR and SVM for Credit Card Fraud Detection Dataset, 2020"* junto a otros algoritmos con el mismo propósito [CITATION PNa20 \l 3082].

Balanceo de datos [CITATION Cal21 \l 3082]:

Dentro de la clasificación de aprendizaje automático el desbalanceo de datos es un fenómeno bastante común. El mismo consiste en la distribución de las clases del conjunto de entrenamiento cuando estas se encuentran muy desproporcionadas. La clase mayoritaria es aquella que contiene la mayoría de muestras con la misma salida (etiqueta) o con el mismo resultado de la clasificación. De manera contraria la clase minoritaria contiene un subconjunto de muestras significativamente menor en comparación al resto.

Un conjunto de datos se considera desequilibrado o desbalanceado cuando dentro de él existen clases la cuales presentan una distribución desequilibrada en la cantidad de muestras dentro de cada una de las clases que lo conforman. De manera general cuando abordamos este tema se considera que el conjunto se compone únicamente por dos clases de datos, pero el problema del desequilibrio puede estar dado para conjuntos de datos de clasificación multi-clase.

Este tipo de problema está presente en multitud de campos y sectores y por supuesto, esto incluye los servicios financieros. El reto aparece cuando los algoritmos de AA intentan identificar estos casos raros en conjuntos de datos de gran volumen. Debido a la disparidad de cla-

ses el algoritmo tiende a categorizar en la clase mayoritaria dando la falsa sensación de un modelo preciso. Tanto la incapacidad de predecir eventos raros (clase minoritaria) como la precisión engañosa restan valor a los modelos de predicción que se construyen al día de hoy. Los modelos entrenados en conjuntos de datos desequilibrados suelen tener malos resultados cuando tienen que generalizar (predecir una clase o clasificar observaciones no vistas). Sin importar el algoritmo elegido, algunos modelos serán más susceptibles a los datos desbalanceados que otros.

En última instancia, significará que no se tendrá un buen modelo ya que el algoritmo recibe muchos ejemplos de una clase, lo que lo hace sesgado a dicha clase en particular. No aprende lo que diferencia a la otra clase y no entiende los patrones subyacentes para diferenciar las clases. Esto se convierte en un problema cuando esta propiedad afecta al rendimiento de los algoritmos o a los modelos que se pueden obtener.

Estrategias para el manejo de datos desbalanceados en Aprendizaje Automático [CITATION Jua22 \l 3082].

- **Ajuste de Parámetros del modelo:** Consiste en un ajuste de los parámetros o métricas del propio algoritmo, intenta equilibrar durante el entrenamiento a la clase minoritaria penalizando a la clase mayoritaria. Todos los algoritmos no cuentan con estas posibilidades. En ANN por ejemplo se ajusta la métrica "*Loss"* para la penalización a las clases mayoritarias.
- **Modificar el conjunto de datos:** Para hacer una reducción e intentar equilibrar la situación podemos eliminar muestras de la clase mayoritaria. Existe la amenaza de empeorar el modelo al prescindir de muestras importantes, que brindan información. Entonces deberíamos seguir algún criterio para seleccionar qué muestras eliminar.
- **Muestras artificiales:** Haciendo uso de diversos algoritmos que intentan seguir la tendencia del grupo minoritario se intenta crear muestras sintéticas (no idénticas). Se puede lograr una mejora de los resultados, en dependencia del método. Lo peligroso de esta estrategia es que podemos alterar la distribución "natural" de esa clase y confundir al modelo en su clasificación.
- **Métodos de conjunto equilibrados:** Se apoya en las ventajas de hacer un ensamble de métodos. Se asegura de tomar muestras de entrenamiento equilibradas al entrenar

diversos modelos y entre todos obtener el resultado final (Ejemplo: “mediante una votación”).

Aprendizaje Automático en Flujo de Datos [CITATION Ram19 \l 3082]:

En la actualidad, existen numerosas aplicaciones que constantemente están generando una cantidad inmensa de información. Las técnicas tradicionales de minería de datos realizan un aprendizaje por lotes (Batch Learning), concentrados en encontrar conocimiento en repositorios estáticos; no obstante, debido a las propiedades inherentes en los datos originados por dichas aplicaciones, este tipo de técnicas no puede ser aplicado a estos datos.

Primeramente, no es factible ni tampoco práctico guardar tanta información en bases de datos, puesto que estas poseen un tamaño fijo y en estos casos la cantidad de información originada puede llegar a ser infinita. Característica inabordable por repositorios tradicionales a la hora de entrenar modelos cuyos datos deben estar en la memoria principal, la cual posee poca capacidad de almacenamiento. Por otra parte estas aplicaciones generan información a gran velocidad y, a diferencia de los algoritmos habituales, los patrones que subyacen a dicha información pueden cambiar dinámicamente debido al entorno no estacionario donde se originan. De manera que es necesario que las técnicas de AA sean capaces de construir modelos que de forma continua se adapten a dichos cambios para mantener un buen rendimiento.

Los modelos generados por los algoritmos de AA deben estar actualizados a medida que se van originando nuevos datos para que ofrezcan un buen desempeño. Deben tener en cuenta una serie de restricciones temporales, las cuales no pueden ser llevadas a cabo sí, a la hora de entrenarlos, realizamos varios ciclos de lectura de los datos.

Por tanto, los sistemas modernos deben tener en cuenta la rapidez y la continuidad con la que se generan los datos hoy en día. Dada las propiedades de estos datos, éstos reciben el nombre de flujo de datos y dada la importancia de extraer conocimiento a partir de este tipo de datos, en los últimos años se han realizado una gran cantidad de investigaciones en el campo del AA aplicado a FD.

A la hora de desarrollar algoritmos de AA para el manejo de FD, teniendo en cuenta los problemas que presentan los algoritmos tradicionales, deben asumir una serie de desafíos y restricciones:

- Las instancias del flujo de datos deben ser procesadas una sola vez.

- No hay control sobre el orden en el que los objetos de datos deben ser procesados.
- El tamaño de un FD se debe suponer que es ilimitado.
- El proceso de generar un FD puede ser no estacionario.
- La memoria utilizada por los algoritmos es limitada.
- El trabajo de realizado por los algoritmos debe cumplir unas restricciones estrictas de tiempo.
- El modelo inducido por los algoritmos debe llevar a cabo tareas de predicción en cualquier momento.
- Pueden aparecer nuevas clases que requieran ser modeladas para un buen desempeño del modelo.
- Pueden ocurrir, al igual que en las tareas de clasificación, problemas relacionados con valores faltantes, sobreajuste del modelo, variables irrelevantes y redundantes y desbalanceo de las clases.

Para hacer frente a estos problemas en FD, existen tres aproximaciones principales:

- Entrenar un clasificador cada vez que se disponga de nuevos datos. Opción poco adoptada por poseer altos costes computacionales.
- Detectar cambios en los patrones de los datos (CD: Concept Drifts), de manera que si son relevantes, se vuelve a entrenar el modelo tras la ocurrencia del CD.
- Llevar a cabo un aprendizaje incremental con el objetivo de adaptar el modelo a los cambios en el concepto subyacente de los datos de manera gradual.

Para la comprensión de estos algoritmos, es fundamental tener una idea general de las nociones sobre la que se basan.

Concept Drift:

Uno de los desafíos a los que debe hacer frente el AA para lidiar con flujos de datos es que la distribución subyacente en los datos puede cambiar durante el transcurso del tiempo (distribución no estacionaria). Este fenómeno se denomina CD, de tal forma que la palabra *"concept"* se refiere al concepto que describe y está inherente en los datos.

Métricas para comparar modelos en AA:

- **F1 Score:** es dada por la media harmonía de precision y recall, combina ambas métricas en una sola, donde la puntuación de la F1 alcanza su mejor valor en 1 (precision y recall perfectas) y el peor en 0.
- **Sensitivity:** La sensibilidad se refiere a la tasa positiva verdadera y resume qué tan bien se predijo la clase positiva.
- **Specificity:** La especificidad es el complemento de la sensibilidad, o la verdadera tasa negativa, y resume qué tan bien se predijo la clase negativa.

Para la clasificación desequilibrada, la sensibilidad podría ser más interesante que la especificidad.

- **G-Mean:** La sensibilidad y la especificidad se pueden combinar en una sola puntuación que equilibra ambas preocupaciones, denominada media geométrica o G_Mean.
- **ROC AUC:** Aunque generalmente son efectivos, la curva ROC y el AUC de ROC pueden ser optimistas bajo un desequilibrio de clase severo, especialmente cuando el número de ejemplos en la clase minoritaria es pequeño.

Soluciones informáticas para la detección del fraude en transacciones bancarias. Fundamentos que se constituyeron referentes de la investigación tanto en el contexto nacional como internacional.

Algoritmo genético para la detección de fraude electrónico en tarjetas de débito en Perú.

Sigue el enfoque Iterative Rule Learning (IRL). Según esta estrategia propone un modelo heurístico basado en el comportamiento transaccional de los clientes y la determinación de los patrones de desviación que sean catalogadas como sospechosas, para ello se emplean técnicas basadas en algoritmos genéticos.

Como objetivo persigue maximizar la precisión de la predicción frente a los datos reales. Los objetivos específicos serán:

- La minimización de los falsos positivos que son aquellas transacciones que el sistema deja pasar aún siendo fraudulentas.
- la minimización de los falsos negativos que son aquellas en que el sistema devuelve una transacción como sospechosa siendo una operación válida.

Utiliza como paradigmas de detección de fraudes la fusión de la teoría *Dempster-Shafer* y *Bayesian Learning*, así como el modelo de detección *Fuzzy Darwinian* [CITATION Lav13 \l 3082].

Análisis de fraudes en transacciones bancarias aplicando Minería de Datos.

Se realiza un análisis para la detección de fraudes con un conjunto de datos financieros sintéticos obtenido por el simulador *"PaySim"* que se recopiló del trabajo de [CITATION Lóp16 \l 3082], en la sección de Conjunto de datos se presenta la estructura del conjunto de datos y su preprocesamiento. Sin embargo, existe el cuestionamiento si es suficiente trabajar con datos sintéticos en lugar de datos reales [CITATION Víc19 \l

3082].

Esto es una preocupación primordial para cualquier investigador que desee realizar pruebas científicas, pero no tiene o tiene acceso limitado a un conjunto de datos financieros reales. El análisis que se presenta en el artículo se centra en el tipo de transacción que representa un fraude en el conjunto de datos, estos tipos de transacción son cinco: CASHIN, CASHOUT, DEBIT, PAYMENT y TRANSFER.

- CASH-IN es el proceso de aumentar el saldo de la cuenta mediante el pago en efectivo a un comerciante.
- CASHOUT es el proceso opuesto de CASH-IN, significa retirar dinero de un comerciante, lo que disminuye el saldo de la cuenta.
- DEBIT es un proceso similar al de CASH-OUT e implica enviar el dinero del servicio de dinero móvil a una cuenta bancaria.
- El PAYMENT es el proceso de pago de bienes o servicios a los comerciantes que disminuye el saldo de la cuenta y aumenta el saldo del receptor.
- TRANSFER es el proceso de enviar dinero a otro usuario del servicio a través de la plataforma de dinero móvil.

Algoritmo de Random Forest aplicando a la detección de fraude en el sistema bancario ecuatoriano.

Lo que se propone en este estudio, es el uso de una nueva técnica para detectar el fraude crediticio en las entidades bancarias, técnica no utilizada actualmente en Ecuador y que genera mejores y correctos resultados en comparación a las técnicas utilizadas habitualmente, de acuerdo a las investigaciones de [CITATION Hid14 \l 3082] y [CITATION Zha08 \l 3082].

Es importante recalcar, que no se trata de construir un modelo que detecte a la perfección el fraude (aunque sí será robusto gracias la eficacia misma de la nueva técnica propuesta), sino más bien, hacer notar la existencia de una técnica estadística moderna y confiable, que supera apropiadamente las dificultades presentadas en el problema del fraude, con la ayuda de un método de remuestreo. Además, la exposición de esta nueva herramienta brindará a las entidades bancarias una solución a diversos problemas con características similares al problema de fraude. Con la finalidad de mostrar su gran poder predictivo se realizará una comparación con la regresión logística con estimación a través del método de *“Firth”,* la cual ha resultado ser mejor que cualquier otro método tradicional en problemas de detección del fraude [CITATION EAn19 \l 3082].

Algoritmos de aprendizaje automático para detección de fraudes con tarjetas de crédito: Análisis y comparativa [CITATION Cal21 \l 3082].

Aborda el estudio de las técnicas existentes para tratar con conjuntos de datos desequilibrados y la detección de fraudes con tarjetas de crédito mediante algoritmos de aprendizaje automático. Para ello, se lleva a cabo un análisis comparativo de las técnicas de submuestreo (aleatorio) y sobremuestreo (SMOTE) de datos y, tras balancear la muestra, se implementan cuatro algoritmos clasificadores clásicos, a saber: LR, KNN, SVM y DT. Por último, se desarrollan dos redes neuronales profundas (una para cada método de balanceo) cuyos resultados se comparan con el algoritmo que mayor rendimiento haya ofrecido en sus predicciones.

Este trabajo de fin de grado (TFG) presenta una síntesis de aspectos teóricos y comparativos de algoritmos y técnicas tanto alternativas como complementarias, que denotan el valor teórico que posee este trabajo. La elaboración de este TFG se justifica también desde el punto de vista del valor económico aportado, pues este está orientado a reducir los costes por el beneficio que supondría para las empresas conseguir un sistema seguro de detección anti-fraude en este tipo de pagos.

1.3 Metodología de la Minería de Datos.

Minería de Datos (MD) también conocida como KDD (Knowledge Discovery in Databases por sus siglas en inglés). Es definida comúnmente como el proceso para descubrir patrones útiles o conocimientos a partir de fuentes de datos tales como Bases de Datos, textos, imágenes, la web, etc. Los patrones deben ser válidos, potencialmente útiles y entendibles. La minería de datos es un campo multidisciplinario que incluye: aprendizaje automático, estadísticas, sistemas de base de datos, inteligencia artificial, recuperación de información, visualización de la información, etc [CITATION ITE22 \l 3082].

El objetivo general del proceso de minería de datos consiste en extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior. Existen muchas técnicas y tareas dentro de DM. Algunas de las más comunes consisten en el aprendizaje supervisado, aprendizaje no supervisado, minería de asociación de reglas y minería de secuencia. En resumen, MD es el conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto [CITATION ITE22 \l 3082].

1.3.1 Selección de la metodología de Minería de Datos.

Metodologías dominantes para el proceso de la minería de datos [CITATION ELG05 \l 3082]:

- Knowledge Discovery in Databases (KDD): Es una metodología propuesta por Fayyad en 1996, propone 5 fases: Selección, preprocesamiento, transformación, minería de datos y evaluación e implantación. Es un proceso iterativo e interactivo.
- SEMMA: Acrónimo a las cinco fases: (Sample, Explore, Modify, Model, Assess). La metodología es propuesta por SAS Institute Inc, la define como: “… proceso de

selección, exploración y modelamiento de grandes cantidades de datos para descubrir patrones de negocios desconocidos…"

- Cross-Industry Standard Process for Data Mining (CRISP-DM): Iniciativa financiada por la Comunidad Europea la cual se ha unido para desarrollar una plataforma para Minería de Datos. Persigue como objetivos: Fomentar la interoperabilidad de las herramientas a través de todo el proceso de minería de datos y eliminar la experiencia misteriosa y costosa de las tareas simples de minería de datos.

En este aspecto varias son las metodologías que posibilitan el proceso de MD pero para los oficios de estos estudios se propone aplicar la metodología KDD la cual consiste en un proceso organizado de identificación válida mediante patrones entendibles de un gran y complejo conjunto de datos. La misma realiza un análisis exploratorio, modelado automático de grandes repositorios de datos [CITATION Mai10 \l 3082].

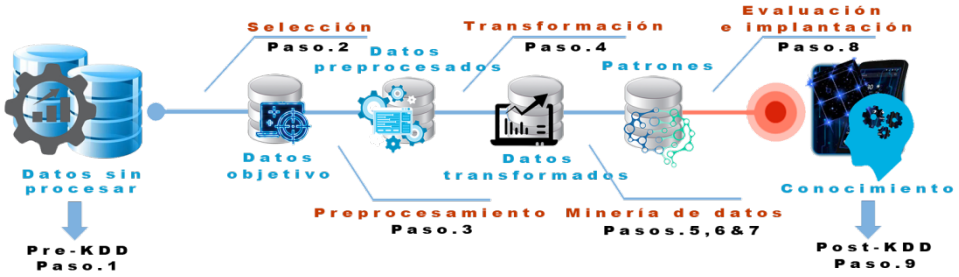


Figura 1. Secuencia KDD. Elaboración propia.

Pasos de KDD para orientar el desarrollo del proyecto.

- ✓ Paso 1. Entendimiento, conocimientos previos e identificación de la meta.
- ✓ Paso 2. Selección de un conjunto de datos.
- ✓ Paso 3. Limpieza y preprocesamiento de datos.
- ✓ Paso 4. Transformación de los datos.
- ✓ Paso 5. Colocar el objetivo del KDD a un método de MD.
- ✓ Paso 6. Elección de los algoritmos de MD.
- ✓ Paso 7. Implementación de los algoritmos de MD.
- ✓ Paso 8. Evaluación.
- ✓ Paso 9. Aplicación del conocimiento adquirido.

1.4 Tecnologías y herramientas.

Plataformas y servicios clave de transmisión de datos [CITATION KWF22 \l 3082]:

- **Amazon Kinesis:** Permite recopilar, procesar y analizar datos de transmisión en tiempo real a escala. Tiene tres servicios para datos (Data Streams, Data Firehose y Data Analytics) y uno para medios (Video Streams). Kinesis Data Streams es un servicio de ingestión que puede capturar continuamente gigabytes de datos por segundo de miles de fuentes. Kinesis Data Analytics puede procesar flujos de datos en tiempo real con SQL o Apache Flink. Kinesis Data Firehose puede capturar, transformar y cargar flujos de datos en almacenes de datos de AWS para realizar análisis casi en tiempo real con las herramientas de inteligencia empresarial existentes. Puede usar las funciones sin servidor de AWS Lambda en lugar de Kinesis Data Analytics si desea procesar la transmisión con un programa en lugar de usar SQL o Flink.

- **Apache Flink:** Es un marco Java/Scala/Python de código abierto y un motor de procesamiento distribuido para cálculos con estado sobre flujos de datos ilimitados y limitados. Flink ha sido diseñado para ejecutarse en todos los entornos de clúster comunes y realizar cálculos a la velocidad de la memoria y en cualquier escala. Flink se integra con administradores de recursos de clúster comunes, como Hadoop YARN, Apache Mesos y Kubernetes, pero también puede ejecutarse como un clúster independiente.

- **Apache Kafka:** Es una plataforma de transmisión de eventos distribuidos Java/Scala de código abierto para canalizaciones de datos de alto rendimiento, análisis de transmisión, integración de datos y aplicaciones de misión crítica. Los eventos de Kafka se organizan y almacenan de forma duradera en temas. Kafka se desarrolló originalmente en LinkedIn y actualmente tiene la mayor parte del mercado de transmisión de eventos, incluida la versión comercial de Confluent.

- **Apache Pulsar:** Es una plataforma de streaming y mensajería pub-sub distribuida de Java/C++/Python nativa de la nube y de código abierto. Pulsar fue desarrollado originalmente en Yahoo.

- **Apache Samza:** Es un marco de procesamiento de flujo Scala/Java de código abierto distribuido que se desarrolló originalmente en LinkedIn, junto con (Apache) Kafka. Samza le permite crear aplicaciones con estado que procesan datos en tiempo real desde múltiples fuentes, incluido Apache Kafka.

- **Apache Spark:** Es un motor multilingüe, escrito principalmente en Scala, para ejecutar ingeniería de datos, ciencia de datos y aprendizaje automático en clústeres o máquinas de un solo nodo. Maneja tanto datos por lotes como datos de transmisión en tiempo real. Spark se originó en UC Berkeley, y los autores de Spark fundaron Databricks.

- **Apache Storm:** Es un marco de computación de procesamiento de flujo distribuido escrito principalmente en Clojure. En Storm, un *flujo* es una secuencia ilimitada de tuplas que se procesa y crea en paralelo de forma distribuida. Una *topología* es un gráfico de *picos* y *pernos* que están conectados con agrupaciones de flujo; Las topologías definen la lógica que procesa los flujos. Un *spout* es una fuente de flujos en una topología. Todo el procesamiento en topologías se realiza en *pernos* . Storm se integra con muchos otros sistemas y bibliotecas, incluidos Kafka, Cassandra, Redis y Kinesis.

- **Azure Stream Analytics:** Es el servicio recomendado para el análisis de flujos en Azure. Agregar un trabajo de Azure Stream Analytics a la aplicación es la manera más rápida de obtener análisis de flujos en funcionamiento en Azure mediante el lenguaje SQL que ya conoce. Azure Stream Analytics es un servicio de trabajo, por lo que no tiene que perder el tiempo administrando clústeres, ni preocuparse por el tiempo de inactividad, con un SLA del 99,9% en el nivel de trabajo. La facturación también se realiza en el nivel de trabajo, lo que hace que los costos iniciales sean bajos (una unidad de streaming), pero escalables (hasta 192 unidades de streaming). Es mucho más rentable ejecutar algunos trabajos de Stream Analytics que ejecutar y mantener un clúster. Azure Stream Analytics ofrece una experiencia muy completa de fábrica.

La propuesta planteada se intenta implementar para las operaciones cuya autorización ha sido emitida y propone un modelo de AA corriendo sobre componentes basados en herramienta estándar del ecosistema de Apache: Kafka_2.13 v 3.1.0.

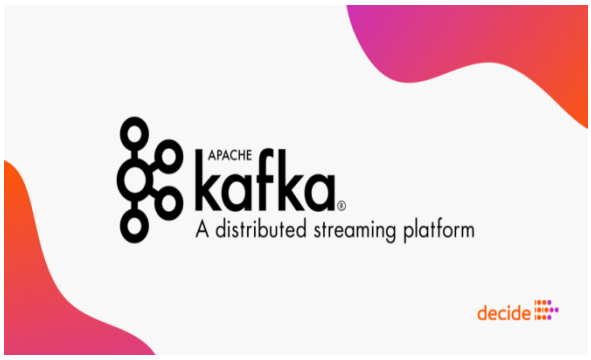


Figura 2. Apache Kafka.

Apache: Kafka_2.13. Versión 3.1.0 [CITATION Red22 \l 3082]: Apache Kafka es una plataforma distribuida de transmisión de datos que permite publicar, almacenar y procesar flujos de registros, así como suscribirse a ellos, de forma inmediata. Está diseñada para administrar los flujos de datos de varias fuentes y distribuirlos a diversos usuarios. En pocas palabras, transfiere cantidades enormes de datos, no solo desde el punto A hasta el B, sino también del punto A al Z y a cualquier otro lugar que necesite, y todo al mismo tiempo. Apache Kafka es la alternativa a un sistema de mensajería tradicional para empresas. Comenzó como un sistema interno que LinkedIn desarrolló para gestionar 1,4 billones de mensajes por día. Ahora, es una solución código abierto de transmisión de datos que permite satisfacer diversas necesidades empresariales.

Descargas binarias: Construimos para múltiples versiones de Scala. Esto solo importa si está utilizando Scala y desea una versión creada para la misma versión de Scala que utiliza. De lo contrario, cualquier versión debería funcionar (se recomienda la 2.13).

Kafka 3.1.0 incluye una serie de características nuevas importantes. Aquí hay un resumen de algunos cambios notables:

- Apache Kafka es compatible con Java 17.

- FetchRequest admite ID de tema (KIP-516).
- Ampliar SASL/OAUTHBEARER con soporte para OIDC (KIP-768).
- Agregar métricas de conteo de intermediarios (KIP-748).
- Diferenciar la latencia métrica consistentemente medida en milisegundos y nanos (KIP-773).
- El protocolo de reequilibrio ansioso está en desuso (KAFKA-13439).
- Agrega el campo TaskId a StreamsException (KIP-783).
- Particionadores personalizados en uniones de clave externa (KIP-775).
- Consultas Fetch/findSessions con extremos abiertos para SessionStore / WindowStore (KIP-766).
- Consultas de rango con puntos finales abiertos (KIP-763).
- Agrega la métrica de tiempo total bloqueado a Streams (KIP-761).
- Agrega configuración adicional para controlar la convención de nomenclatura de temas internos de MirrorMaker2 (KIP-690).

Lenguajes de programación:

Los lenguajes de programación que se pueden utilizar para aplicaciones de AA son R, C ++, JavaScript, Java, C #, Julia, Shell, TypeScript, Scala y Python.

- **R:** Es un ambiente de programación formado por un conjunto de herramientas muy flexibles que pueden ampliarse fácilmente mediante paquetes, librerías o definiendo nuestras propias funciones**.** Además es gratuito y de código abierto, un Open Source parte del proyecto GNU, como Linux o Mozilla Firefox [CITATION Máx22 \l 1033].
- **C++:** Es de alto nivel, siendo un lenguaje ensamblador e híbrido, altamente didáctico. C++ colabora en que los programadores puedan escribir otros programas portátiles y rápidos. Unix, por ejemplo, es un sistema operativo escrito en Lenguaje C. Tiene compatibilidad con bibliotecas enriquecidas y se caracteriza por su velocidad en comparación a otras opciones disponibles [CITATION edX22 \l 1033].
- **JavaScript:** Es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla [CITATION MDN22 \l 1033].
- **Java:** Es un lenguaje multiplataforma, orientado a objetos y centrado en la red que se puede utilizar como una plataforma en sí mismo. Es un lenguaje de programación rápido, seguro y confiable para codificar todo, desde aplicaciones móviles y software em-

presarial hasta aplicaciones de big data y tecnologías del lado del servidor [CITATION Ama22 \l 1033].

- **C#:** Es un lenguaje de programación orientado a componentes, orientado a objetos. C# proporciona construcciones de lenguaje para admitir directamente estos conceptos, por lo que se trata de un lenguaje natural en el que crear y usar componentes de software. Desde su origen, C# ha agregado características para admitir nuevas cargas de trabajo y prácticas de diseño de software emergentes [CITATION Mic22 \l 1033].
- **Julia:** Es un lenguaje de tipo dinámico que se puede usar fácilmente de forma interactiva. Está desarrollado como un lenguaje de programación de alto rendimiento. Usa envío múltiple (“multiple dispatch” en inglés), que le permite al programador elegir entre diferentes patrones de programación de acuerdo a la aplicación. Tiene una sintaxis de alto nivel que es fácil de aprender. Julia es un lenguaje de programación con tipos opcionales, cuyos tipos de datos (definidos por el usuario) hacen que el código sea claro y robusto. Tiene una biblioteca estándar extendida, además, están disponibles numerosos paquetes de terceros [CITATION All22 \l 1033].
- **Shell:** Es un lenguaje de control, intérprete y lenguaje de programación, tiene las características que lo hacen sumamente flexibles para las tareas de un centro de cómputo e incluye: Características de control normales (secuenciación, iteración condicional, selección y otras más), paso de parámetros y comunicación entre ordenes de Shell. Shell permite modificar en forma dinámica las características con que se ejecutan los programas en Unix: la salida estándar puede direccionarse a un archivo, a un proceso ó a un dispositivo, asimismo, es posible interconectar entre sí, y a la vez usuarios pueden ver versiones "distintas" de Sistema Operativo debido a la capacidad del Shell para configurar diversos ambientes de trabajo [CITATION MNV22 \l 1033].
- **TypeScript (TS):** Es un lenguaje de programación construido a un nivel superior de JavaScript (JS). Esto quiere decir que TypeScript dota al lenguaje de varias características adicionales que hacen que podamos escribir código con menos errores, más sencillo, coherente y fácil de probar, en definitiva, más limpio y sólido [CITATION Pro22 \l 1033].

- **Scala:** Es un lenguaje de programación de propósito general, diseñado para el desarrollo utilizando patrones de una forma concisa, elegante y con tipos. Es de código abierto, integra principios de orientación a objetos y la programación funcional, permitiendo a los programadores ser más productivos y aprovechar los conocimientos y estructuras de otros lenguajes como Java [CITATION Kee22 \l 1033].
- **Python:** Python es un lenguaje donde su código se ejecuta en el navegador al cargar la página, es independiente de la plataforma y orientado a objetos, está listo para realizar cualquier tipo de programa desde aplicaciones de Windows hasta servidores de red o incluso páginas web. Es un lenguaje interpretado, lo que ofrece ventajas como la velocidad de desarrollo e inconvenientes como una velocidad más baja al ser ejecutado [CITATION MÁl22 \l 1033].

Según [CITATION MÁl22 \l 3082] Python es sin duda el mejor lenguaje de programación para aplicaciones de aprendizaje automático. Para el desarrollo de este proyecto se propone el uso del lenguaje de programación Python en su versión 3.7.8, este lenguaje lleva un destacado recorrido en el área del AA.

Figura 3. Lenguaje Python.

Python versión 3.7.8 ofrece las siguientes mejoras del modelo de datos de Python:

- ✓ PEP 562, personalización de acceso a los atributos del módulo.
- ✓ PEP 560, soporte básico para módulo de escritura y tipos genéricos.
- ✓ Se ha declarado que la naturaleza de preservación del orden de inserción de los objetos dict es una parte oficial de la especificación del lenguaje Python.

Características de Python para el desarrollo de Aprendizaje Automático [CITATION Man20 \l 1033]:

- La asociación Python de AA: se ha visto favorecida por aplicaciones que van desde el desarrollo web hasta la automatización de scripts y procesos. Amplia selección de bibliotecas y marcos: Uno de los aspectos que hace que Python sea una opción tan popular en general es su abundancia de bibliotecas

y macros que facilitan la codificación y ahorran tiempo en el desarrollo.

- Código legible y conciso: facilidad de uso y simplicidad, especialmente para los nuevos desarrolladores.
- Agilidad: La sintaxis simple de Python significa que también es más rápido en desarrollo que muchos lenguajes de programación y permite al desarrollador probar algoritmos rápidamente sin tener que implementarlos.
- Colaboración: fácil de leer es de gran valor para la codificación cooperativa, o cuando los proyectos de Python de AA cambian de manos entre los equipos de desarrollo.
- Python es un lenguaje de programación de código abierto y está respaldado por una gran cantidad de recursos y documentación de alta calidad.

Bibliotecas de Python para Aprendizaje Automático.

Una de las grandes ventajas que ofrece Python sobre otros lenguajes de programación; es cuán grande y proliferante es la comunidad de desarrolladores que lo rodea; comunidad que ha contribuido con una gran variedad de librerías de primer nivel que amplían las funcionalidades del lenguaje. En el caso de AA, las principales bibliotecas que se utilizan son las descritas en la Tabla [CITATION Man20 \l 1033].

Librerías del proyecto:

Librería	Aplicación o Uso
Pandas	Paquete de Python que provee una estructura diseñada de datos rápida, flexible y expresiva para hacer trabajo con la clasificación de datos.
Matplotlib	Librería comprensiva para la agrupación estática, animada y visualizaciones interactivas en Python.
Seaborn	Seaborn es una biblioteca de visualización de datos de Python utilizada para hacer gráficos estadísticos.
Scikit-Learn/Sklearn	Se utiliza para clasificaciones, extracción de características, regresiones, agrupaciones, reducción de dimensiones, selección de modelos o preprocesamiento.
Kafka	Una biblioteca de código abierto basada en la comunidad. Entre sus dependencias debe incluir Zookeeper y Kafka-Python.
Pickle	Implementa protocolos binarios para serializar y deserializar una estructura de objetos de Python.
Colorama	Hace que las secuencias de caracteres de escape ANSI (para producir texto de terminal coloreado y posicionamiento del cursor) funcionen en MS Windows.
json	Python tiene un paquete incorporado llamado json,

	que se puede usar para trabajar con datos JSON.
Config	Permite un esquema de configuración jerárquica con soporte para mapeos y secuencias, la capacidad de acceder de manera flexible a objetos Python reales sin eval() en toda regla.
incremental_trees	Agrega el método de ajuste parcial a los estimadores de bosque de sklearn para permitir el entrenamiento incremental sin estar limitado a un modelo lineal.
Imblearn	Imbalanced-learn (importado como imblearn) es una biblioteca de código abierto con licencia del MIT que se basa en scikit-learn (importado como sklearn) y proporciona herramientas cuando se trata de la clasificación con clases desequilibradas.

Tabla 1. Librerías de Python para el proyecto.

IDE's para Aprendizaje Automático.

Para programar AA se necesita una plataforma de alto rendimiento para ejecutar el código, aquí hay una lista completa de los mejores IDE's de Python [CITATION Apr22 \l 1033]:

- **Spyder:** Es ligero y capaz de ejecutar secuencias de comandos complejas de Python en términos de rendimiento informático. En caso de que ya hayas instalado Anaconda, no necesitas instalar explícitamente Spyder IDE, en realidad viene por defecto con la distribución de Anaconda.
- **PyCharm:** PyCharm es el IDE más famoso en el mundo profesional ya sea para la ciencia de datos o para la programación convencional de Python.
- **Jupyter Notebook:** Al igual que Spyder, la distribución de Anaconda incluye Jupyter Notebook. Tiene integración de Big Data donde uno puede aprovechar las herramientas de Big Data, como Apache Spark, de Scala, Python y R. Se puede explorar los mismos datos con bibliotecas como Pandas, scikit-learn, matplotlib, entre otros.
- **Rodeo:** Es un IDE de Python que está diseñado expresamente para AA y análisis de datos en Python, fue desarrollado por Yhat y utiliza el núcleo de IPython.
- **Geany:** Es principalmente un IDE de AA para Python, es una de las mejores soluciones para IDE de peso ligero, teniendo una configuración de tamaño muy pequeño. Está escrito en C y C++y a pesar de ser un IDE pequeño, es tan capaz

como cualquier otro IDE. Admite resaltado de la sintaxis y la numeración de líneas. Viene equipado con funciones como la finalización del código, el cierre automático de llaves. También viene con código plegable y admite navegación por código.

- **Atom:** Es un IDE de código abierto desarrollado por Github, está disponible para muchos lenguajes de programación como PHP, Java, entre otros. Atom tiene características interesantes que crean una buena experiencia para los desarrolladores de Python.
- **Google Colab:** Google Colaboratory o Google Colab es un producto de Google Research y probablemente el servicio en la nube más popular para el aprendizaje automático. Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador [CITATION Goo22 \l 1033].

Los IDE's propuestos a utilizar son el PyCharm 2016.1, especializado para aplicaciones de Python y el Google Colab, servicio alojado de Jupyter Notebook que no requiere configuración.

Figura 4. IDE PyCharm Professional 2016.1.

IDE PyCharm 2016.1 como de costumbre, PyCharm 2016.1 está disponible como una edición profesional con todas las funciones para Python y desarrollo web, o como una edición comunitaria gratuita y de código abierto para Python puro y desarrollo científico [CITATION The22 \l 1033].

- Mejoras relacionadas con Python:
  - ✓ Sugerencias de tipo Python 2 y Python 3 e inspección de compatibilidad.
  - ✓ Configuración del contenedor Docker Compose y Docker.

- ✓ Apoyo toxicológico.
- ✓ Soporte mejorado para formularios Django.
- ✓ Mejoras significativas del depurador.
- ✓ Compatibilidad mejorada con IPython Notebook.

- Mejoras de la plataforma:
  - ✓ Git Rebase y Renombrar.
  - ✓ Soporte de Git Worktree.
  - ✓ Resaltado de cambios de palabra en Diff Viewer.
  - ✓ Herramientas de base de datos mejoradas.
  - ✓ Mejoras en la terminal local.
- Mejoras en el desarrollo web:
  - ✓ Mejoras en la compatibilidad con ECMAScript 6 y TypeScript.
  - ✓ Mejoras importantes en el soporte de Angular 2.
  - ✓ Compatibilidad con la depuración de código asíncrono del lado del cliente.
  - ✓ Ejecución y depuración de aplicaciones Node.js en hosts remotos.

Figura 5. IDE Google Colab.

IDE Google Colab es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación. Desde un punto de vista más técnico, Colab es un servicio alojado de Jupyter Notebook que no requiere configuración. Como parte del área de Google Cloud, ofrece acceso gratuito a la informática de GPU y TPU y permite un excelente ecosistema para trabajar y desarrollar algoritmos. Una buena alternativa al trabajo a nivel local. Google

Colab es también uno de los servicios en la nube más famosos para científicos de datos experimentados, investigadores e ingenieros de software [CITATION Goo22 \l 1033].

Ventajas concretas de Google Colab [CITATION Goo22 \l 1033]:

- No necesita hacer una configuración de entorno. Viene con paquetes importantes preinstalados y listos para usar.
- Proporciona acceso directo en el navegador a Jupyter Notebook.
- GPU gratis.
- Permite almacenar cuadernos en Google Drive.
- Permite importar cuadernos desde Github.
- Entrega un código de documento con Markdown.
- Da la opción de cargar datos desde la unidad.

Se escogieron estas herramientas ya que corren en un mismo ecosistema y presentan similares sistemas de tolerancia a fallos y tareas distribuidas. Además, los componentes planteados permiten escalamiento horizontal cuando se tenga una gran cantidad de transacciones a ser procesadas.

Conclusiones del capítulo.

En éste capítulo se realizó una revisión de los modelos de AA para la detección de fraude en transacciones bancarias, donde se encontraron variantes de modelos para este propósito, el problema del desbalance en la información provocada por la muestra de los datos o las propiedades de dominio, donde el más común es la desproporción entre las transacciones fraudulentas con 0.1% de representación y las comunes con un 99.9%.
Se definió la metodología de la MD como KDD que permitió una lógica en la obtención de resultados. Así como se definió el lenguaje de programación Python v3.7.8 para el desarrollo y obtención de los resultados junto a las librerías y los IDE's a utilizar.

En el próximo capítulo se abarcará la metodología KDD secuencialmente atendiendo a los pasos definidos anteriormente.

Capítulo 2: Preparación para el proceso de Minería de Datos.

El presente capítulo aborda un estudio exhaustivo del procedimiento que se llevará a cabo para realizar las pruebas. Con este fin el mismo se desglosa en secciones según los primeros 7 pasos de la secuencia de KDD. Además, sirve como guía para el lector, permitiéndole reproducir el experimento llevado a cabo en este proyecto.

2.1 Entendimiento, conocimientos previos e identificación de la meta.

En este paso se desarrolla un entendimiento de la aplicación de dominio, los conocimientos previos y la identificación de la meta del proceso de KDD desde el punto de vista del cliente [CITATION ELG05 \l 1033].
Para lograr un correcto desarrollo y comprensión del dominio de la aplicación, aprender los conocimientos previos relevantes e identificar los objetivos del usuario final partimos del conjunto de informaciones que dan entrada a este paso de la secuencia KDD.

Entrada:

» Problema a resolver: ¿Cómo automatizar la detección de anomalías en escenarios de flujo de datos bancarios para la identificación de fraudes?

» Objetivo general: Desarrollar un método de detección automática de anomalías en transacciones bancarias para la identificación de fraudes.

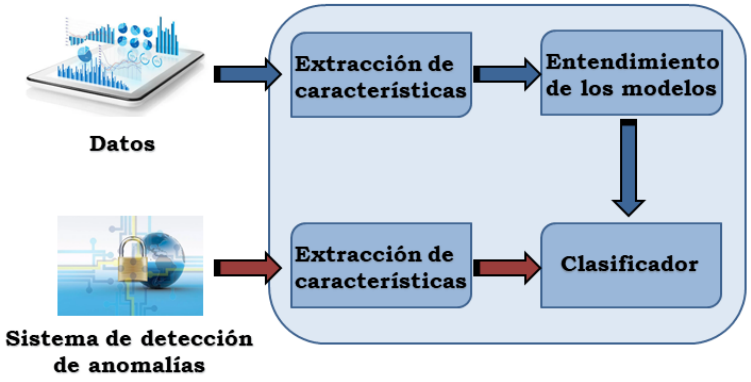


Figura 6. Modelo de la comprensión del negocio o problema. Elaboración propia.

Objetivo del negocio: Detectar el fraude en transacciones bancarias, criterio de éxito cuantitativo: “Número de detecciones de fraude”.

Situación actual: La detección del fraude es realizada de manera manual mediante el estudio de los patrones de comportamiento de cada usuario en un determinado período de tiempo.

Con la aplicación de la minería de datos a este problema se descubre información que no se esperaba obtener, las combinaciones de distintas técnicas otorgan efectos inesperados que se transforman en un valor añadido a la empresa. Enormes bases de datos pueden ser analizadas mediante la tecnología de minería de datos. Los resultados son fáciles de entender: personas sin un conocimiento previo en ingeniería informática pueden interpretar los resultados con sus propias ideas. Contribuye a la toma de decisiones tácticas y estratégicas para detectar la información clave. Los modelos son probados y comprobados usando técnicas estadísticas antes de ser usados, para que las predicciones que se obtienen sean confiables y válidas. En su mayoría, los modelos se generan y construyen de manera rápida. Abre nuevas oportunidades de negocios y ahorra costes a la empresa [CITATION ITE221 \l 1033].

Objetivo de la minería de datos: Determinar un método de clasificación binaria mediante el uso de patrones de comportamiento de los clientes respecto a su capacidad de cometer fraude bancario.

Plan del proyecto:

La última tarea de esta fase tiene como objetivo desarrollar el plan de proyecto considerando los pasos que se deben seguir y los métodos por emplear en cada paso. A continuación se describe el plan del proyecto mediante la construcción de un pipeline representado en la Figura. 7.

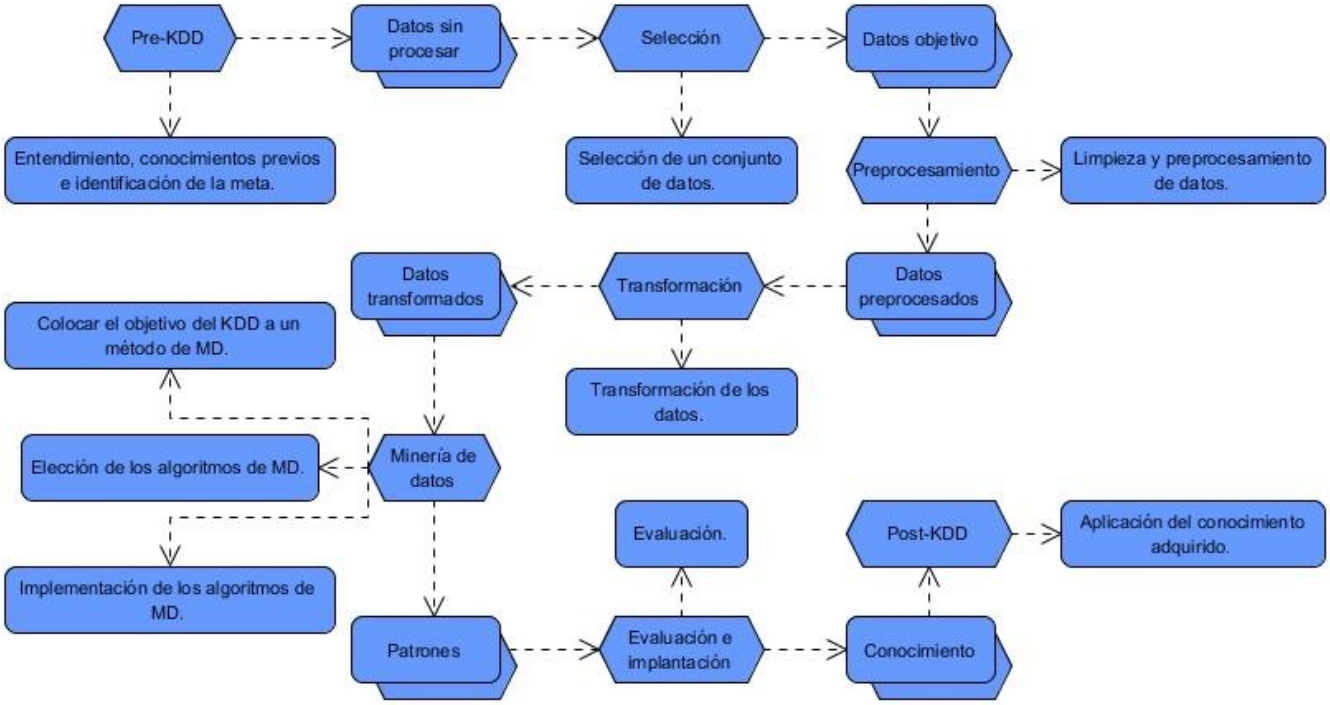


Figura 7. Metodología que guía la realización del proyecto. Elaboración propia.

Salida: Comprensión del problema / dominio / objetivo.

2.2 Selección de los datos.

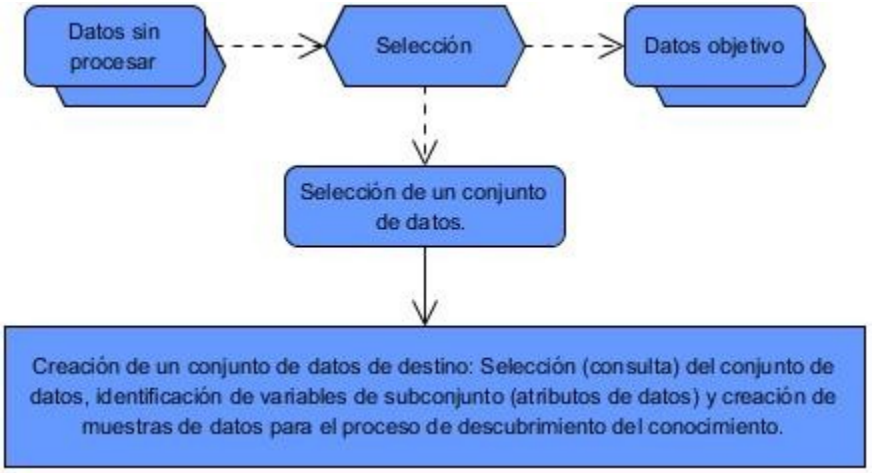


Figura 8. Fase de Selección. Elaboración propia.

Conjunto de datos.

Creación de un conjunto de datos de destino: Selección (consulta) del conjunto de datos, identificación de variables de subconjunto (atributos de datos) y creación de muestras de datos para el proceso de descubrimiento del conocimiento. El conjunto de datos seleccionado

contiene transacciones realizadas con tarjetas de crédito en septiembre de 2013 por titulares de tarjetas europeas. Presenta las transacciones que ocurrieron en dos días.

Salida: Datos de destino / conjunto de datos.

2.3 Limpieza y preprocesamiento de los datos.

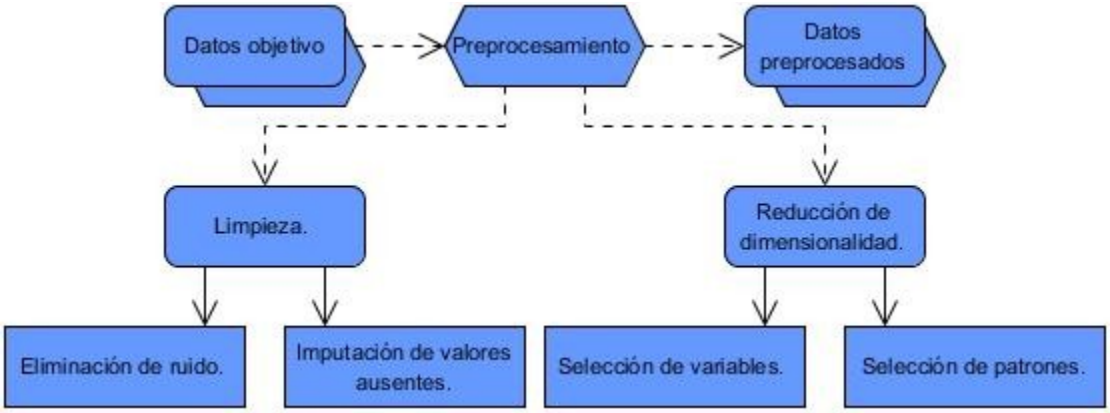


Figura 9. Fase de Preprocesamiento. Elaboración propia.

En la etapa de preprocesamiento/limpieza (data cleaning) se analiza la calidad de los datos, se aplican operaciones básicas como la remoción de datos ruidosos, se seleccionan estrategias para el manejo de datos desconocidos (missing y empty), datos nulos, datos duplicados y técnicas estadísticas para su reemplazo. En esta etapa, es de suma importancia la interacción con el usuario o analista. En el proceso de limpieza todos estos valores se ignoran, se reemplazan por un valor por omisión, o por el valor más cercano, es decir, se usan métricas de tipo estadístico como media, moda, mínimo y máximo para reemplazarlos [CITATION Tim16 \l 3082].

Primeramente se verifica la homogeneidad y completitud de los datos para determinar si son correctos.

Para visualizar el código de este paso de la secuencia KDD consulte el siguiente enlace:

Limpieza / Preprocesamiento y Transformación de los datos

Salida: Datos preprocesados.

```
# visualizar si existen datos duplicados
df [df.duplicated() == True]
# eliminar las filas duplicadas
df1 = df.drop_duplicates()
# re-check: visualizar si existen datos duplicados
df1 [df1.duplicated() == True]
# visualizar si existen valores nulos
nulls = df.isna().sum() # contar valores nulos en cada columna
df_nulls = pd.DataFrame(nulls) # convertir el resultado en un dataframe
df_nulls.transpose() # transponer el marco de datos e imprimir el resultado
```

Figura 10. Preprocesamiento. Fuente: Google Colab.

2.4 Transformación de los datos.

Reducción y proyección de datos: Encontrar características útiles que representen los datos (según el objetivo), incluidas las reducciones y transformaciones de dimensiones.

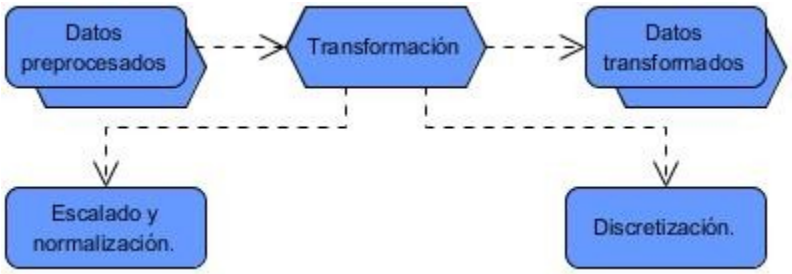


Figura 11. Fase de Transformación. Elaboración propia.

Los datos poseen 31 dimensiones o atributos, la dimensión *"Class"* va a ser separada del conjunto de datos, usada para clasificar los datos preprocesados y compararlos con las predicciones mientras que se trabajará con las 30 dimensiones restantes.

Se realizó la división de los datos en los conjuntos de entrenamiento y prueba, en un 80% y 20% respectivamente. Al entrenar los modelos no es necesario realizar la estandarización y escalado de variables numéricas puesto que al conjunto de datos se le realizó previamente un PCA().

Para visualizar el código de este paso de la secuencia KDD consulte el siguiente enlace:

Limpieza / Preprocesamiento y Transformación de los datos

Para el tratamiento del desbalance de clases se propone hacer uso de SMOTE - Synthetic Minority Over-sampling Technique.

```
# Dividimos los datos en entrenamiento y prueba
df_training = df1.head(int(len(df1) * 0.8))
y_train = df_training['Class']
X_train = df_training.drop('Class', axis=1)
df_test = df.drop(df_training.index)
y_test = df_test['Class']
X_test = df_test.drop('Class', axis=1)
print("Ejemplos usados para entrenar: ", len(X_train))
print("Ejemplos usados para test: ", len(X_test))
# Mostramos los datos de la columna 'Class' para el conjunto de prueba
y_test.value_counts()
```

Figura 12. Transformación. Fuente: Google Colab.

SMOTE - Synthetic Minority Over-sampling Technique [CITATION Ana22 \l 3082].

```
#SMOTE: Synthetic Minority Oversampling Technique
counter=Counter(y_train)
print('Antes', counter)
# oversampling the train dataset using SMOTE
smt=SMOTE(random_state=0)
# X_train, y_train=smt.fit_resample(X_train, y_train)
X_train_sm, y_train_sm=smt.fit_resample(X_train, y_train)

counter=Counter(y_train_sm)
print('Después', counter)
```

Figura 13. SMOTE. Fuente: Google Colab.

En el mundo real, a menudo terminamos tratando de entrenar un modelo en un conjunto de datos con muy pocos ejemplos de una clase determinada (por ejemplo, diagnóstico de enfermedades raras, defectos de fabricación, transacciones fraudulentas), lo que da como resultado un rendimiento deficiente. Debido a la naturaleza de los datos (las ocurrencias son muy raras), no siempre es realista salir y adquirir más.

Una forma de resolver este problema es submuestrear la clase mayoritaria. Es decir, excluiríamos las filas correspondientes a la clase mayoritaria de modo que haya aproximadamente la misma cantidad de filas para las clases mayoritaria y minoritaria. Sin embargo, al hacerlo, perdemos una gran cantidad de datos que podrían usarse para entrenar nuestro modelo, mejorando así su precisión (por ejemplo, un mayor sesgo).

Otra opción es sobremuestrear la clase minoritaria. En otras palabras, duplicamos aleatoriamente las observaciones de la clase minoritaria. El problema con este enfoque es que condu-

ce a un sobreajuste porque el modelo aprende de los mismos ejemplos. Aquí es donde entra SMOTE.

A un alto nivel, el algoritmo SMOTE se puede describir de la siguiente manera:

- ✓ Toma la diferencia entre una muestra y su vecino más cercano.
- ✓ Multiplica la diferencia por un número aleatorio entre 0 y 1.
- ✓ Agrega esta diferencia a la muestra para generar un nuevo ejemplo sintético en el espacio de características.

Continúa con el próximo vecino más cercano hasta el número definido por el usuario.

Salida: datos transformados.

2.5 Modelo de aprendizaje.

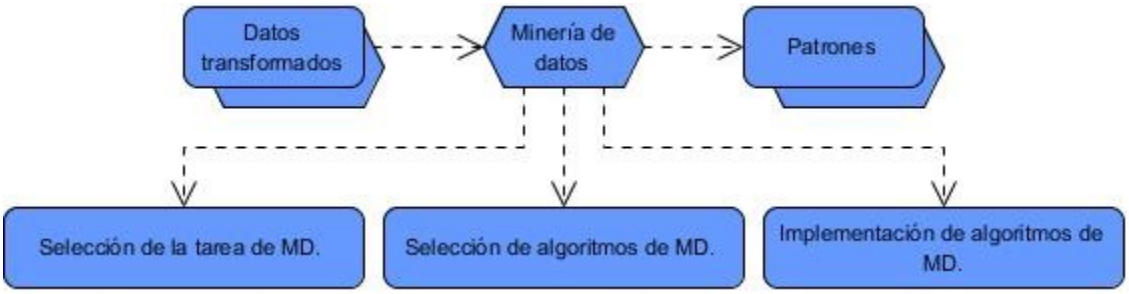


Figura 14. Fase de aprendizaje. Elaboración propia.

Selección de la tarea de minería de datos: Decisión sobre qué métodos aplicar para la clasificación, el agrupamiento, la regresión u otra tarea.

En función de la tarea que realizan, los algoritmos de AA se pueden dividir en algoritmos de clasificación, regresión, agrupación y asociación [CITATION Cri21 \l 3082]:

- Algoritmos de clasificación: se utilizan cuando la etiqueta toma valores discretos dentro de un conjunto finito de resultados. La clasificación puede ser binaria o múltiple.
- Algoritmos de regresión: su objetivo es establecer una relación entre un cierto número de características y una variable objetivo continua.
- Algoritmos de agrupación (clustering): es un procedimiento de agrupación de una serie de datos de acuerdo con un criterio, por lo general distancia o similitud entre casos.
- Algoritmos de asociación: los algoritmos de reglas de asociación tienen como objetivo encontrar relaciones dentro un conjunto de transacciones, en concreto, ítems o atributos que tienden a ocurrir de forma conjunta.

Debido al enfoque del proyecto centrado en la detección de fraude en transacciones bancarias y al hecho de que la predicción es a veces referida como una minería de datos supervisada, el método propuesto a usar en el

presente proyecto es la clasificación. Usada en varias tareas dentro de las que se encuentra el fraude financiero. Específicamente la clasificación binaria ya que el conjunto de datos se encuentra previamente etiquetado en 0 (transacción normal) y 1 (transacción fraudulenta).

Resultado: Método de clasificación binaria.

Selección de algoritmos de minería de datos: Selección del método para la búsqueda de patrones, el cual decida sobre los modelos apropiados y sus parámetros, y haga coincidir los métodos con el objetivo del proceso.

Se realizó una validación cruzada de búsqueda de cuadrícula (GridSearchCV) con el fin de determinar los mejores parámetros para cada modelo. Para visualizar el código de este paso de la secuencia KDD consulte el siguiente enlace:

Ajuste de parámetros con GridSearchCV para los algoritmos de AA.

```
# Iniciamos el modelo
dtree_model = DecisionTreeClassifier(random_state=0)
# operaciones en orden
operations =  [('dtree_model', dtree_model)] # Observe que están escritos en tuplas dentro de una lista
# configurar el pipeline
pipe = Pipeline(operations)
# Estos son los parámetros que se pueden modificar en el clasificador DT
dtree_model.get_params().keys()
# entrenamiento del pipeline
pipe.fit(X_train_sm, y_train_sm)
pipe_pred = pipe.predict(X_test)
tn, fp, fn, tp = confusion_matrix(y_test, pipe_pred).ravel()
print('Leyenda »', 'tn: ', tn, 'fp: ', fp, 'fn: ', fn, 'tp: ', tp)
plt.figure(figsize=(5,2))
plt.title("Matriz de Confusión del Pipeline:", fontsize=16)
sns.heatmap(confusion_matrix(y_test, pipe_pred), annot=True, cmap='Greys', fmt='.0f')
plt.show();
# establecer el parámetro del grid
param_grid = {'criterion':['gini','entropy'],'max_depth':range(1,150,1)}
scoring = {
    'sensitivity': make_scorer(recall_score),
    'specificity': make_scorer(recall_score,pos_label=0)
}
# Poniendo todo junto
full_cv_classifier = GridSearchCV(dtree_model, param_grid, cv=5, scoring=scoring, refit='sensitivity')
# Entrenamos el Pipeline
full_cv_classifier.fit(X_train_sm, y_train_sm)
# Mejores parámetros del modelo
full_cv_classifier.best_estimator_.get_params()
full_cv_classifier.best_estimator_
```

Figura 15. gridSearchCV. Fuente: Google Colab.

La precisión de la matriz de confusión no es significativa para una clasificación desequilibrada, por lo que elección final del método de AA se determinará mediante el uso del F1 Score tras tratar el problema del desbalance.

Resultado: Algoritmos seleccionados.

Implementación de los algoritmos de MD: Búsqueda de patrones de interés en forma específica, como reglas de clasificación, árboles de decisión, modelos de regresión, tendencias, grupos y asociaciones.

La implementación de los algoritmos de MD mediante un pseudocódigo nos permite realizar un análisis y formulación del problema, así como obtener los resultados esperados del mismo para el diseño de un algoritmo con los datos disponibles. Permite además, una traducción del algoritmo mediante el uso de restricciones y llevar a cabo los procesos necesarios para depurar el programa. Para los oficios de este documento de diploma se agruparan una serie de pseudocódigos de algoritmos de AA por familias según las referencias del estudio del estado del arte.

Algoritmos basados en redes neuronales:

Algoritmo 1. Pseudocódigo para ANN.

```
1:  procedimiento MLPClassifier(Entrada, Neurona, Repetir)
        Crear base de datos de entrada.
2:      Entrada ← Base de datos con todas las combinaciones posibles.
        Entrenar ANNs
3:      para Entrada = 1 hasta fin de Entrada hacer
4:          para Neurona = 1 hasta "#_límite" hacer
5:              para Repetir = 1 hasta "#_límite" hacer
6:                  entrenar: ANN
7:                  Almacenamiento-ANN ← guardar la prueba más alta R^2
8:              fin para
9:          fin para
10:         Almacenamiento-ANN ← guardar mejor predicción de ANN dependiendo de las entradas
11:     fin para
12:     devolver Almacenamiento-ANN
13: fin procedimiento
```

Tabla 2. Pseudocódigo para ANN.

Algoritmos basados en árboles de decisión:

Algoritmo 2. Pseudocódigo para DT.

```
1:  procedimiento DecisionTreeClassifier(árbol, punto_prueba)
2:      si árbol es de la forma Hoja(adivinar) entonces
3:          devolver adivinar
4:      sino si árbol es de la forma Nodo( f , izq, der) entonces
5:          si f = no en punto_prueba entonces
6:              devolver DecisionTreeClassifier(izq, punto_prueba)
7:          Sino
8:              devolver DecisionTreeClassifier(der, punto_prueba)
9:          fin si
10:     fin si
11: fin procedimiento
```

Tabla 3. Pseudocódigo para DT.

Algoritmos combinados (basados en árboles de decisión):

Algoritmo 3. Pseudocódigo para GBC.

```
1:  procedimiento GradientBoostingClassifier( )
2:      Entrada: D = {(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)}, L(y, O(x))
3:      inicio
```

4: Inicializar: $O_0(x)=argmin_w \sum_{i=1}^{n} L(y_i, w)$

5: **para** $m=1:M$ **hacer**

6: $\gamma_{\Im}=\frac{-\partial L(y_i, O(x_i))}{\partial O(x_i)}$

7: **entrenar:** alumno débil $C_m(x)$ en los datos de entrenamiento.

8: **calcular:** $\begin{matrix} x \\ (¿¿i) \\ y_i, O_{m-1}(x_i)+w C_m ¿ \\ L¿ \end{matrix}$

$w: w_m=argmin_w \sum_{i=1}^{n} ¿$

9: **actualizar:** $O_m(x)=O_{m-1}(x)+w_m C_m(x)$

10: **fin para**

11: **fin procedimiento**

12: **Salida:** $O_m(x)$

Tabla 4. Pseudocódigo para GBC.

Algoritmo 4. Pseudocódigo para IF.

1: **procedimiento** IsolationForest(*X, t,* ψ)

2: **Entrada:** X – datos de entrada, t – número de árboles, ψ – tamaño de submuestreo

3: **inicializar:** *Bosque*

4: **establecer:** límite de altura $\begin{matrix} \log \\ (¿¿2\psi) \\ ceiling ¿ \end{matrix}$

5: **para** $i=1$ hasta t **hacer**

6: $X' \leftarrow sample(X, \psi)$

7: $Bosque \leftarrow Bosque\, iTree(X', 0, l)$

8: **fin para**

9: **devolver** *Bosque*

10: **fin procedimiento**

11: **Salida:** un conjunto de t *iTrees*

Tabla 5. Pseudocódigo para IF.

Algoritmo 5. Pseudocódigo para RF.

1: **procedimiento** StreamingRFC( )

2: Para generar **c** clasificadores

3: **para** $i=1$ hasta c **hacer**

4: Muestrear aleatoriamente los datos de entrenamiento D con reemplazo para producir D_i

5: **crear:** un nodo raíz, N_i que contenga D_i

6: BuildTree(N_i)

7: **fin para**

8: **BuildTree(** N **):**
9: **si** N contiene instancias de una sola clase **entonces**
10: **devolver**
11: **sino**
12: **seleccionar:** aleatoriamente x de las posibles funciones de división en N
13: **seleccionar:** las F características con la mayor ganancia de información para dividir
14: **crear:** f nodos secundarios de $N, N_1, \ldots, N_f$, donde F tiene f valores posibles $(F_1, \ldots, F_f)$
15: **para** $i=1$ hasta f **hacer**
16: **establecer:** contenido de N_i en D_i , donde D_i son todas las instancias en N que coinciden
17: F_i
18: BuildTree(N_i)
19: **fin para**
20: **fin si**
21: **fin procedimiento**

Tabla 6. Pseudocódigo para RF.

Algoritmos de aprendizaje basados en instancias:

Algoritmo 6. Pseudocódigo para KNN.

1: **procedimiento** KNeighborsClassifier( )
Construir el conjunto de datos normales de entrenamiento D;
2: **para** cada proceso X en los datos de prueba **hacer**
3: **si** X tiene una llamada de sistema desconocida **entonces**
4: X es anormal;
5: **sino entonces**
6: **para** cada proceso D_j en los datos de entrenamiento **hacer**
7: **calcular:** $¿(X, D_j)$;
8: **si** $¿(X, D_j)$ igual 1.0 **entonces**
9: X es normal;
10: **fin si**
11: **fin para**
12: **buscar:** k mayor puntaje de $¿(X, D)$;
13: **calcular:** $¿_{avg}$ para k-vecinos más cercanos;
14: **si** $¿_{avg} > threshold$ **entonces**
15: X es normal;
16: **sino entonces**
17: X es anormal;
18: **fin para**
19: **fin procedimiento**

Tabla 7. Pseudocódigo para KNN.

Algoritmo 7. Pseudocódigo para SVM.

1: **procedimiento** SVC( )
2: candidatoSV = { par más cercano para clases opuestas }

3: **mientras** haya puntos violatorios **hacer**
4: **encontrar:** violador
5: candidatoSV = candidatoSV
6: S
7: violador
8: **si** cualquier $\alpha_p<0$ debido a la adición de c a S **entonces**
9: candidatoSV = candidatoSV \ p
10: **repetir:** hasta que se eliminen todos esos puntos
11: **fin si**
12: **fin mientras**
13: **fin procedimiento**

Tabla 8. Pseudocódigo para SVM.

Algoritmo de regresión:

Algoritmo 8. Pseudocódigo para LR.

1: **procedimiento** LogisticRegression(***D***)
2: **Entrada:** Datos de entrenamiento D
3: **inicio**
4: **para** $i=1$ hasta k **hacer**
5: **para** cada instancia de datos de entrenamiento D_j **hacer**
6: **establecer:** valor objetivo para la regresión en $Z_i=\frac{y_i-P(1\vee d_j)}{[P(1\vee d_j)(1-P(1\vee d_j))]}$
7: **inicializar:** peso de la instancia d_j en $[P(1\vee d_j)(1-P(1\vee d_j))]$
8: **finalizar:** $f(j)$ a los datos con valor de clase $(\overset{Z}{¿¿}j)$ y peso $(\overset{w}{¿¿}j)$
9: **fin para**
10: **fin para**
11: Decisión de etiqueta clásica
12: **si** $P_{id}>0.5$ **entonces**
13: Assign(class label: 1)
14: **sino entonces**
15: Assign(class label: 2)
16: **fin si**
17: **fin**
18: **fin procedimiento**

Tabla 9. Pseudocódigo para LR.

Algoritmo probabilístico:

Algoritmo 9. Pseudocódigo para NBC.

1: **procedimiento** GaussianNB(***S***)
2: Dado un conjunto de datos $S(x,c)$;
3: **mientras** haya N ejemplos de entrenamiento disponibles **hacer**
4: **para** $i=0$ hasta L **hacer**
5: **estimar:** $P(C=ci)$ con ejemplo en S ;
6: **para** $j=0$ hasta n **hacer**
7: **para** $k=0$ hasta N **hacer**

8: **estimar**: $P(Xj=xjk|C=ci¿$;
9: **fin para**
10: **fin para**
11: **fin para**
12: **fin mientras**
13: **mientras** haya ejemplos de prueba disponibles **hacer**
14: **obtener:** mayores probabilidades condicionales calculadas para cada clase
15: **fin mientras**
16: **fin procedimiento**

Tabla 10. Pseudocódigo para NBC.

Resultado: Patrones.

Conclusiones del capítulo

Se definieron las metas y procedimientos a utilizar para poder guiar la experimentación según la secuencia de la metodología KDD. Para la obtención de los resultados se caracteriza y selecciona el conjunto de datos sujeto. Se realiza la limpieza y prepocesamiento del conjunto de datos, para definir si la detección de fraude realiza la transformación de los datos, y así comparar las salidas de los modelos con las predicciones.

Se determina la clasificación como método de MD por el enfoque que posee. En la obtención de los resultados se eligen los parámetros para ajustar los modelos mediante la técnica de *"Grid Search CV".*

Concluido el trabajo realizado en este capítulo es posible proceder a la evaluación de los algoritmos. La evaluación y aplicación del conocimiento conforman los pasos finales de la secuencia KDD aplicada para el desarrollo de este proyecto. Para ello se realizarán experimentos donde sus resultados serán detallados y representados en el capítulo siguiente.

Capítulo 3: Evaluación y aplicación de los modelos de AA.

El presente capítulo abarca la evaluación de los modelos de AA seleccionados en los pasos anteriores de la secuencia KDD y se definen los parámetros y métricas.

3.1 Evaluación.

Para la realización de este proyecto se tomó un fragmento del conjunto de datos original, donde se tienen 64 fraudes de 20 000 transacciones. Existe un desequilibrio en cuanto a la clase positiva (fraudes) que representa el 0,1 % de todas las transacciones.

Solo contiene variables de entrada numéricas que son el resultado de una transformación de PCA. Desafortunadamente, debido a problemas de confidencialidad, no se pueden proporcionar las características originales y más información de antecedentes sobre los datos. Las características V1, V2, ... V28 son los principales componentes obtenidos con PCA, las únicas características que no se han transformado con PCA son *"Time"* y *"Amount"*. La dimensión *"Time"* contiene los segundos transcurridos entre cada transacción y la primera transacción en el conjunto de datos. La dimensión *"Amount"* es la cantidad de la transacción, esta función se puede utilizar para el aprendizaje sensible a los costos, por ejemplo. La dimensión *"Class"* es la variable de respuesta y toma el valor 1 en caso de fraude y 0 en caso contrario.

A continuación se definen las tablas obtenidas de los resultados de las comparaciones según las métricas seleccionadas, permitiendo realizar la definición de superioridad de los algoritmos. Para visualizar el código de este paso consulte el siguiente enlace:

Experimentos de evaluación

```
model = DecisionTreeClassifier(random_state=0)
operations = [('model', model)]
pipe = Pipeline(operations)
pipe.fit(X_train, y_train)
pipe_pred = pipe.predict(X_test)
import warnings
warnings.filterwarnings('ignore')
auc=roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
tn, fp, fn, tp = confusion_matrix(y_test, pipe_pred).ravel()
print('tn : ', tn)
print('fp : ', fp)
print('fn : ', fn)
print('tp : ', tp)
a=accuracy_score(y_test, model.predict(X_test))
print("(accuracy_score) =  {}".format(a))
p=precision_score(y_test, model.predict(X_test))
print("(precision_score) =  {}".format(p))
r=recall_score(y_test, model.predict(X_test))
print("(recall_score) =  {}".format(r))
f1=f1_score(y_test, model.predict(X_test))
print("(f1_score) =  {}".format(f1))
print("(auc_score) =  {}".format(auc))
specificity = tn / (tn+fp)
print('specificity : ', specificity)
sensitivity = tp / (tp+fn)
print('sensitivity : ', sensitivity)
G_Mean= np.sqrt(sensitivity * specificity)
print('G-Mean  : ', G_Mean)
```

Figura 16. Ejemplo de los experimentos. Fuente: Google Colab.

3.1.1 Experimento 1: Comparación entre los resultados del conjunto de datos original.

En este experimento se obtienen los resultados de cada métrica para realizar una comparación de superioridad entre los algoritmos propuestos con el conjunto de datos en su estado original, determinando los 5 lugares del promedio de los mejores algoritmos donde se representará para el mejor valor como , el segundo mejor valor como y el tercer mejor valor como .

	Métricas		
Modelo	**F1S**	**Sen**	**G-M**
ANN	0.0	0.0	0.0
DT	0.4737	0.5	0.7061
GBC	0.5625	0.5	0.7067
IF	0.0036	0.3889	0.1285
KNN	0.0	0.0	0.0
LR	0.7586	0.6111	0.7817
NBC	0.2143	0.6667	0.8080
RF	0.4186	0.5	0.7057
SVM	0.0	0.0	0.0

Tabla 11. Comparación del conjunto de datos original.

Las comparaciones anteriores y los resultados expuestos permiten la obtención de las siguientes observaciones en general.

- ✓ Modelo de mayor F1 Score: LR.
- ✓ Modelo de mayor Sensitivity: NBC.
- ✓ Modelo de mayor G-Mean: NBC.
- ✓ Mejores modelos: 1.LR, 2.GBC, 3.NBC, 4.DT y 5.RF.

Teniendo en cuenta estas observaciones, se decide utilizar el modelo LR con los datos originales.

3.1.2 Experimento 2: Comparación entre los modelos de AA teniendo en cuenta los resultados de los pasos 3 y 4 de la secuencia KDD.

En este experimento se obtienen los resultados de cada métrica para realizar una comparación de superioridad entre los algoritmos propuestos con el conjunto de datos preprocesado sin tratar el desbalance, determinando los 5 lugares del promedio de los mejores algoritmos

donde se representará para el mejor valor como , el segundo mejor valor como y el tercer mejor valor como .

Modelo	Métricas		
	F1S	Sen	G-M
ANN	0.0	0.0	0.0
DT	0.4737	0.5	0. 7061
GBC	0.4091	0.5	0. 7056
IF	0. 0036	0.3889	0. 1187
KNN	0.0	0.0	0.0
LR	0.6667	0.5	0. 7071
NBC	0.1890	0.6667	0. 8067
RF	0.4	0.5	0. 7055
SVM	0.0	0.0	0.0

Tabla 12. Comparación de los resultados de los pasos 3 y 4 de KDD.

Las comparaciones anteriores y los resultados expuestos permiten la obtención de las siguientes observaciones en general.

- ✓ Modelo de mayor F1 Score: LR.
- ✓ Modelo de mayor Sensitivity: NBC.
- ✓ Modelo de mayor G-Mean: NBC.
- ✓ Mejores modelos: 1.LR, 2.DT, 3.NBC, 4.GBC y 5.RF.

Teniendo en cuenta estas observaciones, se decide utilizar el modelo LR con los datos limpios, preprocesados y transformados.

3.1.3 Experimento 3: Comparación entre los modelos de AA tras aplicar SMOTE.

En este experimento se obtienen los resultados de cada métrica para realizar una comparación de superioridad entre los algoritmos propuestos con el conjunto de datos preprocesado y balanceado, determinando los 5 lugares del promedio de los mejores algoritmos donde se representará para el mejor valor como , el segundo mejor valor como y el tercer mejor valor como .

Modelo	Métricas		
	F1S	Sen	G-M
ANN	0.5806	0.5	0.7068

DT	0.4211	0.4444	0.6657
GBC	0.2018	0.6111	0.7040
IF	0.0046	0.5	0.1373
KNN	0.0	0.0	0.0
LR	0.0552	0.7222	0.8024
NBC	0.1128	0.6667	0.7979
RF	0.1803	0.6111	0.7727
SVM	0.0	0.0	0.0

Tabla 13. Comparación de los resultados tras aplicar SMOTE.

Las comparaciones anteriores y los resultados expuestos permiten la obtención de las siguientes observaciones en general.

- ✓ Modelo de mayor F1 Score: ANN.
- ✓ Modelo de mayor Sensitivity: LR.
- ✓ Modelo de mayor G-Mean: LR.
- ✓ Mejores modelos: 1.LR, 2.NBC, 3.RF, 4.DT y 5.GBC.

Teniendo en cuenta estas observaciones, se decide utilizar el modelo LR tras aplicar SMOTE.

3.1.4 Experimento 4: Comparación entre los modelos de AA tras aplicar Grid Search CV.

En este experimento se obtienen los resultados de cada métrica para realizar una comparación de superioridad entre los algoritmos propuestos con el conjunto de datos preprocesado y balanceado tras ajustar los modelos con los parámetros obtenidos del Grid Search Cross Validation, determinando los 5 lugares del promedio de los mejores algoritmos donde se representará para el mejor valor como ▉, el segundo mejor valor como ▉ y el tercer mejor valor como ▉.

	Métricas		
Modelo	**F1S**	**Sen**	**G-M**
ANN	0.0088	1.0	0.0
DT	0.2650	0.6111	0.7765
GBC	0.2609	0.1667	0.4081
IF	0.0556	0.6111	0.7455
KNN	0.0	0.0	0.0
LR	0.0552	0.7222	0.8024
NBC	0.0085	0.8889	0.2639

RF	0.0142	0.8333	0.6375
SVM	0.0	0.0	0.0

Tabla 14. Comparación de los resultados tras aplicar GridSearchCV.

Las comparaciones anteriores y los resultados expuestos permiten la obtención de las siguientes observaciones en general.

- ✓ Modelo de mayor F1 Score: DT.
- ✓ Modelo de mayor Sensitivity: ANN.
- ✓ Modelo de mayor G-Mean: LR.
- ✓ Mejores modelos: 1.DT, 2.LR, 3.RF, 4.IF y 5.NBC.

Teniendo en cuenta estas observaciones, se decide utilizar el modelo DT.

Partiendo de los resultados de las tablas previamente mostradas: Se puede observar como el algoritmo DT se mantiene estable dentro de los mejores valores entre el promedio de los valores obtenidos por las métricas para conjuntos de datos desbalanceados previamente mencionadas en el estado del arte. Así como también se reafirma como el algoritmo de mejores resultados tras el cumplimiento de todos los pasos de la secuencia de KDD. Se propone hacer uso de DT para dar solución a la problemática existente.

3.1.5 Entorno de la experimentación.

Entorno de experimentación para Apache Kafka:

- Laptop Fujitsu Lifebook E Series, con un microprocesador Intel(R) Core(TM) i5-4210M CPU @2.60GHz 2.60GHz, 8 GB de RAM y un disco duro de 500 gigabyte. El sistema operativo utilizado es Windows 10 Pro en la compilación 20H2.
- Las herramientas utilizadas para la experimentación se definieron en capítulos anterioriores.
- Para la evaluación de los algoritmos propuestos se procede a realizar los experimentos definidos.

Entorno de experimentación para Colab:

- GPU Nvidia T4 Tensor Core.
- Los cuadernos pueden ejecutarse durante 12 horas como máximo, en función de la disponibilidad y de los patrones de uso.
- Una máquina virtual con una capacidad estándar de memoria RAM de 16Gb.

3.2 Aplicación del conocimiento adquirido

Para aplicar el conocimiento adquirido se decide desarrollar un entorno distribuido, el cual, no es más que un algoritmo que permite a los usuarios realizar la detección de anomalías en tiempo real. Éste método reduce grandemente la cantidad de software a desarrollar, lo que conlleva a la disminución de costos y riesgos, además de una entrega más rápida del software. En ella se puede observar los distintos componentes que forman parte de este entorno distribuido, para el entendimiento de cada uno se dividirán en cuatro partes: primero el conjunto de datos, luego el KafkaProducer y el KafkaConsumer, y por último la predicción.

3.2.1 Arquitectura del entorno distribuido.

La primera funcionalidad recibe csv como la dirección del archivo que almacena el conjunto de datos, se realizan particiones de los datos obteniendo a y_test como la proporción deseada para el conjunto de prueba con respecto al conjunto general. La segunda funcionalidad va orientada a la lectura del conjunto de datos y su transmisión de datos mediante una producción de los mismos imprimidos en pantalla. La tercera funcionalidad va orientada a la creación y ajuste del modelo seleccionado. Luego de ser leídos y particionados los datos, esta funcionalidad aplica la estrategia de SMOTE al conjunto de entrenamiento para tratar el problema del desbalance. Luego, recibe como entrada los datos producidos por el KafkaProducer del conjunto de datos y realiza su clasificación. Y la última funcionalidad, simplemente imprime por consola la información de los resultados de la clasificación y a su vez guarda en un directorio los resultados de las métricas y el modelo.

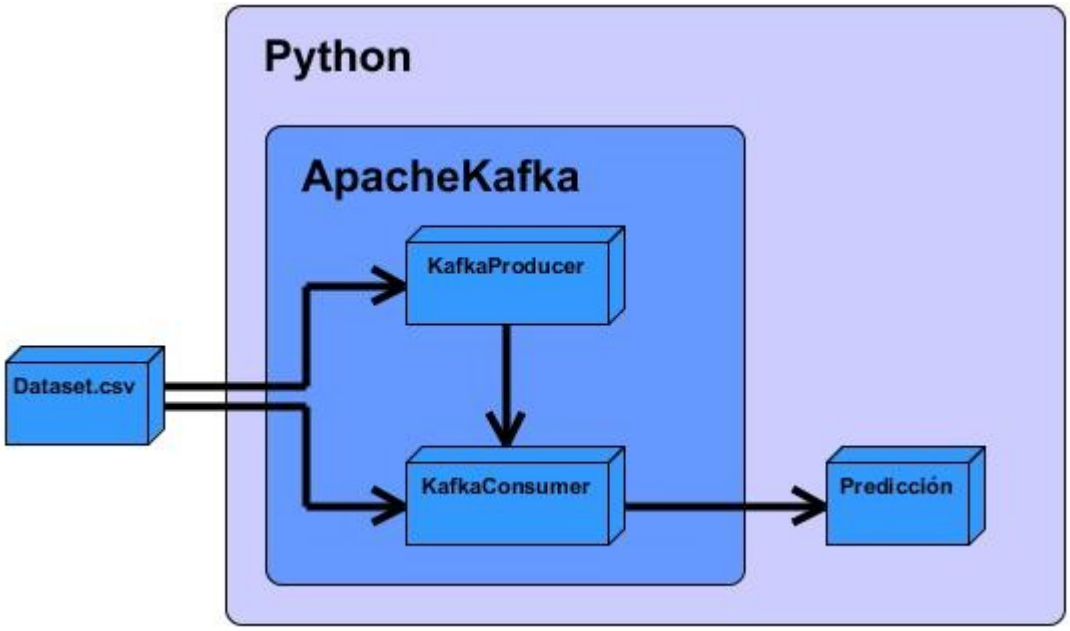


Figura 17. Arquitectura del entorno distribuido. Elaboración propia.

3.2.2 Propuesta de detección de anomalías del método Decision Tree Classifier para la transmisión de datos [CITATION LYu10 \l 3082].

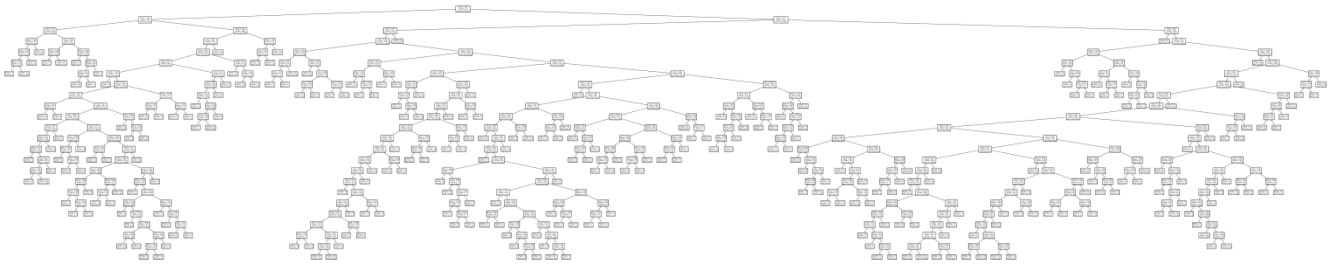


Figura 18. Modelo Decision Tree Classifier.

Construcción del DT. Una forma recursiva de arriba hacia abajo y de dividir - conquistar son las formas que se utilizan para construir un árbol de decisión, el árbol comienza con el nodo raíz para representar todo el conjunto de datos de entrenamiento.

- Si las listas de entrenamiento tienen el mismo resultado, el nodo será hoja y se etiquetará con esa clase.
- De lo contrario, el árbol selecciona el mayor atributo de información para dividir el conjunto y etiqueta al nodo por el nombre del atributo.
- Repita los pasos y deténgase cuando todas las muestras tengan la misma clase o no haya más muestras o nuevos atributos a la porción.

Medidas de selección de atributos. Muchas medidas se pueden utilizar para determinar la dirección óptima para dividir los registros tales como:

Entropía. Es una de las medidas de la teoría de la información; detecta la impureza del conjunto de datos. Si el atributo toma c valores diferentes, entonces la entropía S relacionada con la clasificación c-sabia se define como la ecuación siguiente:

$$E(S)=\sum_{i=1}^{c} -\rho_i \log_2 \rho_i \; Eq\,1$$

ρ_i es la relación de S pertenecientes a la clase i . La entropía es una unidad de la longitud esperada medida en bits, por lo que el algoritmo es base 2.

Ganancia de información. Elige cualquier atributo que se utilice para dividir un determinado nodo. Prioriza nominar atributos que tengan gran número de valores calculando la diferencia de entropía. El valor de la ganancia de información será cero cuando el número de “sí” o “no” es cero y cuando el número de “sí” y “no” es igual, la información alcanza un máximo. La ganancia de información, $Gain(S,A)$ de un atributo A , relativa a la colección de ejemplos S , se define como la siguiente:

$$Gain(S,A)=E(S)-\sum_{v\in Values(A)} \frac{S_v}{S} E(S_v) \; Eq\,2$$

Donde $Values(A)$ es el conjunto de todos los valores potenciales para el atributo A , y S_v es el subconjunto de S para el cual el atributo A tiene valor v . Podemos usar esta medida para agrupar atributos y estructurar el árbol de decisión donde en cada nodo se ubica el atributo con mayor ganancia de información entre los atributos aún no considerados en el camino desde la raíz.

La relación de ganancia. Es una modificación de la ganancia de información que reduce su sesgo en los atributos de rama alta. $SplitInfo(D,T)$ es la información debida a la división de T sobre la base del valor del atributo categórico D como se muestra en las siguientes ecuaciones:

$$GainRatio(D,T)=\frac{Gain(D,T)}{SplitInfo(D,T)} Eq\,3$$

$$SplitInfo(D,T)=-\sum_{i=1}^{k}\frac{D_i}{T}\log_2\frac{D_i}{T} Eq\,4$$

Conclusiones del capítulo

Los resultados de la experimentación permitieron definir la efectividad de los modelos de AA en la detección de fraude bancario. El modelo DT se mantuvo estable dentro de los 5 mejores algoritmos durante la realización de los experimentos. Posee los mejores resultados con los datos balanceados y los modelos ajustados con los mejores parámetros obtenidos del GridSearchCV. Se logró el diseño e implementación de un entorno distribuido para flujo de datos con la arquitectura del modelo utilizado, permitiendo su uso y mejoramiento.

CONCLUSIONES FINALES

Se realizó una caracterización y sistematización del estado del arte referido al problema de la detección de fraudes bancarios y los enfoques basados en detección de anomalías, sustentado en escenarios de Flujo de Datos. Se definió la metodología de la MD, la cual, permite una lógica en la obtención de los resultados. Se determinó usar el lenguaje de programación Python en su versión 3.7.8 para el desarrollo y obtención de los resultados junto a las librerías necesarias y los IDE's a utilizar.

Se desarrolló un método para algoritmos basados en el aprendizaje supervisado y la computación distribuida para la detección de anomalías en operaciones bancarias. Se validó la solución implementada mediante el diseño de experimentos sobre conjuntos de datos de referencia, comparando los resultados con otros algoritmos del estado del arte.

RECOMENDACIONES

Se recomienda para futuras implementaciones: Realizar un método de conjunto equilibrado entre Isolation Forest, Random Forest y Decision Tree, el cual describe un entrenamiento equilibrado de un método de conjunto híbrido para conjuntos de datos desequilibrados, desarrollar el *Online Learning, Incremental Learning, Deep Learning*, así como los métodos de selección de características.

REFERENCIAS BIBLIOGRÁFICAS

Allen Downey, B. L. (2018). *Introducción a la programación en Julia. © 2018 Allen Downey, Ben Lauwens.* Obtenido de https://introajulia.org/

Álvarez, M. (19 de noviembre de 2003). *Lenguaje de programación de propósito general, orientado a objetos, que también puede utilizarse para el desarrollo web.* Obtenido de https://desarrolloweb.com/articulos/1325.php

AprendeIA. (2022). *6 IDEs para Machine Learning con Python. © 2022 AprendeIA.* Obtenido de https://aprendeia.com/ide-para-machine-learning-con-python/

B., M. Q. (2020). *"MODELO DE REFERENCIA PARA LA DETECCIÓN DE FRAUDES EN EL PROCESO DE NÓMINAS BASADO EN TÉCNICAS Y HERRAMIENTAS DE AUDITORÍA ASISTIDA POR COMPUTADORA".* INFOTEC.

Bagnato, J. I. (16 de mayo de 2019). *Aprende Machine Learning*. Obtenido de https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/

BBVA. (2022). *Entidad Financiera.* Obtenido de https://www.bbva.mx/educacion-financiera/e/entidad_financiera.html

Blog, T. P. (22 de marzo de 2016). *Announcing General Availability of PyCharm 2016.1. © 2000-2022 JetBrains s.r.o.* Obtenido de https://blog.jetbrains.com/pycharm/2016/03/announcing-general-availability-of-pycharm-2016-1/

Center, S. E. (6 de abril de 2021). *Computación distribuida: qué es, cómo funciona, ventajas y desventajas.* Obtenido de https://news.sap.com/spain/2021/04/computacion-distribuida-que-es-como-funciona-ventajas-y-desventajas/

Colaboratory., G. C. (2022). *Google Colab. (n.d.). Colaboratory.* Obtenido de https://research.Google.com/Colaboratory/intl/es/faq.html

Concepto, E. (2022). *Dato.* Obtenido de https://concepto.de/dato/

Criach Fernández, P. (2021). *CLASSIFICATION OF MEDICAL IMAGES FROM MARQUÉS DE VALDECILLA UNIVERSITY HOSPITAL PACS WITH A CONVOLUTIONAL NEURAL NETWORK.* . Universidad de Cantabria: Facultad de Ciencias.

Diwan, P. N. (2020). *Relative Analysis of ML Algorithm QDA, LR and SVM for Credit Card Fraud Detection Dataset.* Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 976-981, doi: 10.1109/ISMAC4.

DOCS, M. W. (2022). *Fundamentos de JavaScript. © 2022 MDN WEB DOCS.* Obtenido de https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

Duman, Y. S. (2011). *Detecting credit card fraud by ANN and logistic regression.* International Symposium on Innovations in Intelligent Systems and Applications, 2011, pp. 315-319, doi: 10.1109/INISTA.2011.5946108.

E. L. Guzmán, P. (2005). *Módulo Minería de Datos.* Grupo de Investigación MIDAS, Universidad Nacional de Colombia (2005).

E., M. R. (2020). *Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo.* RISTI, № E28, 04/2020, pp. 586-599.

E., R. (2 de abril de 2021). *Hiberus Blog — Cómo el Machine Learning mejora el sistema bancario.* Obtenido de https://www.hiberus.com/crecemos-contigo/como-el-machine-learning-mejora-el-sistema-bancario/

F., A. (2020). *Machine Learning en la detección de fraudes de comercio electrónico aplicado a los servicios bancarios.* . Ciencia y Tecnología, №20, (2020), pp. 81-95 ISSN 2344-9217.

Formación, M. (18 de agosto de 2022). *¿Qué es R software? © 2022 Máxima Formación.* Obtenido de https://www.maximaformacion.es/blog-dat/que-es-r-software/

GlosarioIT. (2022). *Pipeline - Sección Informática.* Obtenido de https://www.glosarioit.com/Pipeline

Hat, R. (10 de octubre de 2022). *What is Apache Kafka? © 2022 Red Hat, Inc.* Obtenido de https://www.redhat.com/es/topics/integration/what-is-apache-kafka

Hernández Sampieri, R. (2014). *Metodología de la Investigación.* México D.F.: Sexta Edición. McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V. ISBN: 978-1-4562-2396-0 ISBN: 978-607-15-0291-9 (de la edición anterior).

Hidalgo, S. (2014). *Random Forests para detección de fraude en medios de pago. PhD thesis.* Departamento de Ingeniería Informática. Universidad Autónoma de Madrid.

I., C. P. (2021). *ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECCIÓN DE FRAUDES CON TARJETAS DE CRÉDITO: ANÁLISIS Y COMPARATIVA.* ETSI Sistemas Informáticos, junio 2021.

Ikusi. (2022). *Detección de anomalías basada en el Machine Learning.* Obtenido de https://www.ikusi.com/mx/blog/deteccion-de-anomalias-basada-en-el-machine-learning/

ITELLIGENT, I. T. (4 de agosto de 2016). *10 ventajas de la minería de datos.* Obtenido de https://itelligent.es/es/10-ventajas-la-mineria-web/

J. O. Awoyemi, A. O. (2017). *Credit card fraud detection using machine learning techniques: A comparative analysis.* International Conference on Computing Networking and Informatics (ICCNI), 2017, pp. 1-9, doi: 10.1109 / ICCNI.2.

Kaur, D. a. (2020). *Machine Learning Approach for Credit Card Fraud Detection (KNN and Naïve Baiyes)-(March 30, 2020).* Proceedings of the International Conference on Innovative Computing and Communications (ICICC).

KeepCoding. (2022). *¿Qué es Scala y para qué se usa? KeepCoding Tech School © 2022*. Obtenido de https://keepcoding.io/blog/que-es-scala-y-donde-se-usa/

Kiran Sai, G. J. (2018). *Credit card fraud detection using Naïve Baiyes model based on KNN classifier.*

KWFundation. (2022). *¿Cómo elegir una plataforma de transmisión de datos?* Obtenido de https://kwfoundation.org/blog/2022/03/08/como-elegir-una-plataforma-de-transmision-de-datos/

Lavado Napaico, L. E. (2013). *A Genetic Algorithm For Fraud Detection Electronic Debit Cards In Peru. Un algoritmo genético para la detección de fraude electrónico en tarjetas de débito en Perú.* Revista de Investigación de Sistemas e Informática, RISI 10(1), 87-98 (2013) ISSN 1815-0268 (versión impresa) ISSN 1816-3823 (versión electrónica).

LLC, e. (2022). *Programación C++. © 2022 edX LLC.* Obtenido de https://www.edx.org/es/aprende/programacion-c-mas-mas

López, E. E. (2016). *Synthetic Financial Datasets For Fraud Detection, Synthetic datasets generated by the PaySim mobile money simulator.* Obtenido de kaggle: https://www.kaggle.com/ntnutestimon/paysim1#PS_20174392719_1491204439457_log.cs.

M. S. Kumar, V. S. (2019). *Credit Card Fraud Detection Using Random Forest Algorithm.* 3rd International Conference on Computing and Communications Technologies (ICCCT), 2019, pp. 149-153, doi: 10.1109 / ICCC.

Maimon, O. y. (2010). *Data Mining and Knowledge Discovery Handbook.* . Boston, MA: Springer US, 2010. https://doi.org/10.1007/978-0-387-09823-4.

Malebary, A. A. (2020). *An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine.* in IEEE Access, vol. 8, pp. 25579-25587, doi: 10.1109/ACCESS.2020.2971354.

Merino, M. (27 de enero de 2019). *Conceptos de inteligencia artificial: qué es el aprendizaje por refuerzo.* Obtenido de https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-aprendizaje-refuerzo

Microsoft. (22 de septiembre de 2022). *Paseo por el lenguaje C#. © 2022 Microsoft.* Obtenido de https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

MNV. (2022). *Programación en Shell. © 2022 MNV.* Obtenido de https://sites.google.com/a/ciencias.unam.mx/mnv/programacon-en-shell

Mohri, M. R. (2018). *Foundations of machine learning.* . Second edition. Cambridge, MA : The MIT Press.

Murphy, K. (2012). *Machine learning. A probabilistic perspective.* London: The MIT Press.

N. Shirodkar, P. M. (2020). *Credit Card Fraud Detection Techniques. A Survey.* International Conference on Emerging Trends in Information Technology and Engineering (ic ETITE), pp. 1-7, doi: 10.1109/ic-ETITE47903.2020.112.

Niuniu, L. Y. (2010). *"Improved ID3 algorithm."*. Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, 2010, pp. 465-468.

Profile Software Services, S. (25 de octubre de 2021). *TypeScript: Qué es, diferencias con JavaScript y por qué aprenderlo. © 2022 Profile Software Services, S.L.* Obtenido de https://profile.es/blog/que-es-typescript-vs-javascript/

Ramos Fernández, J. (2019). *Aprendizaje automático para flujos de datos.* . Escuela Técnica Superior de Ingenieros Informáticos: Universidad Politécnica de Madrid.

ServiceNow. (2022). *¿Qué es la detección de anomalías?* Obtenido de https://www.servicenow.com/es/products/it-operations-management/what-is-anomaly-detection.html

Services, A. W. (2022). *What is Java? © 2022 Amazon Web Services.* Obtenido de https://aws.amazon.com/es/what-is/java/

Tatamués., E. A. (2019). *Algoritmo de Random Forest aplicado a la detección de fraude en el sistema bancario ecuatoriano.* . Facultad de Ciencias : Escuela Politécnica Nacional.

TECHNOLOGIES., I. I. (4 de agosto de 2016). *10 ventajas de la minería de datos*. Obtenido de https://itelligent.es/es/10-ventajas-la-mineria-web/

Téllez, J. (2004). *Derecho Informático.* México: McGraw-Hill, 3ra edición.

Timarán-Pereira, S. R.-A.-Z.-T. (2016). *El proceso de descubrimiento de conocimiento en bases de datos. En Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional (pp. 63-86).* Bogotá: Ediciones Universidad Cooperativa de Colombia. doi: http://dx.doi.org/10.16925/9789587600490.

V. Vijayakumar, N. S. (2020). *Isolation Forest and Local Outlier Factor for Credit Card Fraud Detection System.* International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 8958, Volume 9 Issue 4, April 2020.

Venngage. (31 de mayo de 2022). *¿Qué es un diagrama de flujo de datos y cómo crear uno?* Obtenido de https://es.venngage.com/blog/diagrama-de-flujo-de-datos/

Víctor G. M. Murillo, M. J. (2019). *Análisis de fraude en transacciones bancarias aplicando Minería de Datos.* Facultad de Ciencias de la Computación, Benemérita:Universidad Autónoma de Puebla.

Vidhya, A. (6 de octubre de 2020). *Overcoming Class Imbalance using SMOTE Techniques*. Obtenido de https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

Vijayshree B. Nipane, P. S. (s.f.). *Deshpande Fraudulent Detection in Credit Card System Using SVM and Decision Tree.*

Zhang, J. Z. (2008). *Random-forests-based network intrusion detection systems.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 38:649 659.

ANEXOS

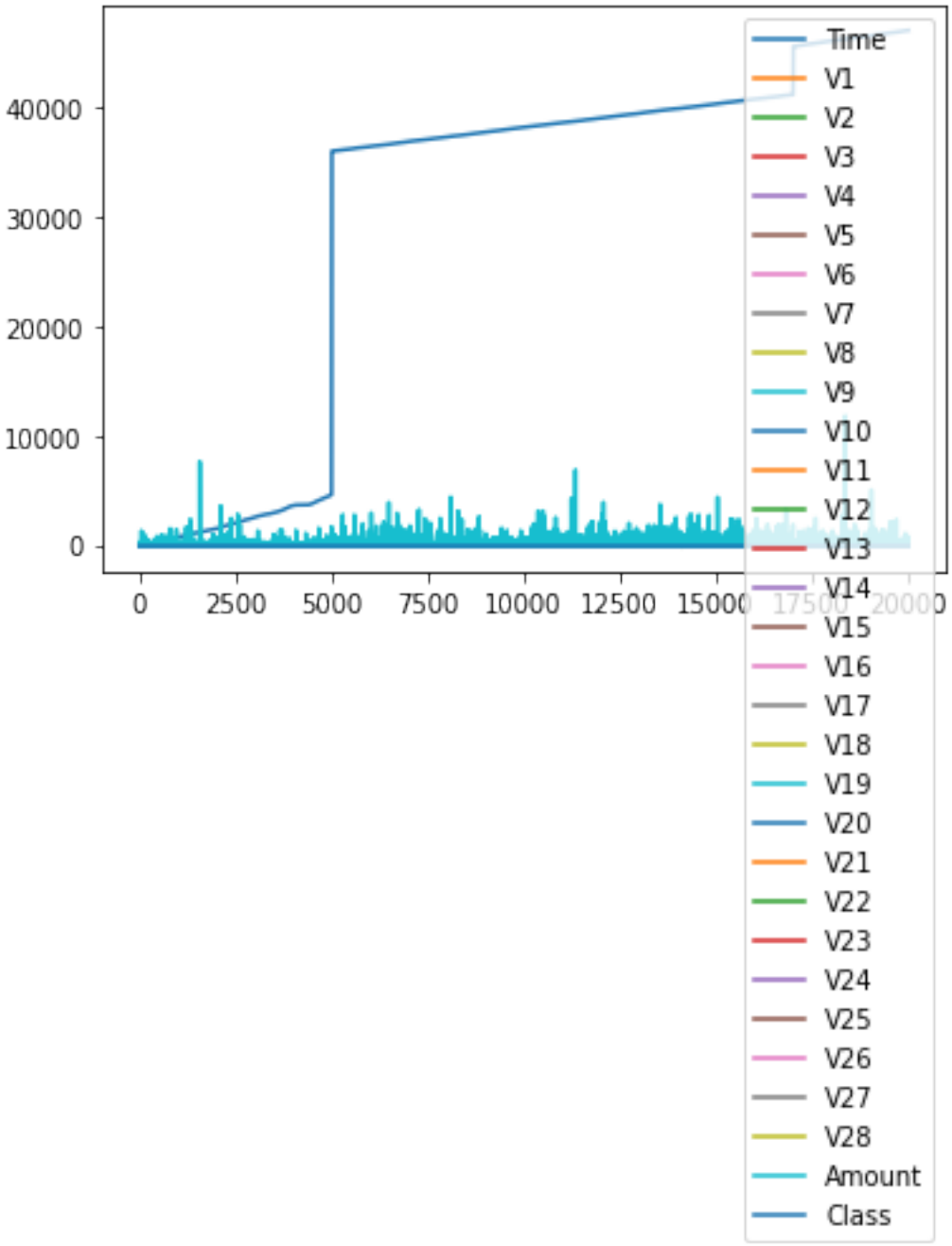


Figura 19. Conjunto de Datos Inicial.

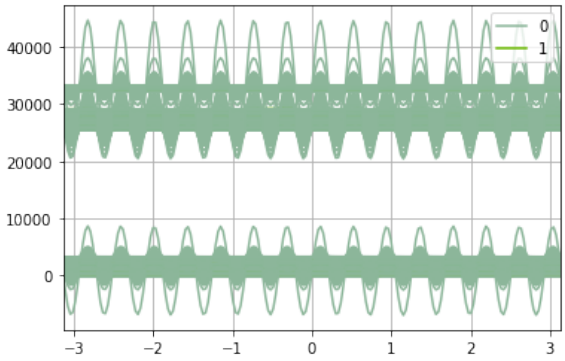


Figura 20. Curvas de Andrews ‘Class’.

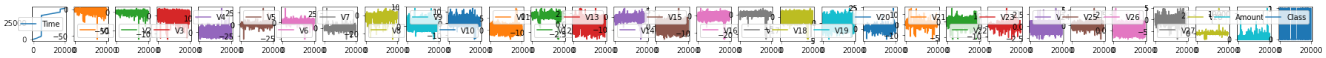


Figura 21. Características del conjunto de datos.

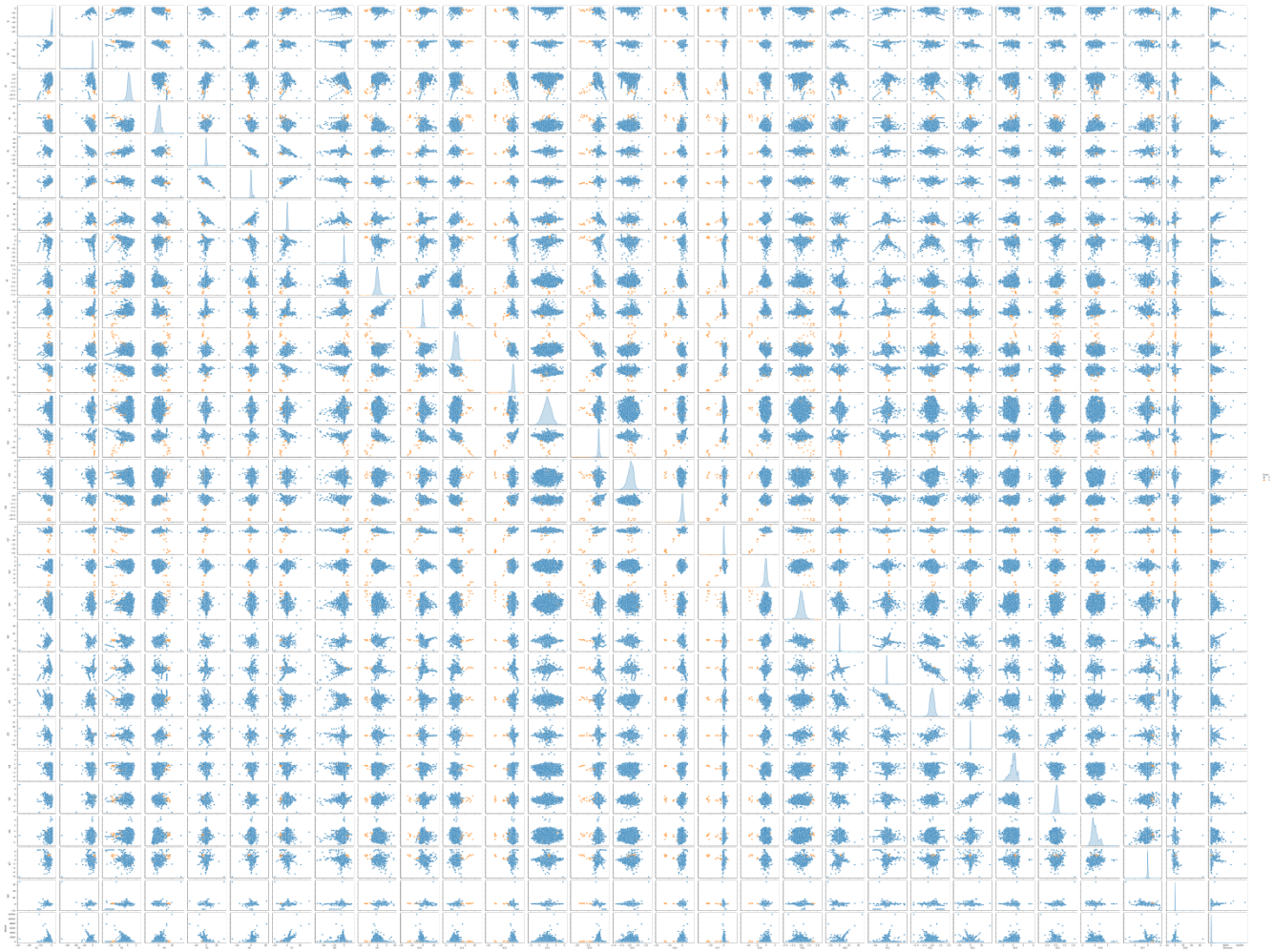


Figura 22. Valores Atípicos.

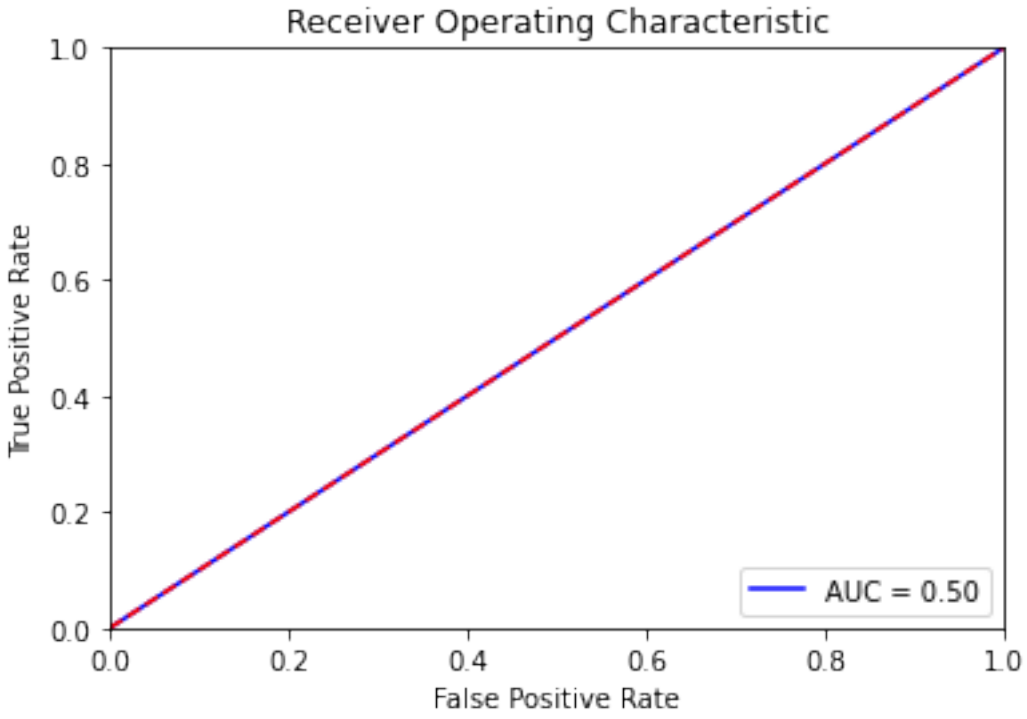


Figura 23. ANN AUC.

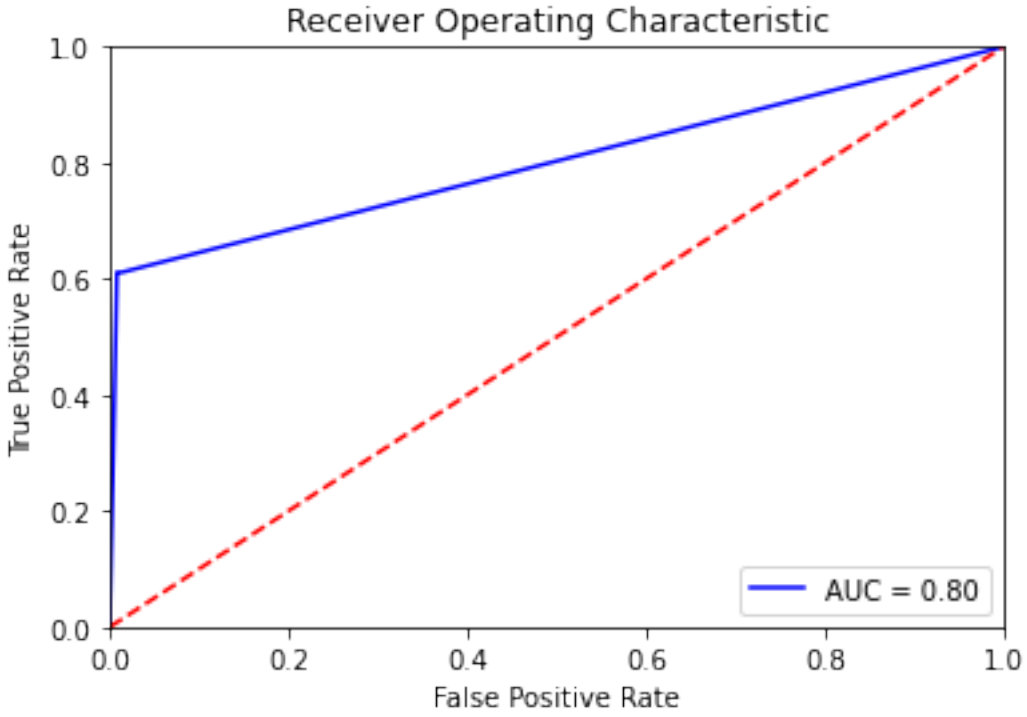


Figura 24. DT AUC.

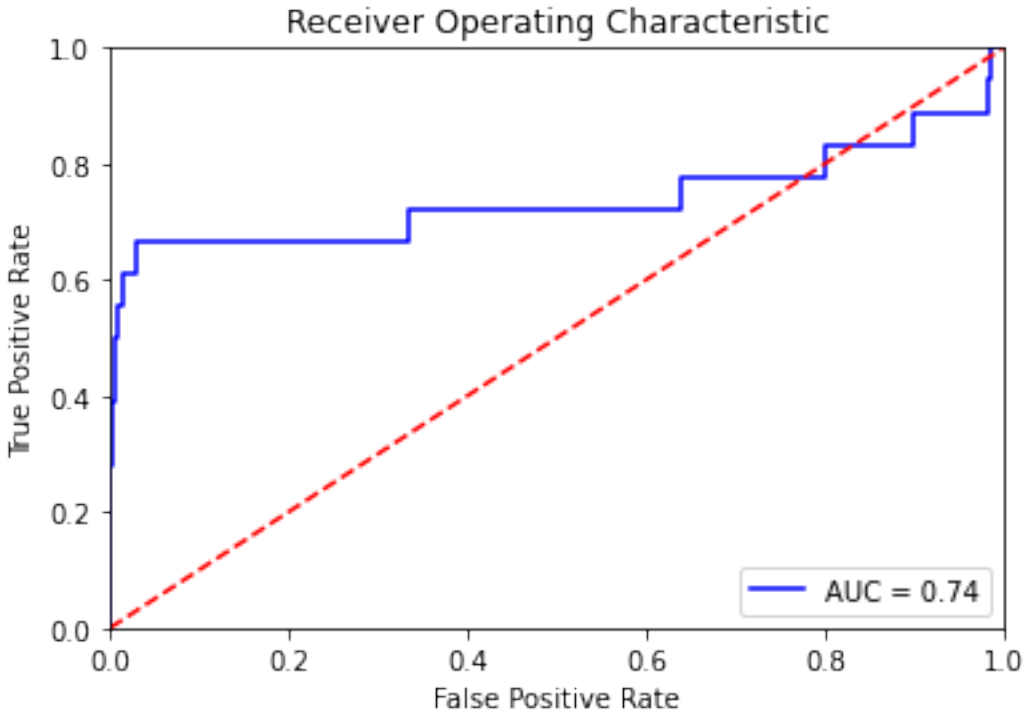


Figura 25. GBC AUC.

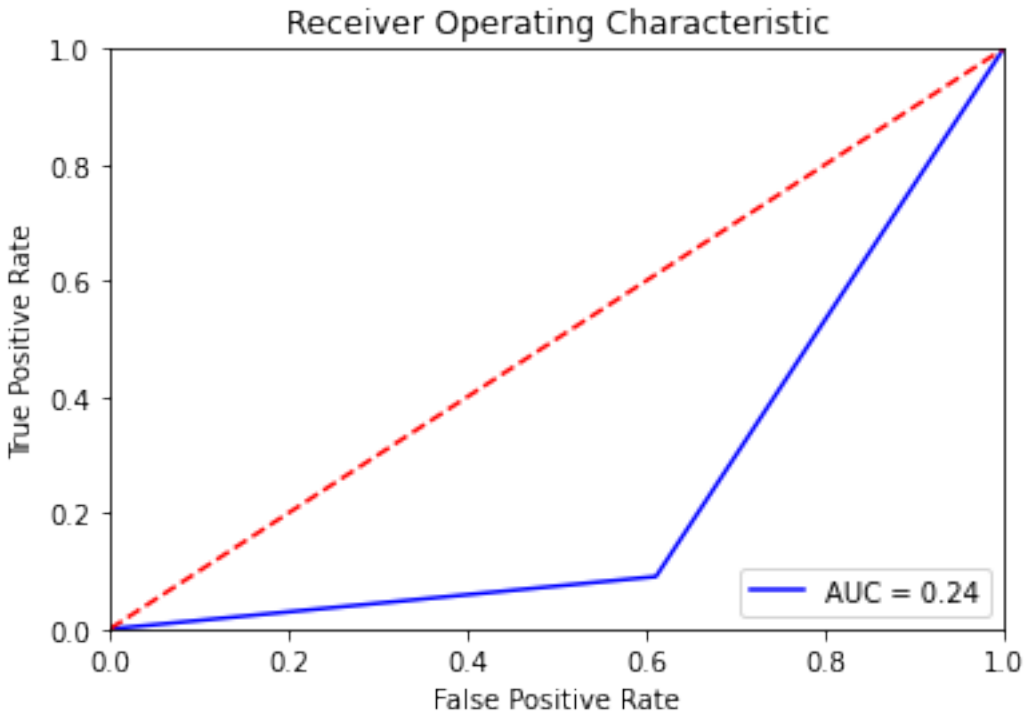


Figura 26. IF AUC.

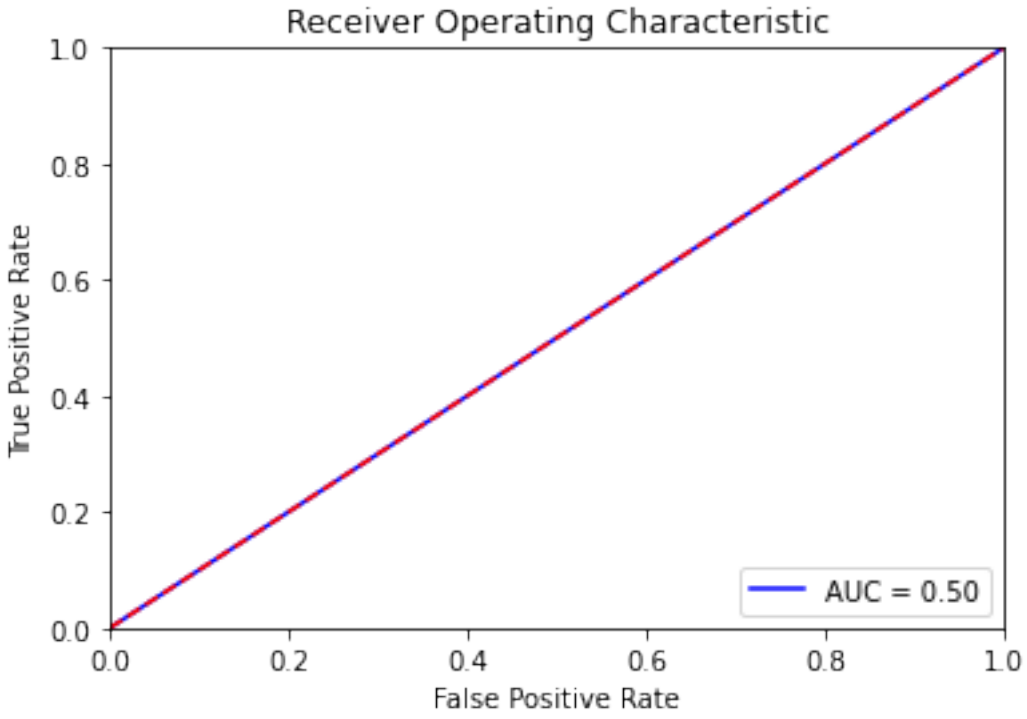


Figura 27. KNN AUC.

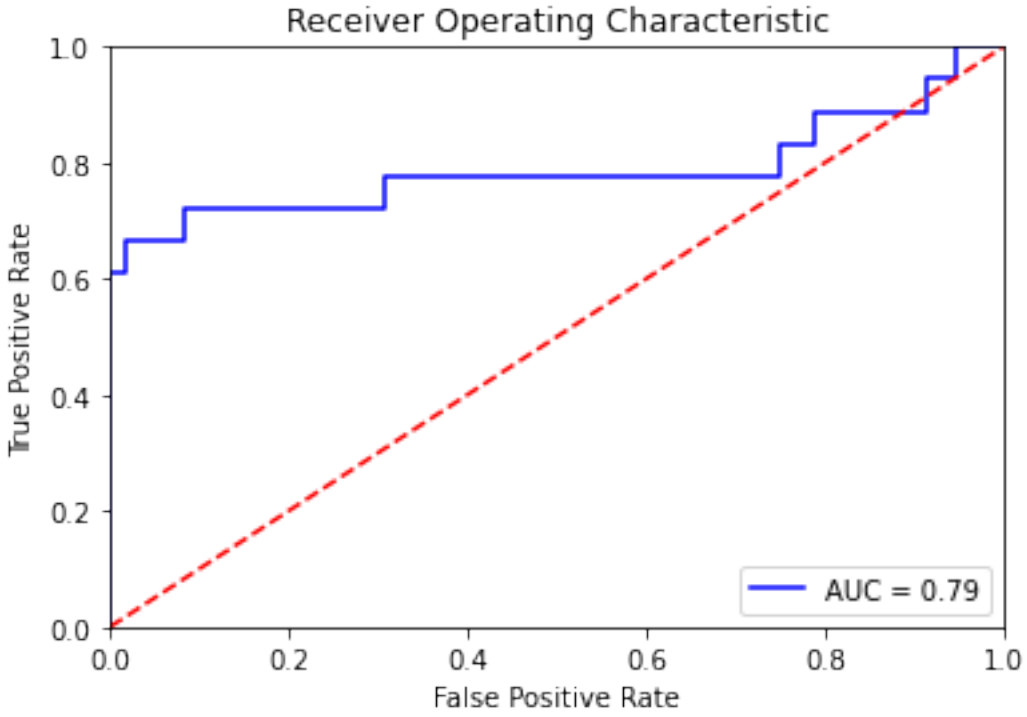


Figura 28. LR AUC.

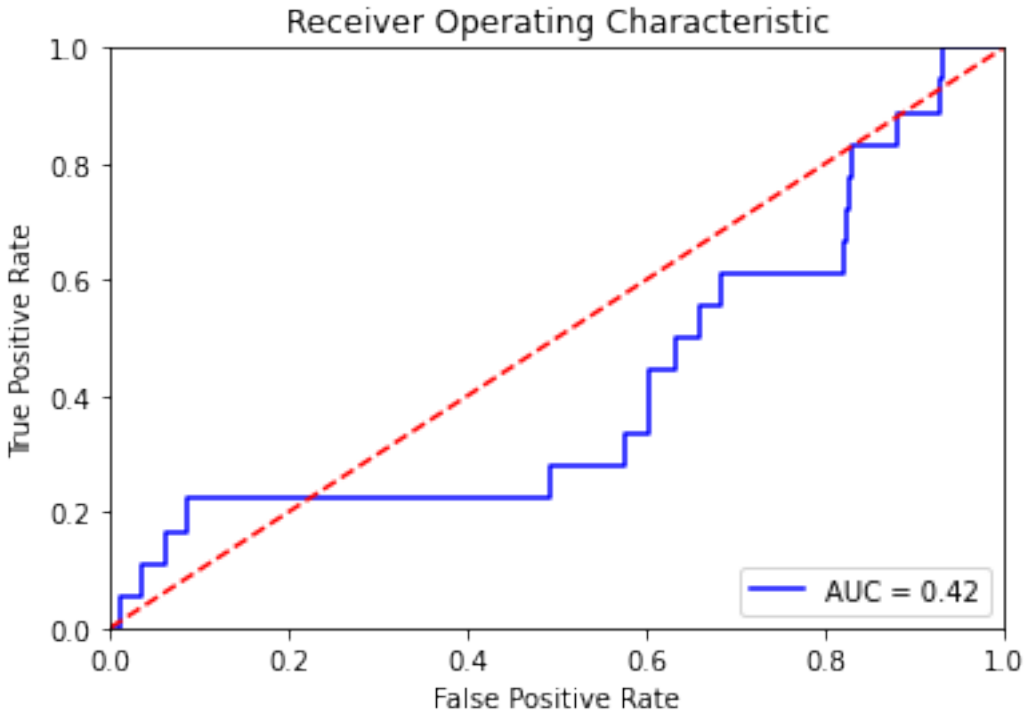


Figura 29. NBC AUC.

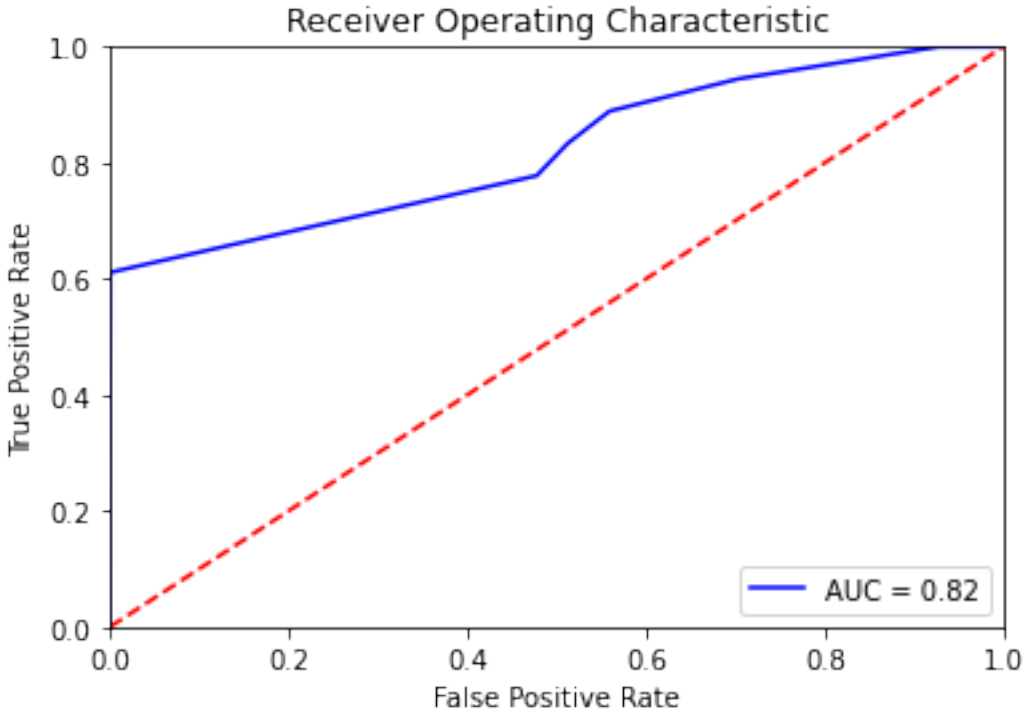


Figura 30. RF AUC.

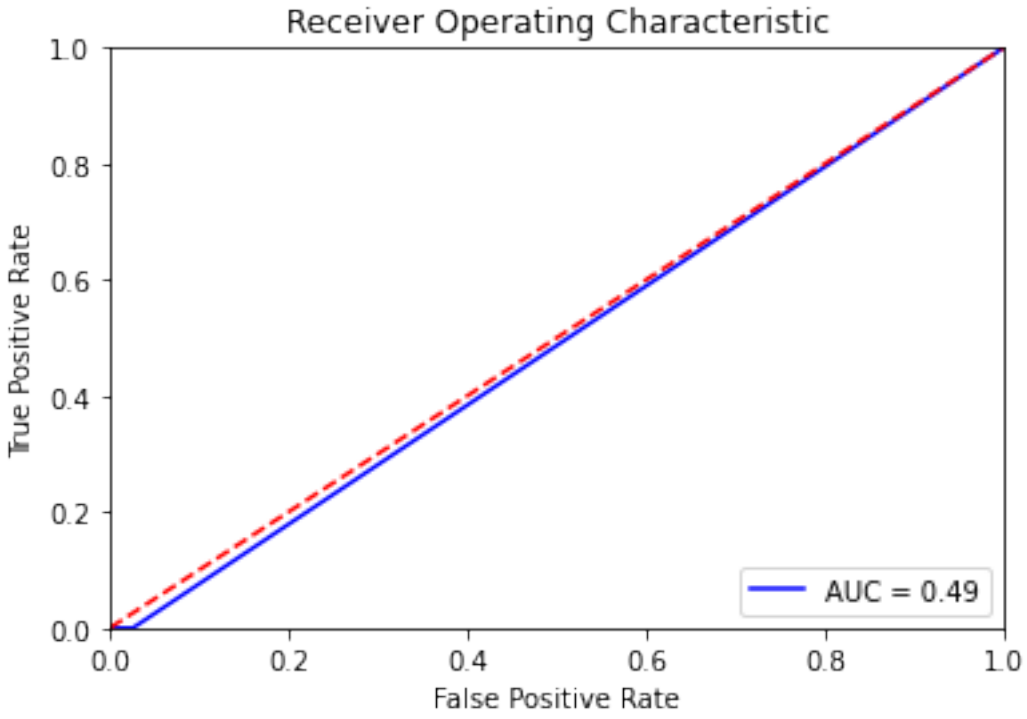


Figura 31. SVM AUC.