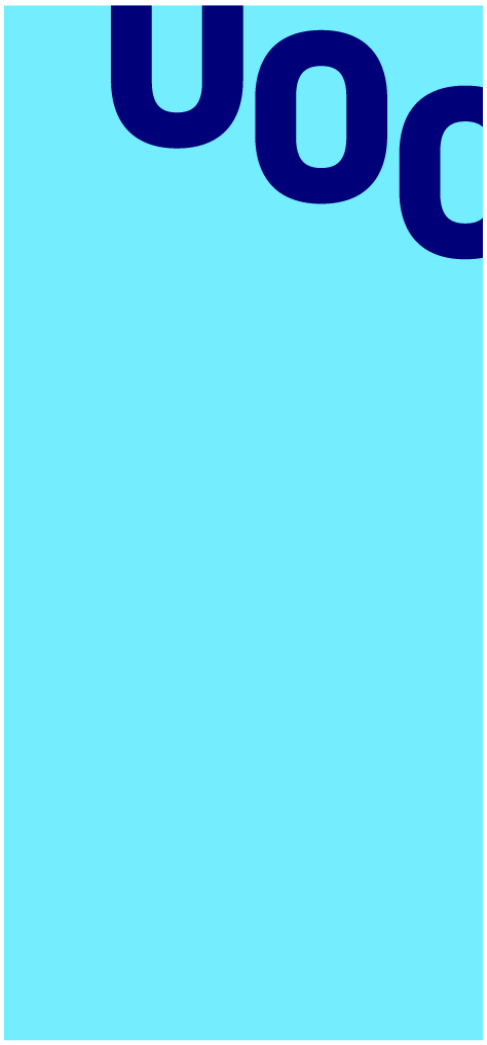
Universitat Oberta de Catalunya

uoc.edu

Prevención del fraude y blanqueo de capitales en entidades financieras.

Un caso de estudio



Universitat Oberta de Catalunya

Nombre Estudiante:
Juan Luis Espinoza López

Trabajo Final de Máster
Área: M2.982

Tutor/a de TF:
Jorge Segura Gisbert
Profesor/a responsable de la asignatura:
Albert Solé Ribalta

Fecha Entrega: 11 de junio del 2023

Ficha del Trabajo Final

Título del trabajo:	Prevención del fraude y blanqueo de capitales en entidades financieras. Un caso de uso.
Nombre del autor/a:	Juan Luis Espinoza López
Nombre del Tutor/a de TF:	Jorge Segura Gisbert
Nombre del/de la PRA:	Albert Solé Ribalta
Fecha de entrega:	06/2023
Titulación o programa:	Trabajo final de máster
Área del Trabajo Final:	M2.982
Idioma del trabajo:	Castellano
Palabras clave	Machine Learning, Financial data analysis, fraud detection, Data Mining

Resumen del Trabajo

Desde la llegada de Internet y las nuevas tecnologías, la revolución digital ha aumentado y se ha infiltrado en todos los aspectos de nuestras vidas. Una de las revoluciones digitales más importantes ocurrió en el sistema financiero y especialmente en la transacción de dinero a alguien de cualquier parte del mundo digitalmente. Las transacciones digitales se han convertido en parte de la vida diaria, como comprar un producto en línea, enviar dinero a amigos, depositar efectivo en una cuenta bancaria, fines de inversión, etc. Tenían muchos beneficios y allanaron el camino para actividades fraudulentas. La gente comenzó a usar medios de transacciones de dinero digital para lavar dinero y hacer parecer que el dinero proviene de una fuente legal. Además, mediante estas tramas delictivas es bien sabido que es una práctica común a la hora de financiar el terrorismo, perpetrado a los largos de los últimos años cobrándose miles de vidas y afectando a la sociedad por el temor de que algo pueda pasar en cualquier momento y en cualquier parte del mundo.

Es por eso que hoy en día las instituciones financieras intentan evitar a toda costa tanto el fraude cómo el blanqueo de capitales. Para ellos utilizan programas que se encargan de analizar enormes cantidades de datos para

poder así detectar transacciones fraudulentas y si aplica parar dicha transacción y hacer la diligencia respectiva.

En este trabajo se partirá de un conjunto de datos financiero ficticio en el cual tendremos una variable objetivo “fraude” donde primeramente se procesarán los datos, buscando inconsistencias, outliers o simplemente adaptar los datos según nuestra conveniencia para posteriormente aplicar algoritmos de machine learning y ver si efectivamente el algoritmo clasifica la transacción cómo fraudulenta o no.

Se implementarán diversos métodos y veremos la performance de cada uno de ellos para ver con cual obtenemos mejor rendimiento un ratio de aciertos elevados.

Abstract

Since the arrival of the Internet and new technologies, the digital revolution has increased and has been present in all aspects of our lives.
One of the most important digital revolutions occurred in the financial system and especially in the digital transaction of money to someone from anywhere in the world. Digital transactions have become a part of daily life such as buying online products, sending money to friends, depositing cash into a bank account, investment purposes, etc. They had many benefits and also paved the way for fraudulent activities. People started using digital money transaction means to launder money and make it look like the money came from a legal source.

In addition, through these criminal plots it is well known that it is a common practice when it comes to financing terrorism, perpetrated over the last few years, claiming thousands of lives, and affecting society for fear that something could happen at any time and anywhere in the world.

That is why today financial institutions try to avoid both fraud and money laundering at all costs. For them, they use programs that are responsible for analyzing huge amounts of data to detect fraudulent transactions and, if applicable, stop said transaction and do the respective diligence.

In this work, we will start from a fictitious financial data set in which we will have an objective variable "fraud" where the data will first be processed, looking for inconsistencies, outliers or simply adapting the data according to our convenience to later apply machine learning algorithms and see if indeed the algorithm classifies the transaction as fraudulent or not.

Various methods will be implemented, and we will see the performance of each one of them to see with which we obtain the best performance with a high hit ratio.

Índice

1. Introducción

1.1. Contexto y justificación del Trabajo

El **lavado de dinero [1]** es el proceso a través del cual es encubierto el origen de los fondos generados mediante el ejercicio de algunas actividades ilegales (siendo las más comunes, tráfico de drogas o estupefacientes, contrabando de armas, corrupción, fraude, trata de personas, prostitución, extorsión, piratería, evasión fiscal y terrorismo). El objetivo de la operación, que generalmente se realiza en varios niveles, consiste en hacer que los fondos o activos obtenidos a través de actividades ilícitas aparezcan como el fruto de actividades legítimas y circulen sin problema en el sistema financiero.

El antiguo director gerente del FMI **[8]** (Fondo Monetario Internacional) Michel Camdessus estimó en 1998 que entre el 2 y el 5 por ciento del PIB mundial es lavado anualmente lo que equivale a aproximadamente de $ 800 mil millones a $ 2 billones en solo un año. Está demostrado que el Money Laundering financia ataques terroristas mundial. Por lo tanto, la investigación contra el lavado de dinero (AML) es de importancia crítica para estabilidad financiera nacional y seguridad internacional.

Estados Unidos fue una de las primeras naciones en introducir leyes contra el lavado de dinero con la Ley de Secreto Bancario (BSA) en 1970. Estas leyes de la BSA fueron un primer intento de detectar y prevenir el lavado de dinero y, desde entonces, han sido enmendadas y fortalecidas por otras leyes contra el lavado de dinero. La Red de Ejecución de Delitos Financieros es ahora la agencia responsable de administrar estas leyes. Su misión es proteger al sistema financiero de los abusos de los delitos financieros, incluido el financiamiento del terrorismo, el lavado de dinero y otras actividades ilegales.

En 1989, varios países y organizaciones fundaron el Grupo de Acción Financiera Internacional (GAFI) o en inglés Financial Action Task Force (FATF). Su tarea es desarrollar y hacer cumplir las pautas internacionales contra el lavado de dinero. Poco después del 11 de septiembre, el FATF amplió su mandato para incluir AML (Anti-Money Laundering) y contrarrestar el financiamiento del terrorismo.

Desde los ataques terroristas de 2001, [**2]** el GAFI ahora también incluye vigilancia terrorista en un esfuerzo por mitigar el financiamiento del terrorismo. Recientemente, la criptomoneda ha sido objeto de escrutinio, ya que brinda anonimato a sus usuarios. Esto ha facilitado un método de menor riesgo para que los delincuentes realicen sus transacciones.

Actualmente, las instituciones financieras se exponen a diversos tipos de multas dependiendo de la organización que haga la auditoria o la due-diligence. Las multas si bien

son grandes cantidades de dinero (miles de millones de euros/dólares), pueden tener un impacto reputacional muy fuerte en las instituciones financieras perdiendo la confianza de clientes, inversores, etc.

Por ello, las instituciones financieras ya han comenzado a montar sus propios departamentos para cumplir con las obligaciones impuestas por las organizaciones como la FATF, GAFI, etc. en la lucha contra el blanqueo de capitales y el fraude. Estos departamentos están formados por profesionales de perfil técnico con habilidades para analizar datos, implementar algoritmos, etc. pero además habrá otro tipo de profesionales con experiencia en el negocio cubriendo así un gran abasto de conocimiento cómo el tipo de instituciones financieras (retail banking, investment banking, correspondent banking, etc.), el flujo de análisis de alertas generadas, etc. De esta manera el trabajo en sintonía de estos profesionales servirá para detectar transacciones fraudulentas y por lo tanto cumplir con las directrices impuestas por dichas organizaciones.

1.2. Motivación personal

El terrorismo es algo que lleva presente en la sociedad y sobre todo durante las últimas décadas. Pensábamos que solo pasaba en países alejados de occidente, pero no es así y es un hecho que occidente está en el punto de mira de estos terroristas. Cómo persona y en los últimos 5 años cómo profesional, ayudar a instituciones financieras a mitigar este tipo de riesgos es algo que me motiva y disfruto haciéndolo en mi día a día.

Desde los ataques terroristas de 2001 en New York, el GAFI ahora también incluye vigilancia terrorista en un esfuerzo por mitigar el financiamiento del terrorismo. Recientemente, la criptomoneda ha sido objeto de escrutinio, ya que brinda anonimato a sus usuarios. Esto ha facilitado un método de menor riesgo para que los delincuentes realicen sus transacciones.

Las industrias bancarias y financieras buscan la manera de protegerse contra el fraude a todos los niveles y el blanqueo de capitales. El objetivo del lavado de dinero es disfrazar los fondos obtenidos ilegalmente (dinero sucio) de tal manera que se dé la impresión de que provienen de una fuente de dinero acreditada.

Para ello los departamentos de compliance tienen muy en cuenta la AML (Anti-Money Laundering). AML se refiere a las actividades que las instituciones financieras realizan para cumplir con los requisitos legales para monitorear activamente y reportar actividades sospechosas.

Cómo integrante del departamento de AFC (Anti Financial Crime) de una importante institución bancaria alemana, la realización de este proyecto me motiva a explorar otras metodologías para reducir la generación de alertas que acabaran siendo falsos positivos y de esta manera ayudar al banco a reducir recursos (económicos, personal, tiempo, etc.)

investigando este tipo de alertas y reducir el tiempo desde que se genera hasta que se cierra la alerta.

1.3. Objetivos del Trabajo

1.3.1. Objetivo principal

Con la llegada de las transacciones digitales, la posibilidad de lavado de dinero también se ha disparado con el uso de la tecnología. Millones de investigadores están en el campo luchando contra las transacciones fraudulentas. En la industria actual, tenemos una gran afluencia de resultados de falsos positivos y lleva mucho tiempo eliminar los resultados de falsos positivos.

Por ejemplo, los clientes de todo el mundo que utilizan plataformas fintech exigen servicios fiables y además rápidos. Por lo tanto, nuestro objetivo es automatizar los resultados con el aprendizaje automático y reducir los resultados positivos falsos. Pero no a costa de dejar fuera los falsos negativos. Por lo tanto, debemos ser más conscientes de los falsos negativos cuando tratamos de reducir los falsos positivos.

Dada la base de datos que tenemos podemos decir que tenemos un problema de clasificación. Se utilizarán diversos algoritmos y modelos en los que se el algoritmo utilizado tendrá que decidir si se trata de una transacción fraudulenta (1) o no fraudulenta (0).

1.3.2. Objetivos secundarios

Hay una falta de datos disponibles de carácter público sobre servicios financieros y especialmente en el dominio emergente de transacciones de dinero. Parte del problema es la naturaleza intrínsecamente privada de las transacciones financieras, que conduce a que no haya conjuntos de datos disponibles públicamente.

Debido a la privacidad de los datos, es difícil encontrar datos de transacciones de carácter público. Por lo tanto, para llevar a cabo este proyecto usaremos datos simulados por Paysim que simula transacciones de dinero basadas en una muestra de transacciones reales extraídas de un mes de registros financieros de un servicio de dinero móvil implementado en un país africano.

Para llevar a cabo el objetivo principal comentado en el punto anterior se partirá de otros procesos y tareas, partiendo de unos datos expuestos anteriormente. Las tareas serán: el análisis del ciclo de vida de estos datos hasta la final predicción del modelo, entrenamiento del modelo y el testing del modelo.

Podemos obtener una lista aproximada de objetivos que a lo mejor cambia a lo largo del proyecto en función de los inconvenientes que me encuentre durante del desarrollo **[5]:**

- Conocimiento del negocio
- Recopilación de datos
- Limpieza de los datos
- Análisis exploratorio de los datos
- Preprocesamiento de los datos
- Extracción de características
- Desarrollo del modelo
- Comparación entre modelos
- Conclusión del proyecto

1.4. Enfoque y método seguido

La legislación AML se está volviendo cada vez más estricta para los proveedores de servicios financieros. Debe evitarse que se financie el blanqueo de capitales y/o el terrorismo siguiendo las pautas de la OFAC (Office of Foreign Assests Control) y las recomendaciones del FATF (Financial Action Task Force). Estos procedimientos se utilizan para identificar y bloquear las actividades fraudulentas en el sistema. Se implementa un algoritmo basado en reglas en el sistema para generar alertas cada vez que se identifican transacciones fraudulentas. Estas alertas luego son investigadas por los analistas y tomarán decisiones sobre si esta transacción es un fraude (Verdadero positivo) o no fraude (Falso positivo).

En el escenario actual, hay muchas alertas de falsos positivos dadas por el algoritmo y los investigadores cierran esas alertas dentro de un tiempo estipulado. Dado que hay una gran

cantidad de alertas de falsos positivos, se invierte más tiempo en eliminar las alertas y las organizaciones requieren una gran cantidad de recursos humanos para investigar estas alertas. Por lo tanto, se desperdicia mucho tiempo y dinero solo porque el algoritmo basado en reglas no es lo suficientemente inteligente para identificar las transacciones no fraudulentas.

A continuación, se explicará el método seguido para desarrollar este proyecto. Se deberá tener claro que modelo se seguirá pues esto nos ayudará a tener una mayor organización en cuanto a seguir bien los pasos a desarrollar en los siguientes capítulos.

Para el desarrollo de este proyecto se va utilizar como modelo de referencia el **Cross Industry Standard Process for Data Mining)** para la minería de datos (CRISP-DM) **[6]** ya que es un conocido modelo de referencia de minería de datos. La metodología CRISPDM es flexible y permite la creación de un modelo que se adapta a las necesidades específicas de los proyectos. La secuencia de ejecución de las fases no es rígida y depende de los resultados obtenidos en cada fase (Figura x):

El ciclo de vida del proyecto en esta metodología consta de seis fases:

1. **Entendimiento del negocio**: Esta fase inicial se enfoca en comprensión de los objetivos y requisitos del proyecto desde una perspectiva comercial, y luego convertir ese conocimiento en una definición del problema de la minería de datos y un plan preliminar diseñado para alcanzar los objetivos.

2. **Comprensión de datos:** Comienza la fase de comprensión de datos con la recolección inicial de datos y continúa con actividades que permiten la familiarización con los datos, la identificación de problemas de calidad de datos, es decir, si los datos son consistentes, si hay valores atípicos, etc. el descubrimiento de los primeros conocimientos sobre los datos y/o la detección de subconjuntos interesantes para formar hipótesis sobre la información desconocida.

3. **Preparación de datos:** La fase de preparación de datos concentra todas las actividades necesarias para la construcción del conjunto de datos final que se utilizará en la fase de modelado. La preparación de datos se suele realizar varias veces y no en ningún orden en concreto. Las tareas incluyen seleccionar, limpiar, construir, integrar y formatear datos con fines de modelado, por ejemplo, cambiar el tipo de variables de algunos campos para poder trabajar con modelos predictivos.

4. **Modelado:** En esta etapa, varias técnicas de modelado ya se han elegido y aplicado y sus parámetros se ajustan a los valores óptimos. Por lo general, hay diferentes técnicas para el mismo problema de minería de datos. Algunas técnicas tienen requisitos específicos en cuanto a la forma de los datos. Por lo tanto, a menudo es necesario volver a fases previas para realizar ajustes. En este proyecto habrá diferentes tipos de modelos: modelos supervisados y no supervisados.

El **aprendizaje supervisado [3]** es un enfoque de aprendizaje automático que se define mediante el uso de conjuntos de datos etiquetados. Estos conjuntos de datos están diseñados para entrenar o "supervisar" algoritmos para clasificar los datos o predecir los resultados con precisión. Usando entradas y salidas etiquetadas, el modelo puede medir su precisión y aprender con el tiempo. El aprendizaje supervisado puede separarse en dos tipos de problemas en la minería de datos: **clasificación y regresión**

El **aprendizaje no supervisado [4]** utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de datos sin etiquetar. Estos algoritmos descubren patrones ocultos en los datos sin necesidad de intervención humana (por tanto, están "sin supervisar").
Los modelos de aprendizaje no supervisado se utilizan para tres principales tareas: agrupación, asociación y reducción de la dimensionalidad.

Cabe destacar que existen muchas técnicas o algoritmos que podrían adaptarse perfectamente a los requerimientos del proyecto, sin embargo, existe una limitación de tiempo que impide que todas ellas sean estudiadas y mejoradas. Es por este motivo que para la realización del proyecto únicamente se seleccionarán aquellas que se crean más relevantes y que se ajusten, tanto a los tiempos como objetivos definidos del trabajo.

5. **Evaluación:** En esta fase es importante evaluar y revisar los pasos realizados para crear el modelo final (o modelos), antes del despliegue final, para asegurarse de que Se logra los objetivos del negocio. Es importante tratar de determinar si hay aspectos de negocio importantes aun no considerados. Se aplicarán métricas que permitan compararlos para seleccionar aquel o aquellos que tengan un porcentaje de exactitud o precisión más alto a la hora de detectar transacciones fraudulentas que realmente lo son. Se compararán los resultados de diversos algoritmos o modelos utilizados para ver si se ha conseguido mejorar el porcentaje de aciertos. En este paso también se utilizarán técnicas de análisis como tablas y gráficas que permitan interpretar de manera intuitiva y amigable que algoritmo o algoritmos utilizados son los que mejores resultados presentan. Al final de esta fase, es importante decidir si los resultados son satisfactorios y si el modelo final se debe utilizar o no.

6. **Despliegue:** El conocimiento obtenido con los modelos generado debe ser aplicado en la entidad financiera o banco y este conocimiento debe ser difundido y presentado a los usuarios de manera que puedan usarlo.

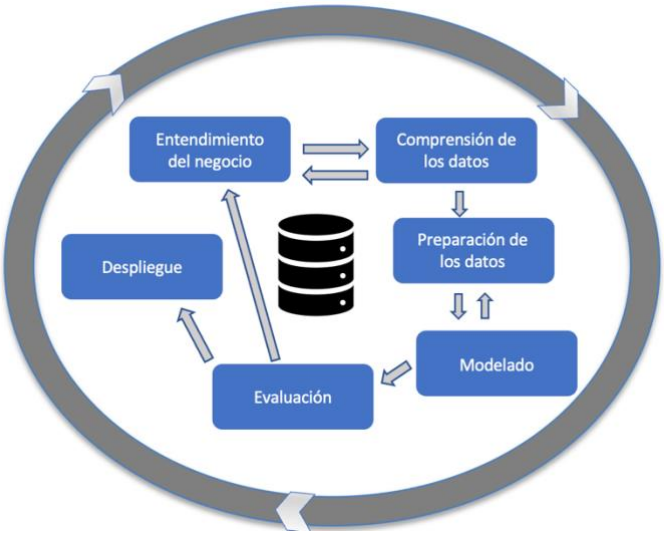


Figura 1: Flujo CRISP-DM

1.5. Planificación del trabajo

La planificación de cualquier proyecto para su correcto desarrollo y consecución es esencial. Para la presente tesis se ha desarrollado una planificación acorde a los tiempos que se consideran necesarios para la realización de un proyecto de este calibre.
En primer lugar, se han definido los distintos recursos que van a ser necesarios para el desarrollo del proyecto. Dichos recursos son los siguientes:

- **Ordenador personal MacBook** Pro: es el único componente hardware requerido para este proyecto, el cual se utilizará tanto para el desarrollo teórico como práctico de la tesis.

- **R Studio**: se utilizará como lenguaje de programación principal, donde se utilizarán las principales librerías cómo dplyr, ggplot2, kntir entre otras.

- **Sistema operativo MacOS**: es el sistema operativo del ordenador personal.

- **Microsoft Office 365**: Se utilizará para la redacción de la memoria de la tesis y la creación de la presentación para la defensa de la tesis.

En cuanto a la planificación general del proyecto, se han identificado 6 fases o hitos que deben cumplirse:

- **Fase 1** – Definición y planificación del proyecto (12 días). En esta fase se establecen las hipótesis y principales objetivos que se pretenden cumplir para la correcta ejecución del proyecto. Asimismo, se definen las distintas tareas y el tiempo que se le va a dedicar a cada una.

- **Fase 2** – Estado del arte o análisis de mercado (13 días). En esta fase se realiza una investigación en profundidad de otros trabajos científicos relacionados con el tema principal del proyecto. Una vez seleccionada la información, se clasifica y se selecciona aquella que se considere relevante para el proyecto.

- **Fase 3** – Diseño e implementación (62 días). En esta fase se diseñan y simulan las distintas problemáticas que abarca el proyecto y se desarrollan e implementan las distintas soluciones. Es el hito más importante y que conlleva más tiempo en llevarse a cabo del proyecto.

- **Fase 4** – Redacción de la memoria (27 días). En esta fase se unirán todas las piezas realizadas a lo largo del proyecto con el objetivo de generar una memoria de este.

- **Fase 5** – Preparación de la defensa (6 días). En esta fase se va a crear y preparar la presentación de la defensa del proyecto.

- **Fase 6 –** Defensa Pública (12 días). En la última fase se presentará la tesis del máster ante un jurado establecido por la universidad.

En cada una de las fases divididas existen un conjunto de tareas que deben llevarse a cabo para poder considerar que cada hito o fase se han cumplido:

- Búsqueda y clasificación de información sobre estado del arte (6 días)
- Lectura y comprensión del estado del arte (11 días)
- Preparación y elaboración de la información recopilada (4 días)
- Búsqueda de algoritmos de aprendizaje automático (7 días)
- Análisis de problemáticas de los tipos de algoritmos (7 días)
- Selección de los algoritmos a utilizar (5 días)
- Ejecución de los modelos seleccionados (20 días)

- Testeo de los modelos desarrollados (5 días)
- Comparación de algoritmos utilizados (13 días)
- Conclusiones del proyecto (3 días)
- Revisión del contenido del documento (8 días)
- Revisión de la estructura del documento (7 días)
- Revisión del documento después del feedback de la primera entrega (12 días)
- Creación de la presentación power point (3 días)
- Preparación de la defensa del trabajo (3 días)

En la figura 1.1 se puede observar la planificación del proyecto con las distintas fases definidas, sus tareas asociadas y las fechas de inicio y fin estimadas:

Name	Begin date	End date
Fase 1 – Definición y planificación del proyecto	1/3/23	12/3/23
Fase 2 – Estado del arte o análisis de mercado	13/3/23	26/3/23
Búsqueda y clasificación de información sobre estado del arte	13/3/23	18/3/23
Lectura y comprensión del estado del arte	19/3/23	22/3/23
Preparación y elaboración de la información recopilada	23/3/23	26/3/23
Fase 3 – Diseño e implementación	27/3/23	28/5/23
Búsqueda de algoritmos de aprendizaje automático	27/3/23	3/4/23
Análisis de problemáticas de los tipos de algoritmos	4/4/23	11/4/23
Selección de los algoritmos a utilizar	12/4/23	16/4/23
Ejecución de los modelos seleccionados	17/4/23	7/5/23
Testeo de los modelos desarrollados	8/5/23	12/5/23
Comparación de algoritmos utilizados	13/5/23	25/5/23
Conclusiones del proyecto	26/5/23	28/5/23
Fase 4 – Redacción de la memoria	29/5/23	25/6/23
Revisión del contenido del documento	29/5/23	6/6/23
Revisión de la estructura del documento	7/6/23	11/6/23
Revisión del documento después del feedback de la primera entrega	12/6/23	25/6/23
Fase 5 – Preparación de la defensa	26/6/23	2/7/23
Creación de la presentación power point	26/6/23	28/6/23
Preparación de la defensa del trabajo	29/6/23	2/7/23
Fase 6 – Defensa Pública	3/7/23	14/7/23

Figura 2: Diagrama de tiempo del proyecto

La figura 1.2 muestra el diagrama de Gantt generado con la planificación del proyecto, el cual permite ver de forma visual las principales fases del proyecto, las tareas asociadas y la duración de cada una. Esta vez he utilizado la app team Gantt proporcionado en los recursos del campus virtual.

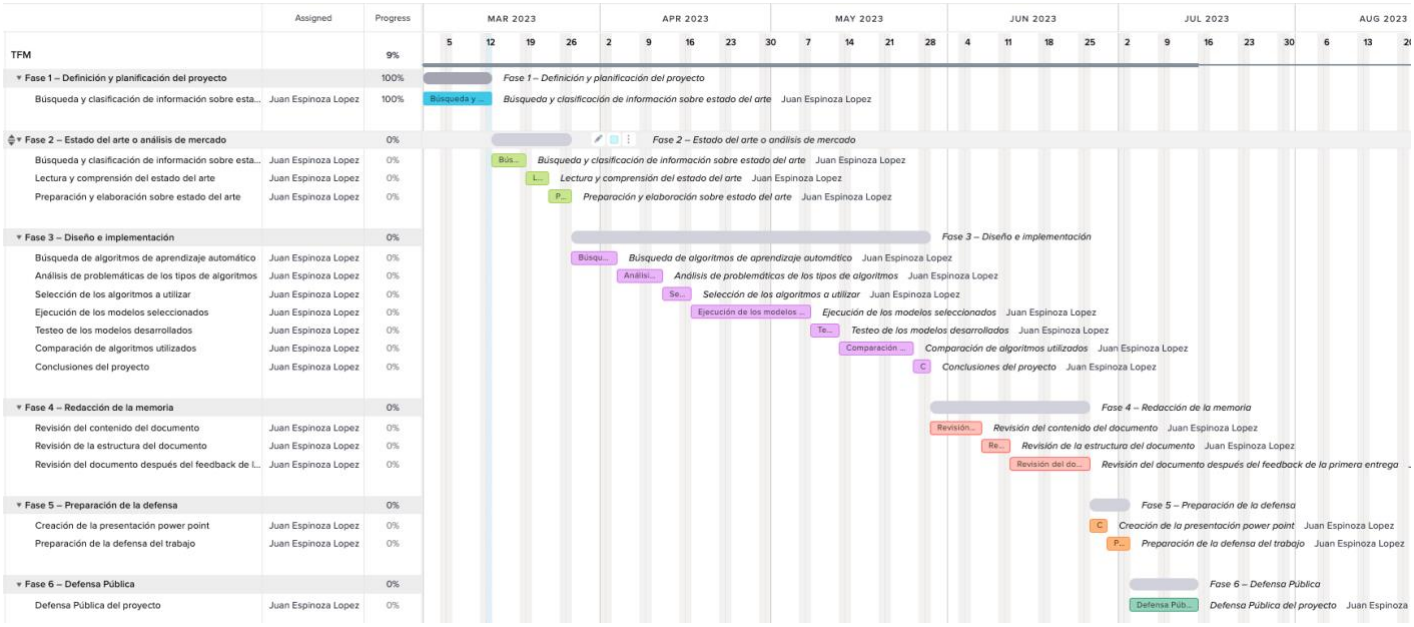


Figura 3: Diagrama de Gantt

1.6. Breve sumario de los productos obtenidos

Los productos obtenidos han sido diversos modelados. Hemos aplicado algoritmos supervisados y no supervisados. Cómo era de esperar los algoritmos supervisados han conseguido mejor rendimiento ya que tienen una etiqueta que es la variable objetivo.

De todo los algoritmos supervisados el que mejor rendimiento ha dado es el Random Forest y en el lado de los algoritmos no supervisados, el que mejor precisión obtuvo fue el algoritmo de clustering K-Means, donde se han obtenido 5 clusters diferentes.

En los próximos capítulos, veremos con más detalle de cifras el rendimiento que han dado, así como los diferentes parámetros que han usado los algoritmos o el tiempo de ejecución que alguno de ellos.

1.7. Breve descripción de otros capítulos de la memoria

El **capítulo 2** del proyecto es un análisis del estado del arte, donde se ha buscado información relacionada al blanqueo de capitales. Desde los problemas que hoy en día

existen a las posibles soluciones que se pueden aplicar usando diferentes métodos, algoritmos, etc. Además, se ha visto que a día de hoy ya existen algunos softwares en el mercado con dicha tecnología y los bancos ya usan uso de ellos.

El **capítulo 3** del proyecto consiste en tratar los datos que se utilizan en este proyecto. Preprocesamiento de datos, desde la carga hasta la limpieza, extracción de outliers, etc. En este capítulo empezaría la implementación del proyecto.

El **capítulo 4** del proyecto presenta los análisis gráficos de las variables que presentan los datos del capítulo anterior. En este capítulo se extrae mucha información sobre las variables que tenemos y por lo tanto nos ayudará en los capítulos posteriores.

El **capítulo 5** presenta la parte del modelado en donde se aplicarán los respectivos algoritmos ya seas supervisados o no supervisados y se verá con cual obtenemos mejor rendimiento. Además, conoceremos ventajas y desventajas de cada uno de estos tipos de aprendizajes.

El **capítulo 6** del proyecto es un resumen en modo de tabla de los resultados obtuvimos en los capítulos anteriores. De esta forma queda todo un poco más claro y conciso.

El **capítulo 7** del proyecto hace referencia a las limitaciones que podemos tener en los algoritmos tanto de aprendizaje supervisado cómo no supervisado.

El **capítulo 8** del proyecto es n resumen de las conclusiones obtenidas tras realizar el trabajo.

Finalmente, en el **capítulo 9**, se muestra un resumen de las líneas de trabajo futuras que se podrían añadir al proyecto realizado.

1.8. Impacto en sostenibilidad, ético-social y de diversidad

La competencia de compromiso ético y global (CCEG) está definida por nivel de Máster cómo: actuar de forma honesta, ética, sostenible, socialmente responsable y respetuosa con los derechos humanos y la diversidad, tanto en la práctica académica como en la profesional, y diseñar soluciones para mejorar estas prácticas.

Hay que considerar, pues, que aborda tres grandes dimensiones:

I. Sostenibilidad
II. Comportamiento ético y responsabilidad social (RS)
III. Diversidad (género, entre otros) y derechos humanos

Los Objetivos de Desarrollo Sostenible (ODS) **[7]**, también conocidos como Objetivos Globales, fueron adoptados por las Naciones Unidas en 2015 como un llamamiento universal para poner fin a la pobreza, proteger el planeta y garantizar que para el 2030 todas las personas disfruten de paz y prosperidad.

Las tres dimensiones comentadas anteriormente se alinean con los ODS (Objetivos de Desarrollo Sostenible 2030, ONU), con los que la UOC está públicamente comprometida. Esta alineación puede guiar el espíritu de la incorporación de la CCEG en general y también en este proyecto.

En este sentido procederemos a listar los posibles ODS que podríamos encuadrar en cada una de las tres grandes dimensiones.

Sostenibilidad
Dentro del ámbito de la sostenibilidad podemos encontrar las siguientes ODS.

- **ODS 9:** industria, innovación e infraestructura
  La inversión en infraestructura y la innovación son motores fundamentales del crecimiento y el desarrollo económico. Los avances tecnológicos son esenciales para encontrar soluciones permanentes a los desafíos económicos al igual que la oferta de nuevos empleos. En este caso, en los últimos años, ha habido un crecimiento de profesionales en el ámbito de la ciencia de datos que puedan ayudar en este caso a entidades financieras a detectar el blanqueo de capitales utilizando diferentes métodos cómo el Machine Learning. No solo eso, sino que, con la llegada de las criptomonedas, se han creado nuevos nichos y por lo tanto crece la oferta de cursos para aprender sobre la llamada tecnología Blockchain. Esto cómo consecuencia hará crecer el número de profesionales en el mundo laboral que puedan optar a empleos, fomentando así un desarrollo sostenible.

Comportamiento ético y responsabilidad social (RS)
Dentro del ámbito del comportamiento ético y responsabilidad social podemos encontrar la siguiente ODS.

- **ODS 16:** Paz, justicia e instituciones solidas

Sin paz, estabilidad, derechos humanos y gobernabilidad efectiva basada en el Estado de derecho, no es posible alcanzar el desarrollo sostenible. Vivimos en un mundo cada vez más dividido. Algunas regiones gozan de niveles permanentes de paz, seguridad y prosperidad, mientras que otras caen en ciclos aparentemente eternos de conflicto y violencia.
Es un hecho que el terrorismo se está financiando mediate el blanqueo de capitales y cómo consecuencia cada año estos actos de violencia se cobran muchas vidas

alrededor del mundo. Además del terrorismo hay otros actos delictivos cómo el tráfico de drogas, la prostitución o la trata de personas que ponen en riesgo la paz o la justicia en el mundo. Es por eso por lo que diversas organizaciones cómo FATF o OFAC en conjunto con los gobiernos tratan de erradicar o luchar contra estos creando leyes que las entidades financieras o bancarias deberán cumplir sino quieren ser sancionadas severamente.

Diversidad (género, entre otros) y derechos humanos

Dentro del ámbito de la diversidad (género, entre otros) y derechos humanos podemos encontrar la siguiente ODS.

- **ODS 10:** reducción de las desigualdades

La desigualdad de ingresos ha aumentado en casi todas partes en las últimas décadas, pero a diferentes velocidades. La más baja es en Europa y la más alta es en el Medio Oriente. La desigualad de ingresos es un problema mundial que requiere soluciones globales. Estas incluyen mejorar la regulación y el control de los mercados y las instituciones financieras y fomentar la asistencia para el desarrollo y la inversión extranjera directa para las regiones que más lo necesiten. En este sentido las instituciones financieras deben identificar no solo las transacciones potencialmente fraudulentas sino identificar los clientes que hacen dichas transacciones. Deben hacer la debida diligencia según las alertas que hayan sido generadas por el sistema. De esta manera se podrá colaborar con las desigualdades sancionando a aquellos clientes que quisieran cometer tales fraudes o acciones ilegales con el motivo de enriquecerse a costa de otros.

1.9. Código empleado

El código empleado y los resultados obtenidos se han publicado en el presente repositorio de la plataforma GitHub, la cual permite subir código y documentos:

https://github.com/jespinlo/EspinozaJuan_TFM/tree/main

Cabe destacar, que, debido a la enorme cantidad de datos, no ha sido posible subir el archivo csv ya que ocupa más de 25MB.

2. Estado del arte

2.1. Metodología

La metodología empleada para realizar el estado del arte del proyecto ha consistido en utilizar las principales palabras claves en motores de búsqueda como Google Scholar o Web of Science. Dichos buscadores permiten acceder a información de artículos científicos y documentos de una forma más directa y ordenada.
Los principales términos o palabras claves utilizados para la búsqueda de documentación relacionada con el presente proyecto han sido los siguientes:

- Money Laundering
- AML detection
- AML rule based algorithm
- IA in AML
- DM in AML
- AML solution

Utilizando los términos listados anteriormente se han consultado una serie de artículos científicos, los cuales han sido clasificados en cuatro áreas de interés relacionadas con el presente proyecto:

- Concepto del lavado del dinero
- Aplicación del Data Mining y Machine Learning en la detección AML
- Aplicación del Deep Learning en la detección AML
- Soluciones en la Industria AML

Cabe destacar, que, a pesar de haber consultado muchos documentos, no todos han sido relevantes para realizar el estado del arte, ya que muchos de los documentos están relacionados con las leyes o temas legislativos relacionados con el AML y por lo tanto no

entraremos en detalles ya que no es el objetivo principal de este proyecto. No obstante, sí que algunos han sido útiles y gracias a ello se ha podido profundizar en temas relacionados con el objetivo principal del proyecto que han permitido encontrar la información adecuada. No solo la parte técnica sino los conceptos básicos de lo que es el Money Laundering y cómo se lleva a cabo.

Por lo tanto, gracias a la metodología utilizada de búsqueda y clasificación de artículos científicos se ha conseguido profundizar sobre el estado actual de sistemas y tecnologías que se han empleado para solucionar problemas similares al planteados en este proyecto.

2.2. Síntesis del estado del arte

Tal y como se ha mencionado en el subapartado anterior, tras la búsqueda exhaustiva de información, se han definido cuatro áreas de interés. A continuación, se va a explicar detalladamente, para cada tema definido, cuáles son los principales trabajos y estudios que se han considerado que tienen alguna relación y que pueden servir de comparativa y ayuda para la realización de la tesis.

2.2.1. Concepto del blanqueo de capitales

El blanqueo de capital **[9]** es el proceso de crear la apariencia que grandes sumas de dinero obtenidas de delitos graves, como el narcotráfico, la actividad terrorista, fraude, robo o evasión de impuestos, son originadas mediante una fuente legítima. A través del lavado de dinero, el infractor transforma los productos monetarios derivados de actividades delictivas en fondos con una fuente aparentemente legal. El sistema que trabaja contra el lavado de dinero es el llamado Anti-Money Laundering (AML).

Se estima que cada año se lavan 800 billones y 2 trillones de dólares en fondos ilegales, lo que equivale entre el 2 y el 5% del PIB global. El lavado de dinero es un proceso dinámico y complejo que requiere de tres etapas:

1. **Colocación:** Es el movimiento de efectivo desde su origen. En ocasiones, la fuente puede disfrazarse fácilmente o tergiversado. A continuación, se pone en circulación. a través de instituciones financieras, casinos, tiendas y otros negocios, tanto locales como en el extranjero. La colocación se puede llevar a cabo a través de muchos procesos.

2. **Estratificación**: El propósito de esta etapa es hacer más difícil el detectar y descubrir el lavado de dinero. Esa etapa está destinada a dificultar el seguimiento de los ingresos ilegales para las fuerzas del orden.

3. **Integración**: Es la etapa en la que el dinero previamente ya lavado es integrado en la economía principalmente a través del sistema bancario, y, por lo tanto, ese dinero parece ser algún tipo de ganancias negocios normales legales. Esto es diferente a la estratificación, ya que en el proceso de integración se proporciona detección e identificación de fondos lavados a través de investigadores.

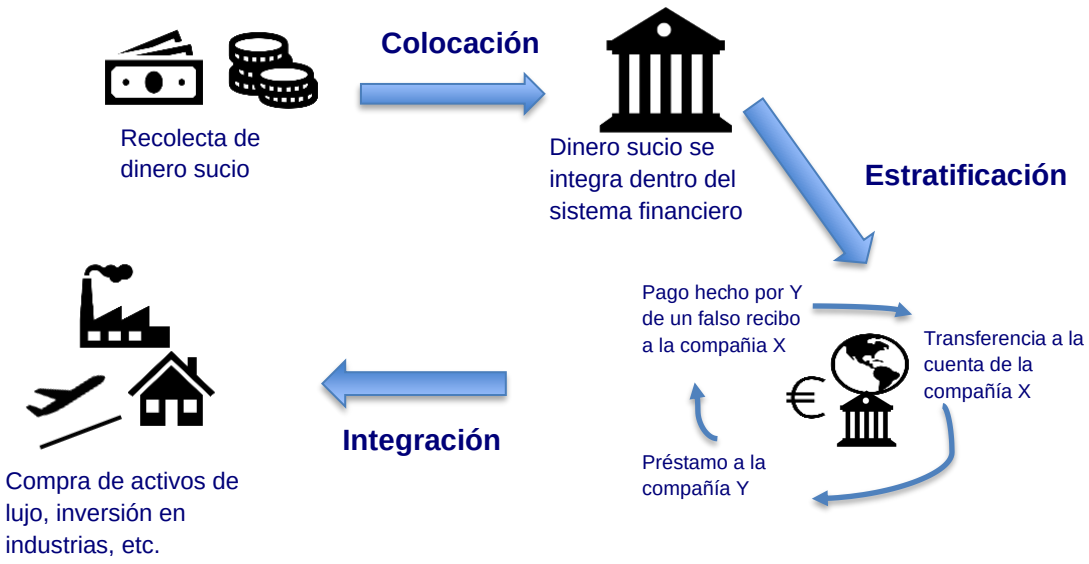


Figura 4: Ciclo del lavado de dinero

2.2.2. Problemas en las instituciones financieras

Las industrias bancarias y financieras buscan la manera de protegerse contra el fraude a todos los niveles y el blanqueo de capitales.

El blanqueo de capitales es una forma de delito económico. El objetivo es disfrazar los fondos obtenidos ilegalmente (dinero sucio) de tal manera que se dé la impresión de que provienen de una fuente de dinero acreditada.

Para ello los departamentos de Compliance tienen muy en cuenta la AML (Anti Money Laundering). AML se refiere a las actividades que las instituciones financieras realizan para

cumplir con los requisitos legales para monitorear activamente y reportar actividades sospechosas.

La legislación AML se está volviendo cada vez más estricta para los proveedores de servicios financieros. Debe evitarse que financien el blanqueo de capitales y/o el terrorismo siguiendo las pautas de la **OFAC (Office of Foreign Assests Control)** y las recomendaciones del **FATF (Financial Action Task Force).** Estos procedimientos se utilizan para identificar y bloquear las actividades fraudulentas en el sistema. Se implementa un algoritmo basado en reglas en el sistema para lanzar una alerta cada vez que se identifican transacciones fraudulentas. Estas alertas luego son investigadas por el analista y tomarán decisiones sobre si esta transacción es un fraude (Verdadero positivo) o no fraude (Falso positivo).

En el escenario actual, hay muchas alertas de falsos positivos dadas por el algoritmo y los investigadores cierran esas alertas dentro de un tiempo estipulado. Dado que hay una gran cantidad de alertas de falsos positivos, se invierte más tiempo en eliminar las alertas y las organizaciones requieren una gran cantidad de recursos humanos para investigar estas alertas. Por lo tanto, se desperdicia mucho tiempo y dinero solo porque el algoritmo basado en reglas no es lo suficientemente inteligente para identificar las transacciones no fraudulentas.

2.2.3. Caso de estudio en la AML

Supongamos que un algoritmo basado en reglas podría detectar una transacción de alto valor entre marido y mujer. Aquí, el algoritmo trata a las partes como cliente A y cliente B y no comprende el motivo subyacente de la transacción y la relación entre ellos. Además, el algoritmo basado en reglas es fijo y no aprende ni se adapta a las tendencias. Los infractores están tratando de evitar que sus transacciones sean detectadas por este algoritmo y mejoran su estrategia o maneras de actuar para evitar ser atrapados.

Por lo tanto, es necesario que el sistema también avance para aprender los patrones fraudulentos en las transacciones y activar solo las transacciones fraudulentas reales y evitar los falsos positivos. Esto se puede lograr entrenando un modelo de aprendizaje automático con los datos anteriores que contienen detalles de transacciones, ya sea que el algoritmo lo marque como fraude y el resultado del investigador en esas alertas marcadas (en realidad fraude o no).

2.2.4. AML vs Sanciones

Las sanciones y el cumplimiento AML son distintos, a menudo combinados, y frecuentemente convergente. Los requisitos AML varían según el país. A menudo surgen

malentendidos acerca de las expectativas regulatorias para sanciones y cumplimiento AML, por lo que los profesionales de cumplimiento deben educar con frecuencia a las partes interesadas internas sobre estas diferentes expectativas y requisitos.

En los Estados Unidos, los requisitos AML se aplican a las instituciones financieras cubiertas por el Bank Secrecy Act (BSA). Como tal, el universo de entidades sujetas a varias formas de requisitos AML bajo la BSA son órdenes de magnitud menor que el universo de personas estadounidenses que están obligadas a cumplir con las sanciones.

Los requisitos AML se centran principalmente en la recopilación de información, retención de registros e intercambio de información con el Treasury Department's Financial Crimes Enforcement Network (FinCEN) y otros organismos gubernamentales.

A diferencia de AML, las sanciones económicas se aplican a todas las personas estadounidenses y a algunas personas no estadounidenses. Las personas estadounidenses incluyen ciudadanos estadounidenses y residentes permanentes independientemente de dónde se encuentren, todas las entidades incorporadas en los EE. UU. y sus sucursales en el extranjero. Las personas no estadounidenses incluyen extranjeros subsidiarios de propiedad o controladas por empresas estadounidenses. Ciertos programas también requieren que las personas extranjeras en posesión de bienes de origen estadounidense cumplan con las normas establecidas.

2.2.5. Multas y sanciones en el siglo XXI

Para bien o para mal, gran parte del mundo mide el éxito o la falta de él en la lucha contra los delitos financieros a través de las multas o sanciones cada vez mayores impuestas a los bancos más grandes del mundo.

Si bien las multas y las sanciones continuarán y, sin duda, proliferarán aún más, si han sido efectivas y si es hora de pensar más profundamente y considerar nuevos enfoques es digno de un estudio y discusión más profunda. En general, cualquier multa para una institución financiera no solo supone un duro golpe económico, sino que además esta en juego la reputación que se verá duramente afectada mediante la perdida de confianza de los clientes.

En el siglo XXI, [26] se impusieron 96 multas en 17 países, a 57 bancos por 38,47 mil millones de dólares, siendo 21,47 mil millones para AML y 16,9 mil millones de dólares para Sanciones: un **promedio anual de 1,77 billones de dólares** (977 millones para AML y 800 millones para Sanciones). Las multas se pueden estructurar por áreas, de esa manera podemos agruparlas en las siguientes áreas dependiendo del delito o irregularidad que se haya cometido: ABC (Anti Bribery and Corruption), AML Non CBR (Non Correspondent Banking Relationship), AML CBR (Correspondent Banking Relationship), AML Fraud AML Markets. (Se excluye multas relacionadas a impuestos o abuso de mercados).

Correspondent Banking: se refiere a una institución financiera que brinda servicios a otra, generalmente en otro país (Correspondent Bank o Bancos corresponales). Actúa como intermediario o agente, facilitando las transferencias electrónicas, realizando transacciones comerciales, aceptando depósitos y recopilando documentos en nombre de otro banco. Los bancos corresponsales son utilizados por bancos nacionales para atender transacciones que se originan o se completan en países extranjeros. Los bancos nacionales suelen utilizar bancos corresponsales para obtener acceso a los mercados financieros extranjeros y atender a clientes internacionales sin tener que abrir sucursales en el extranjero.

Las multas AML impuestas más altas son:

1. Goldman Sachs 7.200 millones de dólares - Caso de esquema de soborno de Malasia
2. JP Morgan 2,600 millones de dólares – Caso Madoff
3. HSBC 1,600 millones de dolares – México/EE.UU. ML

Las multas más frecuentes incluyen:

- Deutsche Bank (7 multas)
- Goldman Sachs (5 multas)
- Credit Suisse (4 multas)
- J.P. Morgan (3 multas)
- HSBC (3 multas)

2.2.6. Aplicación del Data Mining y Machine Learning en el sistema AML

Los enfoques tradicionales de AML siguen un trabajo manual e intensivo ya que el lavado de dinero es un actividad sofisticada y variada pues se puede producir de muchas maneras. Estos enfoques pueden ser clasificados en: identificación, detección, evitación y vigilancia o investigación de actividades de lavado de dinero **[10].** Dado que el volumen de datos bancarios y datos transaccionales van incrementando de varias maneras y constantemente, estos enfoques deben ser apoyados por herramientas automatizadas para detectar patrones de lavado de dinero. Sin embargo, las tradicionales soluciones basadas en reglas tienen una serie de inconvenientes, tales como umbrales ineficaces, alto problema de falsos positivos, falta de reconocimiento de patrones y datos insuficientes para combatir el lavado de dinero.

La mayoría de los enfoques AML basados en la minería de datos (DM) intentan reconocer patrones de ML (Money Laundering) mediante diferentes técnicas, como máquina de vectores de soporte (SVM) **[11],** análisis de correlación **[12],** análisis de histogramas **[12],** etc. El objetivo es proporcionar técnicas para detectar una variedad de actividades de ML

explorando una dimensionalidad muy grande de los conjuntos de datos (clientes, cuentas, transacciones, geografía, tiempo, etc.)

En **[13]**, los autores proponen una extensión del vector de soporte máquina (SVM) **[14]** para detectar el comportamiento inusual del cliente. Presentan una combinación de un kernel RBF (Radial Basis Function) mejorado **[15]** con la definición de distancia distinta **[16]** y algoritmos SVM supervisados/no supervisados. SVM (Support Vector Machine) de una clase **[14]** es un aprendizaje no supervisado utilizado para detectar valores atípicos basados en un conjunto de datos de entrenamiento no etiquetado que son muy adecuados para los conjuntos de entrenamiento de ML. La ventaja de este enfoque es que puede tratar con conjuntos de datos heterogéneos.

En **[12]** aplicaron un proceso de discretización en sus conjuntos de datos para construir agrupaciones. Mapean su espacio de características "cliente x tiempo x transacción" a n+2 espacio de Euclides dimensional: n dimensiones del cliente, 1 dimensión de tiempo y 1 transacción dimensión. En primer lugar, se discretizan toda la línea de tiempo en diferentes instancias de tiempo. Por lo tanto, cada transacción es vista como un nodo en un espacio de línea de tiempo unidimensional. Proyectan todas las transacciones de clientes al eje de la línea de tiempo, acumulando transacciones y la frecuencia de transacciones para formar un histograma.

Luego crean grupos basados en segmentos en el histograma. Los análisis de correlación local y global son entonces aplicados para detectar patrones sospechosos. Este enfoque en primer lugar mejora la complejidad al reducir el problema de agrupamiento a un problema de segmentación **[16].**

Otro enfoque para agrupaciones es el Clustering. Esta técnica se utiliza normalmente para agrupar transacciones con cuentas bancarias en diferentes grupos que tienen muchas similitudes entre sí **[18].** Estas técnicas nos ayudan a detectar patrones de secuencia de transacciones sospechosas o presentar modelos para identificar las cuentas o los clientes de más riesgo para las entidades financieras. Uno de los desafíos más importantes que enfrenta el agrupamiento de transacciones es el tamaño y la cantidad de datos, pues estamos ante miles o millones de transacciones por unidad tiempo.

En **[20]** se han aplicado métodos de aprendizaje automático y minería de datos para discriminar transacciones fraudulentas y para predecir si nuevas transacciones lo son. Se predijo valores atípicos entrenando una SVM para identificar lo inusual, basado en las características que representan las actividades de los usuarios.

Las técnicas de Data Mining o Machine Learning son muy convenientes para detectar el lavado de dinero mediante la detección de patrones y comportamientos inusuales. Dado que los casos de lavado de dinero están cambiando y los delincuentes usan métodos más nuevos, las técnicas de minería de datos no supervisada serán más efectivas para detectar

nuevos patrones de lavado de dinero y puede ser crucial para mejorar el aprendizaje modelos basados en métodos de clasificación.

2.2.7. Aplicación del Deep Learning en el sistema AML

Cómo hemos visto en el anterior subapartado, la minería de datos es un proceso para extraer conocimiento de los datos. Es un proceso que hoy en día ya se utiliza como herramienta en la banca y las finanzas, en general, proporciona información útil de la operativa y datos históricos para permitir una mejor toma de decisiones. Podemos decir que es un campo interdisciplinario pues abarca grandes aspectos de estadística, base de datos, aprendizaje automático y visualización de los datos. Implica pasos que incluyen la selección de datos, integración de datos, transformación de datos, etc.

La detección y prevención del lavado de dinero es notoriamente difícil **[21], [22].** La naturaleza compleja de los productos, servicios y el lavado de dinero en sí mismo es dinámico y se adapta con el tiempo de acuerdo con condiciones cambiantes. Los patrones de comportamiento cambian y los delincuentes conocen las técnicas que se utilizan para combatirlos. Dado tal comportamiento, no hay reglas fijas que se puede aplicar para detectar patrones de lavado de dinero. Las reglas fijas pueden ser aplicadas para mitigar ciertos comportamientos extremos y para hacer cumplir unas normas definidas. Sin embargo, incrustar sistemas estáticos basados en reglas en entornos de transacciones electrónicas no proporciona garantías adecuadas para combatir el blanqueo de capitales o lavado de dinero. Por lo tanto, es vital abordar el problema utilizando una tecnología que se adapte, para que los sistemas puedan ser dinámicos en la forma de responder a los cambios en los patrones de lavado de dinero

A diferencia de los enfoques vistos anteriormente cómo Data Mining o Machine Learning, los métodos de Deep Learning puede aprender representaciones de características a partir de datos sin procesar. En las técnicas de Deep Learning, múltiples capas de representación aprenden a partir de una capa de entrada de datos sin procesar, mediante manipulaciones no lineales en cada nivel de aprendizaje. El Deep Learning ha superado a muchos enfoques convencionales de Machine Learning o Data Mining en funciones diseñadas en varias tareas de Inteligencia Artificial, incluida la comprensión del lenguaje natural, el reconocimiento de imágenes, y reconocimiento de voz **[17].** NLP (Natural Language Processing) es un subdominio de la inteligencia artificial y con frecuencia implica la comprensión y generación de lenguaje natural; más específicamente, emplea un conjunto de técnicas para la sintaxis, la semántica y el análisis del discurso, la extracción y clasificación de textos, la extracción de información y la traducción automática.

NPL y el Deep Learning ya están en uso en muchos niveles de regulación AML. Sin embargo, su aplicación se limita a técnicas de hace una década, y no logran igualar el ritmo actual de investigación.

En **[19]** se ha visto el uso de codificadores automáticos para detectar actividades sospechosas. Dado que hay una gran cantidad de datos disponibles en diferentes instituciones, las técnicas de Deep Learning pueden resultar útiles.

Otro tema importante que hemos encontrado en **[23]** es el llamado Entity Recognition. Es un conjunto de algoritmos capaces de reconocer entidades relevantes (por ejemplo, personas, puestos y empresas) mencionados en una cadena de texto de entrada. La extracción de relaciones detecta la relación entre dos nombres de entidades nombradas (e1, e2) en una oración dada, típicamente expresada como un triplete [e1, r, e2] (donde, r es una relación entre e1 y e2). Entidad de resolución (ER) determina si las referencias a entidades mencionadas en varios registros y documentos se refieren a la misma o diferente entidades. Por ejemplo, una misma persona puede ser mencionada de diferentes maneras y una organización podría tener diferentes direcciones.

Identificar inferencias a través de diferentes redes y relaciones semánticas entre entidades nombradas es aún más desafiante cuando crece la cantidad de datos.
ER puede reducir la complejidad de las tareas mediante la eliminación de duplicados y la vinculación de entidades. La complejidad de una red podría reducirse significativamente mediante la eliminación de duplicados.

La implementación del análisis de sentimientos **[23]** puede ser útil para el sistema AML; su papel principal es el de acortar el período de investigación de un oficial de cumplimiento. Se puede aplicar en diferentes niveles, como la incorporación de clientes o las etapas de seguimiento del perfil del cliente. El objetivo de un sistema de análisis de sentimiento en este contexto es monitorear las tendencias de sentimiento asociadas con un cliente para identificar patrones importantes.

Cuando los investigadores de AML identifican una empresa que potencialmente ha estado involucrada en una transacción sospechosa, generalmente consultan Internet en busca de pruebas. Analizar los niveles de sentimiento de los artículos em noticias sobre una organización específica puede revelar una gran cantidad de evidencia.

El análisis de sentimientos es un tema típico en NPL, y se ha empleado en muchas áreas diferentes **[24].** En términos de IA, numerosas técnicas diferentes se han utilizado para el análisis de sentimientos, incluidos SVM, campos aleatorios condicionales, redes neuronales profundas como CNN y RNN.

2.2.8. Soluciones en la industria AML

En la actualidad, el flujo de trabajo AML típico en la industria es un procedimiento lineal que conecta una fuente de datos a un sistema basado en reglas. Luego, los analistas o investigadores incorporan su propia investigación para determinar si las transacciones son legítimas o fraudulentas. Se sigue un proceso multietapa específico:

1. Los marcos AML recopilan y procesan datos.
2. Examinan y monitorean las transacciones. Si se determina que una transacción es sospechosa, se marca mediante flags, y un analista decidirá si la transacción marcada es fraudulenta o no.

Generalmente, los marcos AML se pueden descomponer en cuatro capas. La primera capa es la **capa de datos**, en la que se recopila, se gestiona y se almacenan los datos relevantes. Esto incluye tanto los datos internos de la institución financiera como los datos externos de fuentes tales como agencias reguladoras, autoridades y listas de vigilancia. La segunda capa es la **capa de detección y monitoreo**, la cual filtra las transacciones y los clientes potencialmente sospechosos. Esta capa ha sido automatizada por las instituciones financieras en un procedimiento de múltiples etapas, a menudo basado en reglas o análisis de riesgo. Si se detecta una actividad sospechosa, se pasará a la **capa de alerta** para una inspección adicional. Este proceso incluye el aumento de los datos con información histórica de transacciones y la evidencia necesaria para revisar la transacción marcada.

La recerca o búsqueda en el contenido web para adquirir información para la investigación están poco desarrollados en los sistemas AML actuales. En consecuencia, los analistas cuentan con recursos insuficientes, lo que aumenta la inexactitud de las decisiones tomadas por el analista y el tiempo requerido para inspeccionar cada transacción. La decisión de bloquear o aprobar una transacción la toma un analista humano en la **capa operativa**.

Hay una serie de softwares comerciales para AML **[25],** como NICE Actimize, SAS, MANTAS, Lexis-Nexis y Logica ISL, entre otros. Las funciones de monitoreo de actividades sospechosas en NICE Actimize tienen una extensa biblioteca que define varios modelos en diferentes circunstancias de AML, como banca o industrias de seguros. Además, también ofrece funciones avanzadas de análisis a través de la visualización de datos y exporta informes específicos.

SAS confronta AML y la regulación del financiamiento del terrorismo con enfoques basados en reglas y calificación de riesgo. La solución también registra actividades de comportamiento para una evaluación posterior.

En el pasado, la mayoría de las soluciones comerciales empleaban soluciones basadas en reglas para filtrar posibles transacciones sospechosas basadas en reglas y umbrales predefinidos. Se generarán alertas si se cumplen las reglas y los criterios de umbral. Cómo

hemos visto en los subapartados anteriores ha habido un desarrollo reciente en el uso de inteligencia artificial para combatir el lavado de dinero. Searchspace fue una de las primeras empresas desarrollar una solución AML sin usar reglas fijas, sino que se basó en estadísticas y técnicas de aprendizaje automático. El CEO de Searchspace describió cómo la solución comercial de Searchspaces emplea máquinas de vectores de soporte (SVM) con umbrales probabilísticos para detectar transacciones anómalas **[20].**

2.2.9. Conclusión del estado del arte

Tras realizar el estudio del arte consultando artículos científicos relacionados con las temáticas planteadas para el desarrollo del presente proyecto se han podido extraer las siguientes conclusiones:

- La mayoría de las entidades financieras ya utilizan sistemas para la detección y monitoreo de actividades sospechosas basadas en reglas predefinidas con unos umbrales predefinidos.

- Esta práctica se está quedando obsoleta ya que los delincuentes conocen las reglas que se están utilizando en los sistemas financieros y pueden fácilmente evitarlas de tal manera que no alcancen el umbral definido previamente. Por ejemplo, x transacciones hechas en las que cada transacción emite una cantidad pequeña de dinero de tal manera que la regla no se rompe y por lo tanto no se generará alerta.

- Hay un mercado en crecimiento en la industria AML. Numerosas empresas están dando soporte a entidades financieras vendiendo un software que ayudará a los bancos a detectar actividades sospechosas. Hemos visto que NICE Acimize es uno de ellos y desde mi experiencia como profesional he podido trabajar en algunas de las soluciones AML como es CDD (Customer Due Diligence), WLF (Watch List Filtering) o SAM (Suspicious Activity Monitoring).

- El Data Mining o el Machine Learning son campos en los que se pueden utilizar diversos algoritmos para ayudar a combatir el fraude o blanqueo de capitales. Por ejemplo, el Clustering podrían ayudarnos a clasificar clientes para poder hacer diversos tipos de perfiles previo al monitoreo de transacciones.

- Hay un campo todavía por explotar cómo es el Deep Learning o el análisis de sentimientos en los que se ayudará a los analistas o investigadores a ahorrarse bastante tiempo a la hora de decidir si una alerta generada es verdaderamente un caso fraudulento o un falso positivo.

Como conclusión general se puede extraer que los artículos consultados han ayudado a comprender mejor que tipo de algoritmos se están utilizando actualmente para el reconocimiento y clasificación de clientes o transacciones. Asimismo, cabe destacar que no ha existido ningún cambio sobre el planteamiento inicial, donde la idea sigue siendo clasificar transacciones hechas por clientes, para saber si son potencialmente fraudulentas o no.

3. Datos del proyecto

3.1. Conjunto de datos

Hay una falta de datos disponibles públicamente sobre servicios financieros y especialmente en el dominio emergente de transacciones de dinero. Parte del problema es la naturaleza intrínsecamente privada de las transacciones financieras, que conduce a que no haya conjuntos de datos disponibles públicamente.

Debido a la privacidad de los datos, es difícil encontrar datos de transacciones de carácter público. Por lo tanto, para llevar a cabo este proyecto usaremos datos simulados por **Paysim** [27] que simula transacciones de dinero basadas en una muestra de transacciones reales extraídas de un mes de registros financieros de un servicio de dinero móvil implementado en un país africano.

Los registros originales fueron proporcionados por una empresa multinacional, que es el proveedor del servicio financiero móvil que actualmente se ejecuta en más de 14 países de todo el mundo.

Utiliza datos agregados del conjunto de datos privados para generar un conjunto de datos sintético que se asemeja al funcionamiento normal de las transacciones e inyecta comportamiento malicioso para evaluar posteriormente el rendimiento de los métodos de detección de fraude.

¿Pero cómo se ha realizado la simulación de datos? El caso de estudio de simulación de pago de dinero móvil se basa en una empresa real que ha desarrollado una implementación de dinero móvil que proporciona a los usuarios de teléfonos móviles la capacidad de transferir dinero entre ellos utilizando el teléfono como una especie de billetera electrónica.

La tarea que se tiene entre manos es desarrollar un enfoque que detecte actividades sospechosas que sean indicativas de fraude. El desarrollo de PaySim abarca dos fases. Durante la primera fase, se modela e implementa un MABS (Multi Agent Based Simulation) que utilizó el esquema del servicio de dinero móvil real y generó datos sintéticos siguiendo escenarios que se basaron en predicciones de lo que podría ser posible cuando el sistema real comience a operar. Durante la segunda fase se obtuvo acceso a los registros financieros transaccionales del sistema y se desarrolló una nueva versión del simulador que utiliza datos transaccionales agregados para generar información financiera más parecida a la fuente original.

3.2. Descripción del dataset

En este apartado procederemos a explicar las columnas que tenemos en nuestro dataset. Hay un total de 11 columnas que hacen referencias a las 11 variables que tenemos. En este dataset tenemos tantas variables cualitativas cómo variables cuantitativas.

- **step**: mapea una unidad de tiempo en el mundo real. En este caso 1 paso es 1 hora de tiempo.
- **type**: CASH-IN, CASH-OUT, DEBIT, PAYMENT y TRANSFER.
- **amount**: cantidad de la transacción en moneda local.
- **nameOrig**: cliente que inició la transacción
- **oldbalanceOrg**: saldo inicial antes de la transacción
- **newbalanceOrig**: nuevo saldo después de la transacción
- **nameDest**: cliente que es el destinatario de la transacción
- **oldbalanceDest**: saldo inicial del destinatario antes de la transacción. Tenemos que tener cuenta que no hay información para clientes que comiencen con M (Merchants).
- **newbalanceDest**: saldo del destinatario después de la transacción. Debemos tener en cuenta que no hay información para clientes que comiencen con M (Merchants).
- **isFraud**: identifica una transacción fraudulenta (1) y una transacción no fraudulenta (0).
- **isFlaggedFraud**: identifica intentos ilegales de transferir más de 200.000 en una sola transacción.

3.3. Preprocesamiento de los datos

A continuación, vamos a cargar el fichero con el cual trabajaremos a lo largo de este proyecto. Para ellos utilizaremos la función read.csv y le pasaremos el nombre del archivo csv y el carácter de separación.

```
# Cargamos el fichero con el cual trabajaremos a lo largo de este proyecto
transacciones <- read.csv('bankTransaction_data.csv', sep=",")
```

Una vez hemos cargado el dataset y lo hemos guardado en una variable, en este caso nuestra variable es transacciones, utilizaremos la función ***str*** que devolverá la estructura de nuestro dataset mostrando las observaciones o filas y el número de columnas que nos indicará las variables del dataset.

```
str(transacciones)
```

```
## 'data.frame':    6362620 obs. of  11 variables:
##  $ step          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ type          : chr  "PAYMENT" "PAYMENT" "TRANSFER" "CASH_OUT" ...
##  $ amount        : num  9840 1864 181 181 11668 ...
##  $ nameOrig      : chr  "C1231006815" "C1666544295" "C1305486145"

"C840083671" ...
##  $ oldbalanceOrg : num  170136 21249 181 181 41554 ...
##  $ newbalanceOrig: num  160296 19385 0 0 29886 ...
##  $ nameDest      : chr  "M1979787155" "M2044282225" "C553264065"

"C38997010" ...
##  $ oldbalanceDest: num  0 0 0 21182 0 ...
##  $ newbalanceDest: num  0 0 0 0 0 ...
##  $ isFraud       : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ isFlaggedFraud: int  0 0 0 0 0 0 0 0 0 0 ...
```

En este caso tenemos un conjunto de datos con **6 millones de observaciones y 11 variables**. Además, podemos ver el tipo de variable que tenemos. En este caso tenemos **8 variables numéricas y 3 variables de tipo char.**

Otra función de R que nos puede ayudar a tener un overview de datos estadísticos del conjunto de datos es ***summary***. A continuación, mostramos la estadística descriptiva de nuestro conjunto de datos.

```
summary(transacciones)
```

```
##       step           type               amount           nameOrig
##  Min.   :  1.0   Length:6362620     Min.   :       0   Length:6362620
##  1st Qu.:156.0   Class :character   1st Qu.:   13390   Class :character
##  Median :239.0   Mode  :character   Median :   74872   Mode  :character
##  Mean   :243.4                      Mean   :  179862
##  3rd Qu.:335.0                      3rd Qu.:  208721
##  Max.   :743.0                      Max.   :92445517
##  oldbalanceOrg      newbalanceOrig       nameDest         oldbalanceDest
##  Min.   :       0   Min.   :       0   Length:6362620     Min.   :       0
```

```
##  1st Qu.:       0   1st Qu.:       0   Class :character   1st Qu.:        0
##  Median :   14208   Median :       0   Mode  :character   Median :   132706
##  Mean   :  833883   Mean   :  855114                      Mean   :  1100702
##  3rd Qu.:  107315   3rd Qu.:  144258                      3rd Qu.:   943037
##  Max.   :59585040   Max.   :49585040                      Max.   :356015889
##  newbalanceDest        isFraud         isFlaggedFraud
##  Min.   :        0   Min.   :0.000000   Min.   :0.0e+00
##  1st Qu.:        0   1st Qu.:0.000000   1st Qu.:0.0e+00
##  Median :   214661   Median :0.000000   Median :0.0e+00
##  Mean   :  1224996   Mean   :0.001291   Mean   :2.5e-06
##  3rd Qu.:  1111909   3rd Qu.:0.000000   3rd Qu.:0.0e+00
##  Max.   :356179279   Max.   :1.000000   Max.   :1.0e+00
```

3.3.1. Limpieza de los datos

Dado que la cantidad de datos es muy grande, limpiar el conjunto de datos de antemano es un requisito previo para la eficiencia, ya que la computadora tendrá que manejar una gran cantidad de datos. El siguiente paso tratará la limpieza de datos, mirando si hay valores vacíos o nulos en el conjunto de datos.

```
colSums(is.na(transacciones))

##           step           type         amount       nameOrig  oldbalanceOrg
##              0              0              0              0              0
## newbalanceOrig       nameDest oldbalanceDest newbalanceDest        isFraud
##              0              0              0              0              0
## isFlaggedFraud
##              0

colSums(transacciones=="")

##           step           type         amount       nameOrig  oldbalanceOrg
##              0              0              0              0              0
## newbalanceOrig       nameDest oldbalanceDest newbalanceDest        isFraud
##              0              0              0              0              0
## isFlaggedFraud
##              0
```

Podemos ver que no tenemos registros vacíos o nulos. Además, podemos observar que, además de las 6 características numéricas, tenemos 3 columnas de enteros (que son 0 o 1, es decir booleans) y 3 de caracteres. Nos centraremos en estos primeros.

De esta primera impresión podemos extraer lo siguiente:

- **isFlaggedFraud,** se parece a la predicción del algoritmo existente. No queremos que nuestro modelo dependa de la predicción de terceros, por lo que nos desharemos de él.
- **isFraud**, la variable objetivo, se convertirá en un factor y se le cambiará el nombre para facilitar la trazabilidad (los niveles de los factores se convertirán en etiquetas significativas).

- **nameDest y nameOrig**, parecen registros aleatorios y sin poder predictivo, pero los exploraremos a fondo en caso de que podamos extraer algún valor de ellos.

- **Type,** es una característica potencialmente importante. Podemos suponer fácilmente que el fraude es propenso a algunos tipos de transacciones. Por ejemplo, es bastante improbable que el ingreso en efectivo y el pago coexistan con un esquema de fraude. Esto también se explorará en detalle.

- **Step**, es solo la variable de tiempo. Probablemente no será de utilidad en nuestra predicción, ya que el fraude puede ocurrir en cualquier momento. Aun así, lo trataremos como predictivo hasta que se demuestre lo contrario.

Para empezar, echaremos un vistazo a la cantidad de valores únicos que contienen las columnas de caracteres. Esto nos ayudará a dar forma a nuestra estrategia de limpieza.

```
print("Numero de valor distintos en campos de tipos char")

## [1] "Numero de valor distintos en campos de tipos char"

transacciones[ , sapply(transacciones,is.character)] %>% sapply(n_distinct)

%>% print()

##     type nameOrig nameDest
##        5  6353307  2722362
```

nameOrig y nameDest (nombre de origen y nombre de destino respectivamente) contienen demasiados valores distintos para ser tratados como factores. Los números son obviamente aleatorios, probablemente generados por una función hash. Sin embargo, el prefijo puede simbolizar alguna información implícita (por ejemplo, el tipo de cuenta ya sea una cuenta de ahorros o corriente, etc.). Para probar eso, veremos la cantidad de prefijos distintos y su frecuencia.

```
print("Distintos prefijos en las columnas nameOrig y nameDest ")

## [1] "Distintos prefijos en las columnas nameOrig y nameDest "

transacciones %>% mutate(origin_name_prefix = str_sub(nameOrig,1,1),
```

```
#creamos columnas con los prefijos
              dest_name_prefix = str_sub(nameDest, 1, 1)) %>%
        select( c( origin_name_prefix,dest_name_prefix ) ) %>%
        table() %>% #count de las distinas ocurrences
        print()

##                   dest_name_prefix
## origin_name_prefix       C       M
##                  C 4211125 2151495
```

Parece que solo hay dos prefijos, C y M, casi igualmente presentes en la columna dest_name_prefix. Sin embargo, origin_name_prefix contiene solo C, por lo que podría no tener poder predictivo.

Como resultado, podemos extraer lo siguiente:

- añadir la columna de prefijo de destino
- omitir la creación de la entidad respectiva a partir del prefijo de origen
- deshacernos de las columnas iniciales
- deshacernos de la columna isFlaggedFraud

Además, convertiremos las columnas restantes de enteros y caracteres en factores.

Finalmente, para que nuestros gráficos sean más legibles, cambiaremos el nombre de las columnas de balance y el de la variable objetivo.

```
transacciones <- transacciones %>% mutate(dest_name_prefix =

str_sub(nameDest, 1, 1)) %>% #creamos columnas creando prefijos
                    select(-c('nameOrig','nameDest','isFlaggedFraud')) %>%

#eliminamos columnas deseadas
                    mutate_if(is.integer, as.factor) %>%

#convertimos variables integers en factors
                    mutate_if(is.character,as.factor) %>%

#convertimos variables chars en factores
                    rename("Old_Balance_of_Origin" = "oldbalanceOrg" ,

#renombramos a formato legible
                           "New_Balance_of_Origin" = "newbalanceOrig",
                           "Old_Balance_of_Destination" = "oldbalanceDest",
                           "New_Balance_of_Destination" = "newbalanceDest")

levels(transacciones$isFraud) <- c("No_Fraud", "Fraud" ) #formato legible
```

```
head(transacciones)

##   step     type   amount Old_Balance_of_Origin New_Balance_of_Origin
## 1    1  PAYMENT  9839.64                170136             160296.36
## 2    1  PAYMENT  1864.28                 21249              19384.72
## 3    1 TRANSFER   181.00                   181                  0.00
## 4    1 CASH_OUT   181.00                   181                  0.00
## 5    1  PAYMENT 11668.14                 41554              29885.86
## 6    1  PAYMENT  7817.71                 53860              46042.29
##   Old_Balance_of_Destination New_Balance_of_Destination  isFraud
## 1                          0                          0 No_Fraud
## 2                          0                          0 No_Fraud
## 3                          0                          0    Fraud
## 4                      21182                          0    Fraud
## 5                          0                          0 No_Fraud
## 6                          0                          0 No_Fraud
##   dest_name_prefix
## 1                M
## 2                M
## 3                C
## 4                C
## 5                M
## 6                M
```

4. Análisis gráfico

En este apartado mostraremos gráficas que nos dará información sobre el conjunto de datos que tenemos.

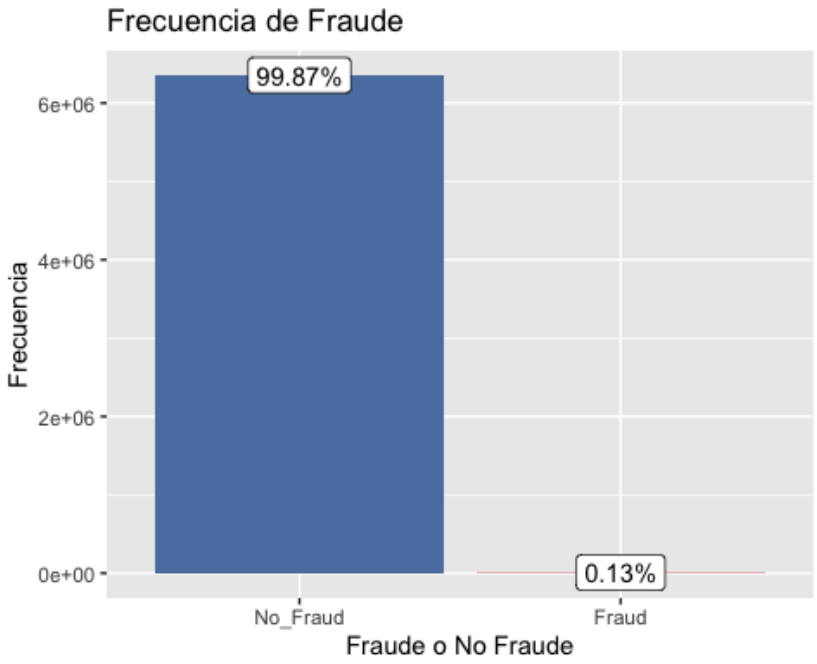
4.1. Variables cualitativas

4.1.1. Frecuencia de la variable objetivo

A continuación veremos la frecuencia y el porcentaje de cada tipo de variable objetivo (Fraude vs No Fraude).

```
transacciones %>% ggplot(aes(x = isFraud, y = (..count..))) +
    geom_bar(fill =  c( "#5b7fb0", "#c2695b" ) , stat = "count")+
    geom_label(stat='count',aes(label=  paste0(round(((..count..)/sum(..count..))
,4)*100 ,  "%" )))+ #añadimos el porcentaje a cada etiqueta
    labs(x = "Fraude o No Fraude", y = "Frecuencia",
```

```
    title = "Frecuencia de Fraude")
```



Como podemos ver tenemos un conjunto de datos no balanceado ya que el porcentaje de observaciones de fraude versus el de no fraude es muy grande.

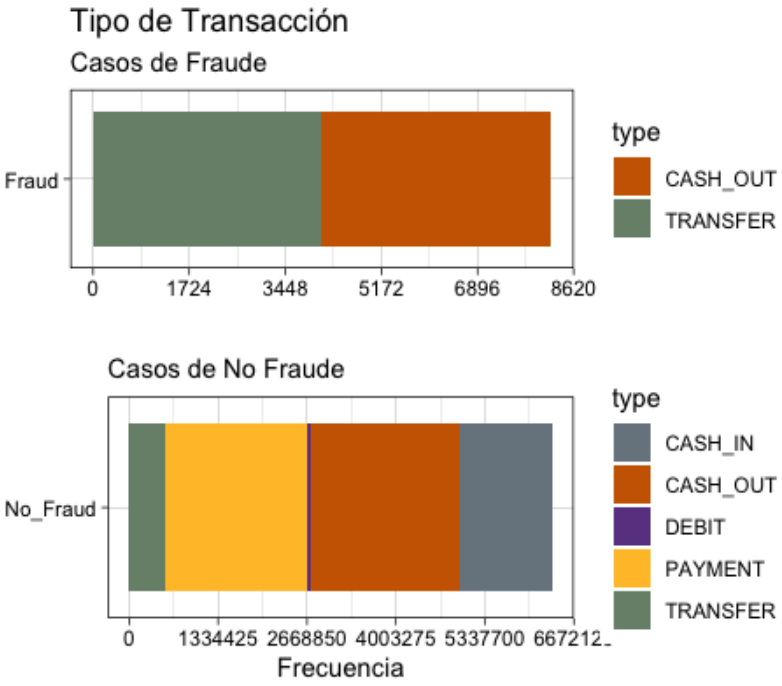
Este gran desequilibrio entre las dos clases objetivo conducirá a nuestro modelo a un sesgo hacia la variable dominante. Esto significa que, dado que más del 99% de los casos no son fraudes, nuestros algoritmos se sentirán cómodos con la predicción de “No Fraude”.
Más adelante, muestrearemos la clase mayoritaria para crear un conjunto de datos más balanceado.

4.1.2. Tipo de transacción

A continuación, exploraremos la presencia de cada tipo de transacción en los casos de fraude y no fraude.

```
#Creamos la paleta de colores
paleta<- c(CASH_IN = "#78858f", CASH_OUT = "#CC6600", DEBIT ="#6A4491",

PAYMENT = "#FFC233", TRANSFER = "#788f78")

#recuento de tipos de transacciones
```

```
p1 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = type ) ) +
    geom_bar()+
    scale_fill_manual(values=paleta) +
    scale_y_continuous(breaks =function(x){ seq(0,max(x), floor((max(x)/5)))} +
    labs(title = "Tipo de Transacción",subtitle ="Casos de Fraude",
    x = "", y = "" ) + coord_flip() + theme_linedraw()

p2 <-transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = type ) ) +
    geom_bar()+
    scale_fill_manual(values=paleta) +
    scale_y_continuous(breaks =function(x){seq(0,max(x), floor((max(x)/5)))})+
    labs(subtitle ="Casos de No Fraude", x = "", y = "Frecuencia")
    +coord_flip() + theme_linedraw()

grid.arrange(p1,p2,nrow = 2)
```

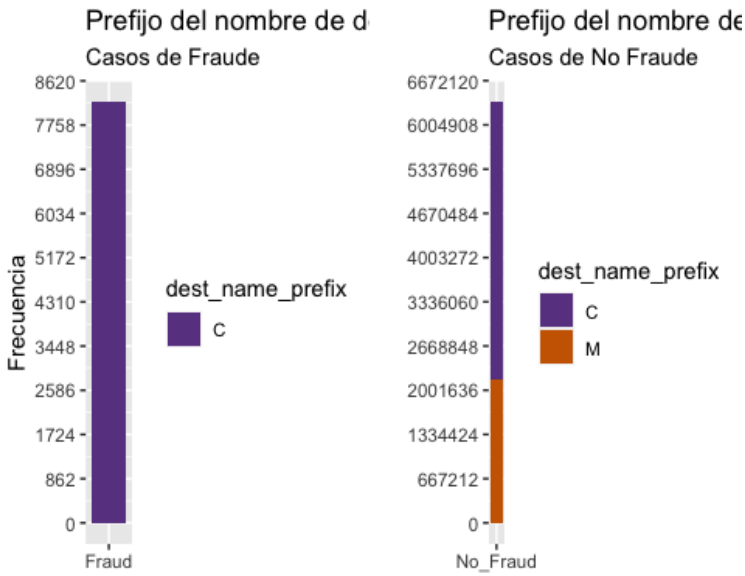


Como podemos ver, los registros de fraude coinciden con las transaccions de tipo Transfer y Cash Out. Eso significa que el resto de los tipos de transacciones (Cash In, Debit, Payment) nunca están asociados con el fraude. Por lo tanto, vamos a deshacernos de todas las observaciones con estas transacciones, ya que solo conducen a “No Fraude”.

4.1.3. Nombre del destinatario

Antes de eliminar estos tipos de transacciones, abordaremos de manera similar la columna dest_name_prefix para tener en cuenta efectos similares.

```
#creamos la paleta de colores asociado a cada tipo de destinatario
paleta_dest<- c("C" = "#6A4491", "M" = "#CC6600")

#recuentoswa de prefijo en casos de fraude

p3 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = dest_name_prefix ) ) +
    geom_bar()+scale_fill_manual(values=paleta_dest) +
    scale_y_continuous(breaks = function(x){seq(0,max(x), floor((max(x)/10)))})
   +labs(title="Prefijo del nombre de destino",
   subtitle = "Casos de Fraude",x = "", y = "Frecuencia" )

p4 <-transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = dest_name_prefix)) +geom_bar()+
    scale_fill_manual(values=paleta_dest) +
    scale_y_continuous(breaks = function(x){ seq(0,max(x),
    floor((max(x)/10)))})+labs(title = "Prefijo del nombre de destino",
    subtitle =  "Casos de No Fraude", x = "", y = "" )

 grid.arrange(p3,p4,ncol = 2)
```



A partir de la gráfica anterior, el fraude coincide únicamente con el prefijo “C” por lo tanto podríamos eliminar las observaciones que contengan la “M”. Después de la eliminación,

esta columna contiene solo “C” (es decir, varianza cero), lo que significa que podemos eliminarla.
Este proceso nos resultó útil no como una nueva columna que puede predecir el resultado, sino como una evidencia que nos llevó a descartar una cantidad significativa de filas que no nos aportan mucho. A continuación, procederemos a eliminar los registros los cuales hemos analizado y razonado el porqué de la eliminación anteriormente.

```
rows_before <- dim(transacciones)[1]
transacciones <- transacciones %>%
filter( (!(type %in% c("CASH_IN", "DEBIT", "PAYMENT") )) ,
#eliminamos estos tipos transacciones
                   dest_name_prefix == "C") %>%
#eliminamos los destinatarios con prefijo M
            select(-dest_name_prefix) #eliminamos la columna dest_name_prefix
```

A continuación, calcularemos el número de registros que se han eliminado y el porcentaje equivalente en el dataset.

```
rows_after <- dim(transacciones)[1]
diff <- rows_before - rows_after
diff_perc <- round(diff/rows_before * 100, 2)
sprintf("%i rows removed, equivalent to %s%% of the data",diff, diff_perc)
```

```
## [1] "3592211 rows removed, equivalent to 56.46% of the data"
```

Más de la mitad de los registros iniciales se han eliminado.

```
head(transacciones)
```

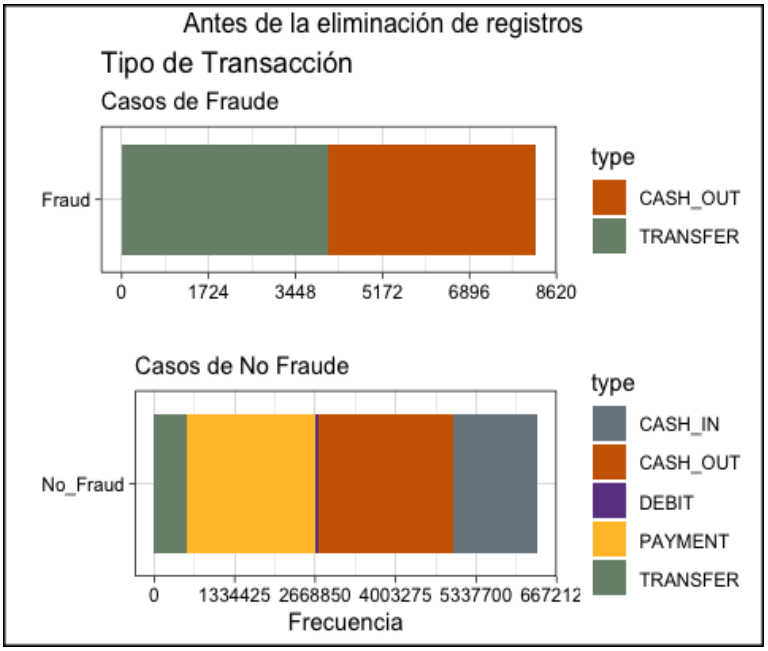
```
##   step     type   amount Old_Balance_of_Origin New_Balance_of_Origin
## 1    1 TRANSFER    181.0                181.00                      0
## 2    1 CASH_OUT    181.0                181.00                      0
## 3    1 CASH_OUT 229133.9              15325.00                      0
## 4    1 TRANSFER 215310.3                705.00                      0
## 5    1 TRANSFER 311685.9              10835.00                      0
## 6    1 CASH_OUT 110414.7              26845.41                      0
##   Old_Balance_of_Destination New_Balance_of_Destination  isFraud
## 1                          0                       0.00    Fraud
## 2                      21182                       0.00    Fraud
## 3                       5083                   51513.44 No_Fraud
## 4                      22425                       0.00 No_Fraud
## 5                       6267                 2719172.89 No_Fraud
## 6                     288800                    2415.16 No_Fraud
```

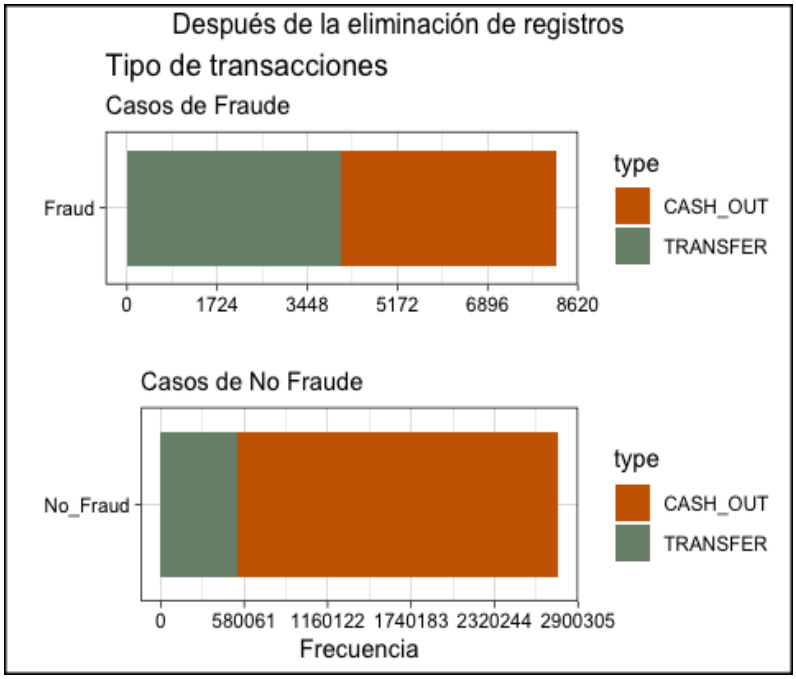
```
dim(transacciones)[1]
```

```
## [1] 2770409
```

Ahora vamos a visualizar el tipo de transacciones antes y despues de la eliminación de registros.

```
p11 <-  transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = type ) ) +
    geom_bar()+
    scale_fill_manual(values=paleta) +
    scale_y_continuous(breaks = function(x){ seq(0,max(x), floor((max(x)/5)))})+
    labs(title = "Tipo de transacciones", subtitle =  "Casos de Fraude",
    x = "", y        = "" ) + coord_flip() + theme_linedraw()

p12 <-transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = type ) ) +
    geom_bar()+
    scale_fill_manual(values=paleta) +
    scale_y_continuous(breaks = function(x){ seq(0,max(x), floor((max(x)/5)))})+
    labs(subtitle =  "Casos de No Fraude", x = "", y = "Frecuencia" )
    + coord_flip   () + theme_linedraw()
library(grid)
grid.arrange(p1,p2,top = "Antes de la eliminación de registros")
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))
```





Al parecer los casos de no fraude contienen más transacciones de retiros en efectivo, mientras que los casos de fraude las transferencias y los retiros en efectivos esta presentes casi por igual. Esto podría ser un indicio del poder predictivo de esta característica especifica.

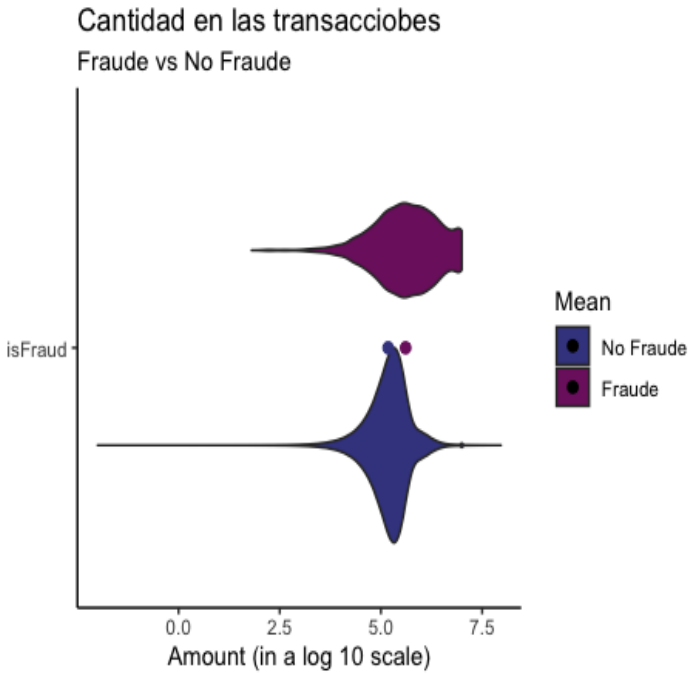
4.2. Variables cuantitativas

4.2.1. Amount

Dado que los valores numéricos van desde cantidades insignificantes hasta cantidades enormes, las graficaremos después de aplicarles la función log10. De esta manera, la distancia entre cantidades bajas y altas se minimiza y se vuelve más fácil para el ojo humano detectar patrones.

```
#Cantidad maxima en la variable Amount
max(transacciones$amount)

## [1] 92445517

#Cantidad minima en la variable Amount
min(transacciones$amount)

## [1] 0
```

```
transacciones %>%
    ggplot(aes(x = "isFraud", y = log(amount,10) , fill = isFraud)) +
    geom_violin() + coord_flip() +
    scale_fill_manual(values=c("#40448f", "#7d1f6d") , name="Mean",
    labels=c( "No Fraude","Fraude"))+
    labs(x = "", y = "Amount (in a log 10 scale)",
    title = "Cantidad en las transacciobes", subtitle = "Fraude vs No Fraude" ) +
    stat_summary(fun= "mean",geom = "point", size = 2,
    col =c("#40448f", "#7d1f6d"))+
    theme_classic()
```

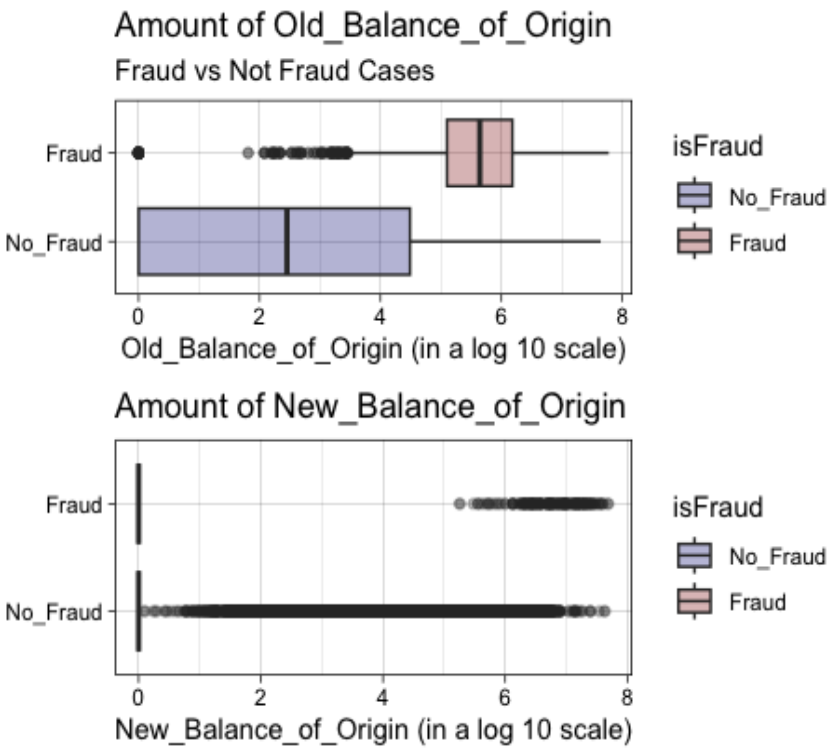


Podemos ver que, el monto de la transacción se ve diferente en los casos de fraude frente a no fraude. Las cantidades de No_fraud oscilan alrededor de 100000 (10^5), mientras que los casos de fraude se distribuyen de manera más uniforme. Esto podría ser una pista de que la columna Amount puede hacer una contribución en nuestro modelo.

4.2.2. Balance

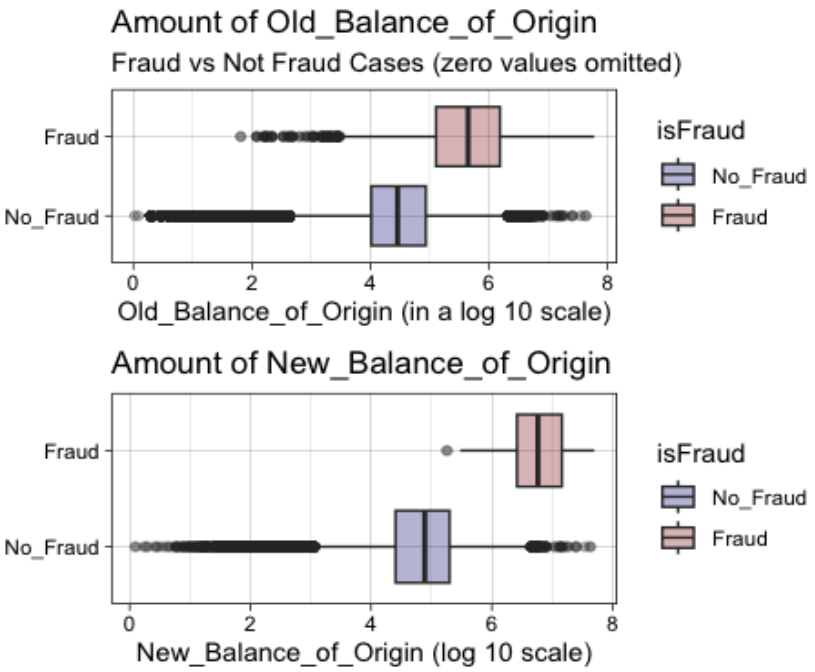
En contraste con la columna Amount, las columnas balance contienen valores con ceros. Los ceros devuelven un -Infinito cuando se les aplica la función logaritmo.
Una solución conveniente a este problema es agregar 1 en nuestro vector numérico antes de pasarlo a la función logarítmica. De esta manera, los 0 se convierten en 1 y producen un 0 después de aplicar la función. Esto sucede porque cualquier número elevado a la potencia de 0 devuelve 1.

```
which(transacciones %>% sapply(is.numeric))[-1] %>% names() -> labs

# store names for use in plot labels

p5 <- transacciones %>%
    ggplot(aes(x = isFraud, y = log(Old_Balance_of_Origin+1,10) ,
     fill = isFraud))    + geom_boxplot(alpha = .3) + coord_flip() +
     scale_fill_manual(values=c("navy", "darkred"))+
     labs(x = "", y = paste(labs[1], "(in a log 10 scale)"),
    title = paste("Amount of", labs[1] ), subtitle = "Fraud vs Not Fraud Cases")
    +theme_linedraw()


p6 <- transacciones %>%
    ggplot(aes(x = isFraud, y = log(New_Balance_of_Origin+1,10) ,
    fill = isFraud)) +
    geom_boxplot(alpha = .3) + coord_flip() +
    scale_fill_manual(values=c("navy", "darkred"))+
    labs(x = "", y = paste(labs[2], "(in a log 10 scale)"),
    title = paste("Amount of", labs[2] ) ) +
    theme_linedraw()

grid.arrange(p5,p6)
```



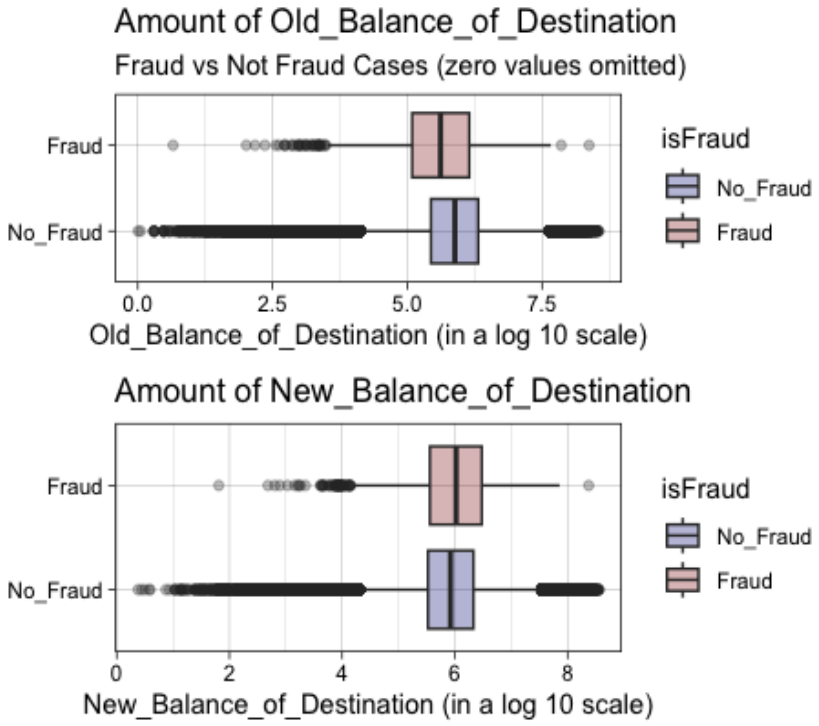
Observamos una concentración muy alta de valores 0 que hacen que nuestros gráficos no sean informativos. A continuación, trataremos de omitirlos.

```
p7 <- transacciones %>%  filter(Old_Balance_of_Origin != 0) %>%
    ggplot(aes(x = isFraud, y = log(Old_Balance_of_Origin+1,10) ,
    fill = isFraud)) +
    geom_boxplot(alpha = .3) + coord_flip() +
    scale_fill_manual(values=c("navy", "darkred"))+
    labs(x = "", y = paste(labs[1], "(in a log 10 scale)"),
    title = paste("Amount of", labs[1] ),
    subtitle = "Fraud vs Not Fraud Cases (zero values omitted)"  )+
    theme_linedraw()


p8 <- transacciones %>% filter(New_Balance_of_Origin != 0) %>%
    ggplot(aes(x = isFraud, y = log(New_Balance_of_Origin+1,10) ,
    fill = isFraud)) +
    geom_boxplot(alpha = .3) + coord_flip() +
    scale_fill_manual(values=c("navy", "darkred"))+
    labs(x = "", y = paste(labs[2], "(log 10 scale)"),
    title = paste("Amount of", labs[2] )  )+
    theme_linedraw()

grid.arrange(p7,p8)
```



Tanto en Old como en New Balance of Origin, los fraudes suelen ocurrir en mayor cantidad de balance. Esto también podría ser un signo de poder predictivo.
Aplicaremos el mismo procedimiento a Balance of destination.

```
p9 <- transacciones %>% filter(Old_Balance_of_Destination != 0) %>%
    ggplot(aes(x = isFraud, y = log(Old_Balance_of_Destination+1,10) ,
    fill = isFraud)) +
    geom_boxplot(alpha = .3) + coord_flip() +
    scale_fill_manual(values=c("navy", "darkred"))+
    labs(x = "", y = paste(labs[3], "(in a log 10 scale)"),
    title = paste("Amount of", labs[3] ),
    subtitle = "Fraud vs Not Fraud Cases (zero values omitted)"  )+
    theme_linedraw()

p10 <- transacciones %>% filter(New_Balance_of_Destination != 0) %>%
    ggplot(aes(x = isFraud, y = log(New_Balance_of_Destination+1,10) ,
    fill = isFraud)) +
    geom_boxplot(alpha = .3) + coord_flip() +
    scale_fill_manual(values=c("navy", "darkred"))+
    labs(x = "", y = paste(labs[4], "(in a log 10 scale)"),
    title = paste("Amount of", labs[4] ) )+
    theme_linedraw()

grid.arrange(p9,p10)
```

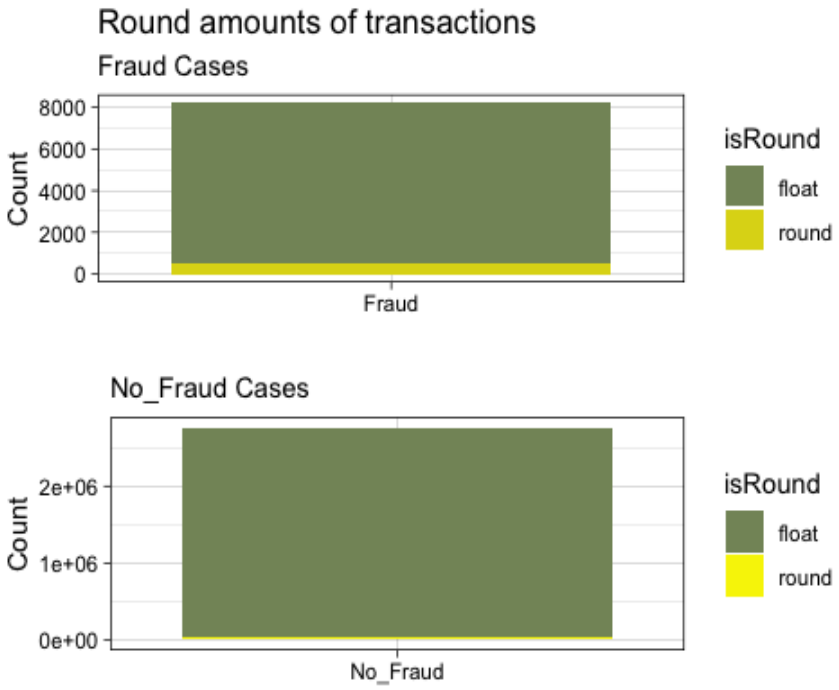


A diferencia de los saldos de origen, los saldos de destino no difieren mucho entre los casos de fraude y no fraude.

4.3. El efecto entero

Los estafadores pueden pensar que transferir una cantidad no redonda (por ejemplo, 12 325 en lugar de 10 000) parecerá una transacción más común y no despertará sospechas. Además, a la mente humana siempre se le ocurren números redondeados y ese sería el primer pensamiento de todos los estafadores (no tan profesionales). Por eso comprobaremos si los importes sin decimales en las transacciones coinciden con el fraude.

```
#Creamos una nueva columan isRound
transacciones <- mutate(transacciones, isRound = as.factor(ifelse( test =
(transacciones$amount - as.integer(transacciones$amount)) == 0 ,
yes = 'round', no = 'float' )))

p11 <- transacciones %>% filter( isFraud == "Fraud" ) %>% ggplot() +
    geom_bar(aes( x = isFraud , fill = isRound ) ) +
    scale_fill_manual(values = c( "#849468", "#ded718" )) +
    labs( title = "Round amounts of transactions" , x= "",
    y = "Count", subtitle = "Fraud Cases" )+
    theme_linedraw()

p12 <- transacciones %>% filter( isFraud == "No_Fraud" ) %>% ggplot() +
    geom_bar(aes( x = isFraud , fill = isRound ) ) +
    scale_fill_manual(values = c( "#849468", "#f7f300" ))+
    labs( x= "", y = "Count", subtitle = "No_Fraud Cases" ) +
    theme_linedraw()

grid.arrange(p11,p12)
```



Curiosamente, los números enteros parecen ser más frecuentes en los casos de fraude. Esa es probablemente una pista de una variable interesante para posteriores cálculos.

4.4. ¿Es el saldo realmente cero?

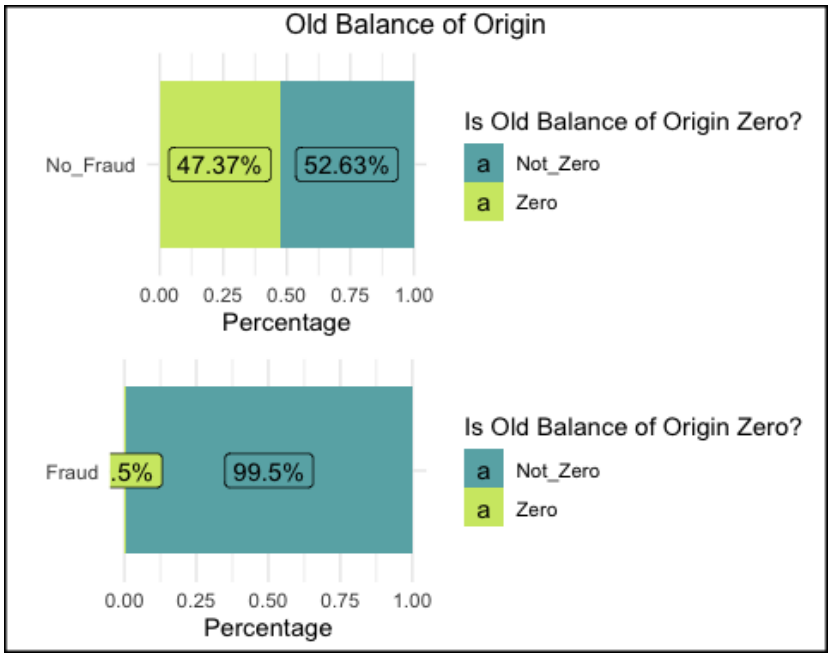
Otra suposición que se podría hacer es que los estafadores usan cuentas ficticias para enviar o (principalmente) recibir dinero. Es por esto que sería interesante explorar si las cuentas con saldos 0 están conectadas con el fraude. Para eso, crearemos un vector lógico para cada columna de Saldo que ya tenemos.

```
iszero <- transacciones %>% select(-amount) %>%

    #excluimos la cantidad que no puede ser 0
    select_if(is.numeric) %>% #select the remaining numerics - ie the balances
    mutate_all(R.utils::isZero) %>% #check whether they are zero
    rename_all(~paste0('isZero_', substr(., 1, nchar(.) - 4)))
    #add an isZero prefix on the names

transacciones <- transacciones %>% cbind(iszero) #merge with original df
```

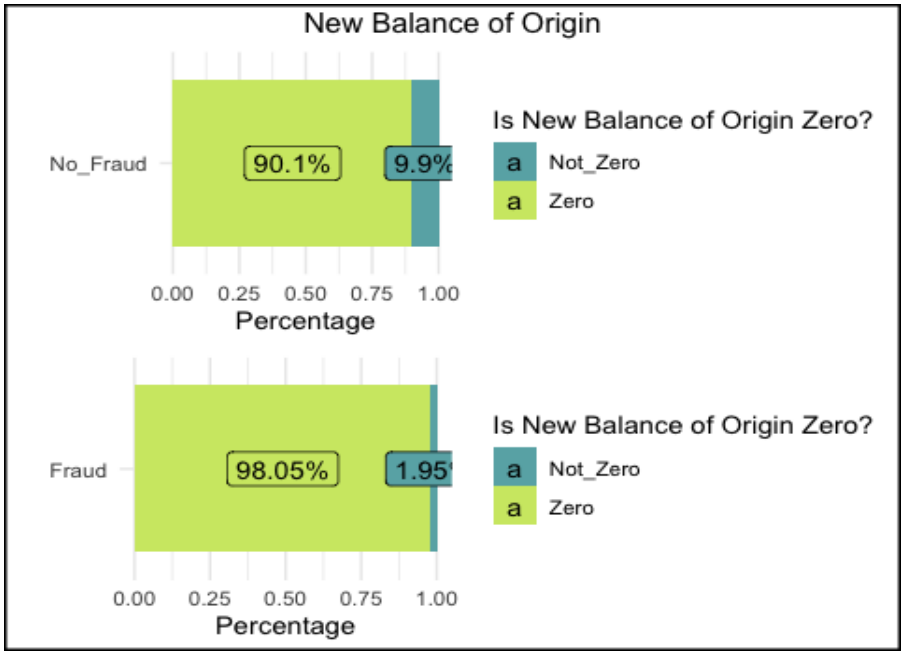
Primero, veremos las cuentas de origen.

```
p13 <- transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_Old_Balance_of_Or ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",aes(label=
paste0(round(((..count..)/sum(..count..)*100),2),"%") ), position =
position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"), #colors
                      labels = c("Not_Zero", "Zero"), #Legend labels
                      name = "Is Old Balance of Origin Zero?") + #Legend
title
    theme_minimal()+
    coord_flip()

p14 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_Old_Balance_of_Or ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",aes(label=
paste0(round(((..count..)/sum(..count..)*100),2),"%") ), position =
position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"), #colors
                      labels = c("Not_Zero", "Zero"), #Legend labels
                      name = "Is Old Balance of Origin Zero?")+ #Legend
title
    theme_minimal()+
    coord_flip()

p15 <- transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_New_Balance_of_Or ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",aes(label=
paste0(round(((..count..)/sum(..count..)*100),2),"%") ), position =
position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"),
                      labels = c("Not_Zero", "Zero"),
                      name = "Is New Balance of Origin Zero?") +
    theme_minimal() +
    coord_flip()

p16 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_New_Balance_of_Or ) )
```

```
geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",aes(label=
paste0(round(((..count..)/sum(..count..)*100),2),"%") ), position =
position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"),
                      labels = c("Not_Zero", "Zero"),
                      name = "Is New Balance of Origin Zero?")+
    theme_minimal() + coord_flip()
grid.arrange(p13,p14,nrow = 2, top = "Old Balance of Origin")
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))
```




```
grid.arrange(p15,p16,nrow = 2, top = "New Balance of Origin")
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))
```

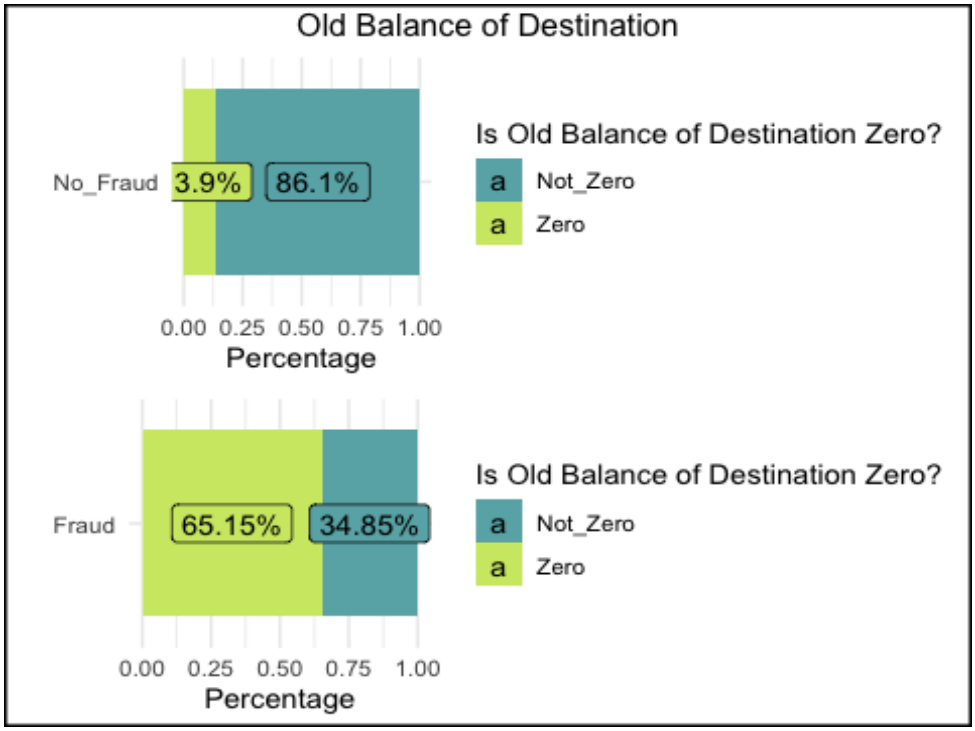


En el Old Balance of Origin, los casos de fraude muy rara vez contienen saldos cero. Eso es de esperar porque no tiene sentido que un estafador tenga un objetivo una cuenta con saldo cero.

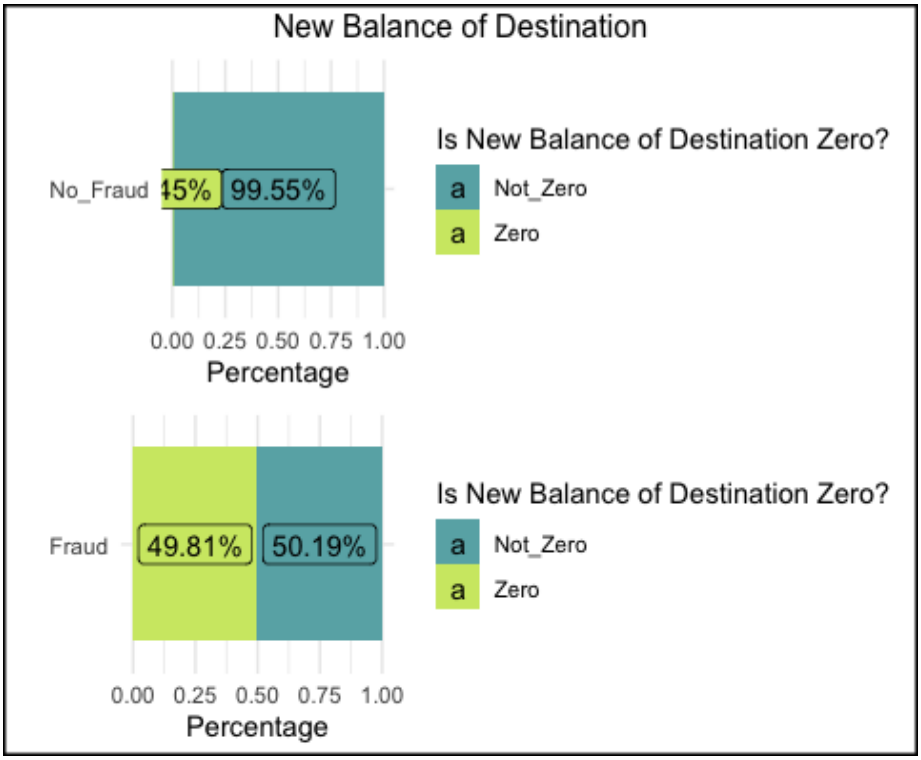
En New Balance of Origin en cambio, el 98% de los casos de fraude lo dejan con saldo cero (es decir, los estafadores saquean las cuentas y las dejan a cero).
Ahora veremos que comportamiento obtenemos con las cuentas de destino, “Balance of Destination”.

```
p17 <- transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_Old_Balance_of_Destina ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",aes(label=
    paste0(round(((..count..)/sum(..count..)*100),2),"%") ),
    position = position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"), #colors
                      labels = c("Not_Zero", "Zero"), #legend labels
                      name = "Is Old Balance of Destination Zero?") +
    #legend title
    theme_minimal()+
    coord_flip()

p18 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_Old_Balance_of_Destina ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",
```

```
    aes(label= paste0(round(((..count..)/sum(..count..)*100),2),"%") ),
     position = position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"), #colors
                      labels = c("Not_Zero", "Zero"), #legend labels
                      name = "Is Old Balance of Destination Zero?")+
    #legend title
    theme_minimal()+
    coord_flip()

p19 <- transacciones %>% filter(isFraud == "No_Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_New_Balance_of_Destina ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",
    aes(label= paste0(round(((..count..)/sum(..count..)*100),2),"%") ),
    position = position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"),
                      labels = c("Not_Zero", "Zero"),
                      name = "Is New Balance of Destination Zero?") +
    theme_minimal() +
    coord_flip()

p20 <- transacciones %>% filter(isFraud == "Fraud") %>%
    ggplot( aes( x = isFraud, fill = isZero_New_Balance_of_Destina ) ) +
    geom_bar(position = "fill") +
    labs(x = "", y = "Percentage") +
    geom_label(stat="count",
    aes(label= paste0(round(((..count..)/sum(..count..)*100),2),"%") ),
    position = position_fill(vjust=0.5)) +
    scale_fill_manual(values =  c("#69b0b3", "#d0e872"),
                      labels = c("Not_Zero", "Zero"),
                      name = "Is New Balance of Destination Zero?")+
    theme_minimal() + coord_flip()
grid.arrange(p17,p18,nrow = 2, top = "Old Balance of Destination")
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))
```




```
grid.arrange(p19,p20,nrow = 2, top = "New Balance of Destination")
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))
```



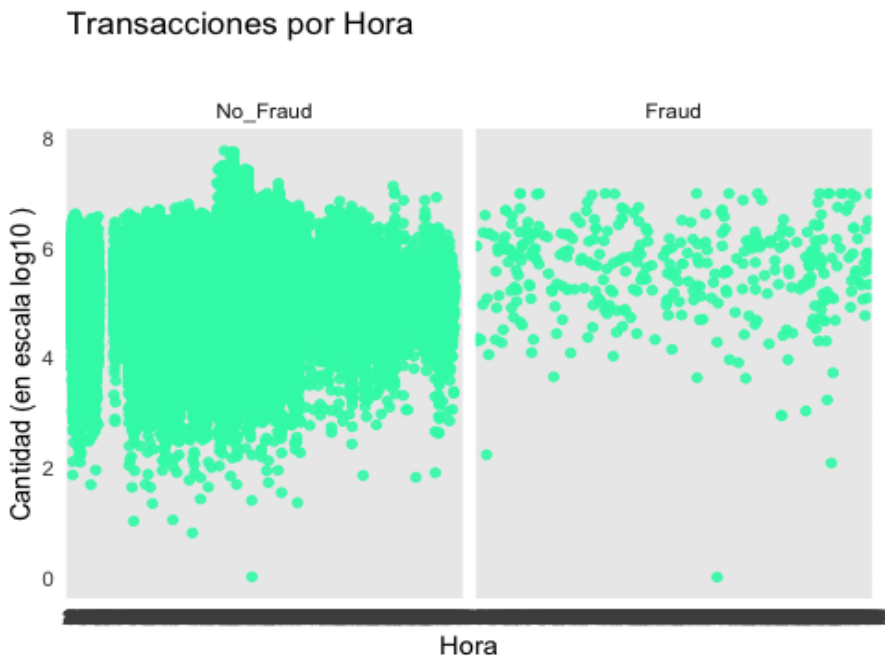
Tanto en Old como en New Balance of Destination, los casos de fraude coinciden con saldos cero con mucha más frecuencia que los casos de no_fraud. Esto puede explicarse fácilmente por el hecho de que los estafadores suelen enviar dinero a cuentas ficticias vacías.

4.5. Tiempo en el que se hacen las transacciones

Si bien puede parecer irracional que los fraudes ocurran en una determinada instancia de tiempo, podemos graficar la cantidad frente al tiempo coloreada por la columna isFraud en caso de que haya algún patrón obvio.

Debido a que las observaciones son tantas, se cubren unas a otras y nuestra trama deja de ser informativa. Para eso, muestrearemos aleatoriamente el 5% de nuestros datos. De esta forma reducimos los puntos graficados evitando de estas maneras posibles sesgos.

```
transacciones %>% sample_n(floor(nrow(transacciones)*.05)) %>%
    ggplot()+
    geom_jitter(aes( x = step, y = log(amount+1,10)), alpha = .9,
    col = "#2ff7b5")+
    facet_grid(~isFraud)+
    labs(x = "Hora", y = "Cantidad (en escala log10 )",
    title = "Transacciones por Hora", subtitle = "")+
    theme(axis.text.x = element_text(angle = 90)) +
    theme_minimal()
```

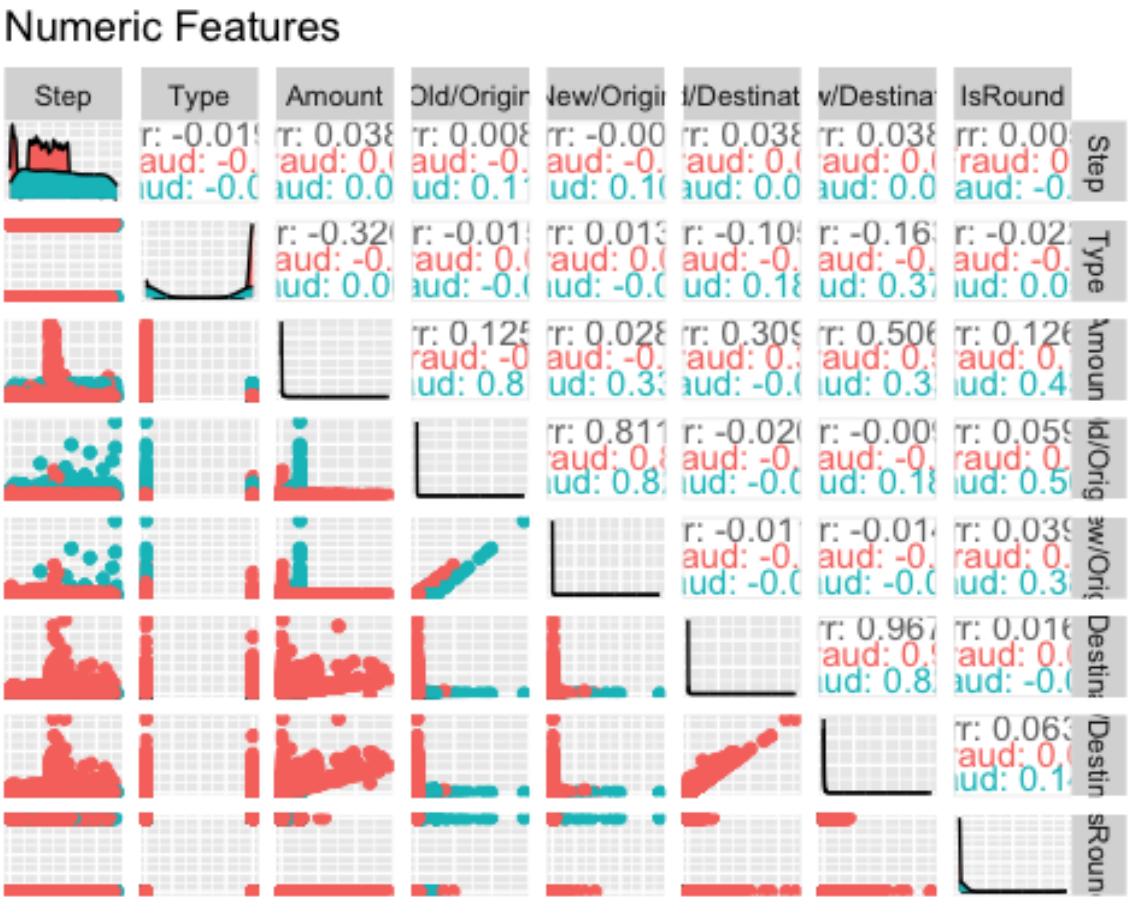


Los casos de fraude y no fraude se ven dispersos aleatoriamente a lo largo del tiempo y no parecen formar grupos. Sin embargo, hay algunas concentraciones leves de fraudes en algunas horas. Esto podría deberse al hecho de que cuando los estafadores logran cometer un fraude, intentan maximizar la cantidad de transacciones en el estrecho margen de tiempo que tienen.

4.6. Correlación

En este apartado verificaremos la correlación entre nuestras características numéricas con la ayuda de un diagrama de pares (ggpairs). Debido a que el diagrama de pares es muy costoso desde el punto de vista computacional, elegiremos el 5 % del conjunto de datos para el diagrama.

```
#Pasamos la variable step que es de tipo factor a numeric
transacciones$step <- as.numeric(as.character(transacciones$step))
#labels for plotting
labs <- c('Amount',"Old/Origin", "New/Origin",
"Old/Destination","New/Destination", "Step")
#Create a smaller dataset
transacciones_small <- transacciones %>% sample_n(floor(nrow(transacciones)*.05))
#Pairplot
transacciones_small[ ,sapply(transacciones,is.numeric)]  %>%
    ggpairs(mapping=ggplot2::aes(colour = transacciones_small$isFraud),
title = "Numeric Features",columnLabels = labs, axisLabels = "none")
```



Del diagrama podemos ver que hay dos correlaciones muy altas se encuentran dentro de nuestros datos numéricos. Parece que old origin y new origin se correlacionan ya que tiene un factor de correlación alto 0.81 y old y new destination con 0.96 de factor de correlación son las variables mas correlacionadas. Para decidir qué columnas conservar, utilizaremos la función findCorrelation del paquete **caret**.

Además, notamos una leve concentración de casos de fraude con valores cercanos a cero para algunas de las columnas de balance. Esto fortalece nuestra suposición de que los saldos cero podrían coincidir con el fraude.

```
high_cor_feats <- findCorrelation(cor(transacciones[ ,sapply(transacciones,
is.numeric)]), cutoff = .8, verbose = TRUE,
                                  names = TRUE, exact = TRUE)
## Compare row 7  and column  6 with corr  0.97
##   Means:  0.249 vs 0.126 so flagging column 7
## All correlations <= 0.8
high_cor_feats
## [1] "New_Balance_of_Destination"
```

Por lo tanto procederemos a eliminar esa variable la cual esta altamente correlacionada.

```
#Eliminamos las variables altamente correlacionadas
transacciones <- transacciones %>% select( -high_cor_feats )
```

4.7. Proceso PCA/SVD

Tanto el análisis de componentes principales, principal componente analysis (PCA) en inglés, como la descomposición de valores singulares, singular value decomposition (SVD) en inglés, son técnicas que nos permiten trabajar con nuevas características llamadas componentes, que ciertamente son independientes entre sí.

En realidad, estas dos técnicas nos permiten representar el juego de datos en un nuevo sistema de coordenadas que llamamos componentes principales. Este sistema está mejor adaptado a la distribución del juego de datos, por lo que recoge mejor su variabilidad.

A continuación vamos a normalizar las demás columnas para asegurarnos de que cada variable contribuye por igual a nuestro análisis.

```
# Definimos la función de normalización
 nor <-function(x) { (x -min(x))/(max(x)-min(x))}
# Guardamos el nuevo dataset normalizado
r = c(1,2,3,4,5,6,9,10,11,12)
transacciones_pca<- transacciones %>% select(all_of(r))
transacciones_pca_NOR <- as.data.frame(lapply(transacciones_pca, nor))
```

Aplicamos el análisis de componentes principales al row data set por eso empezamos ejecutando la función prcomp().

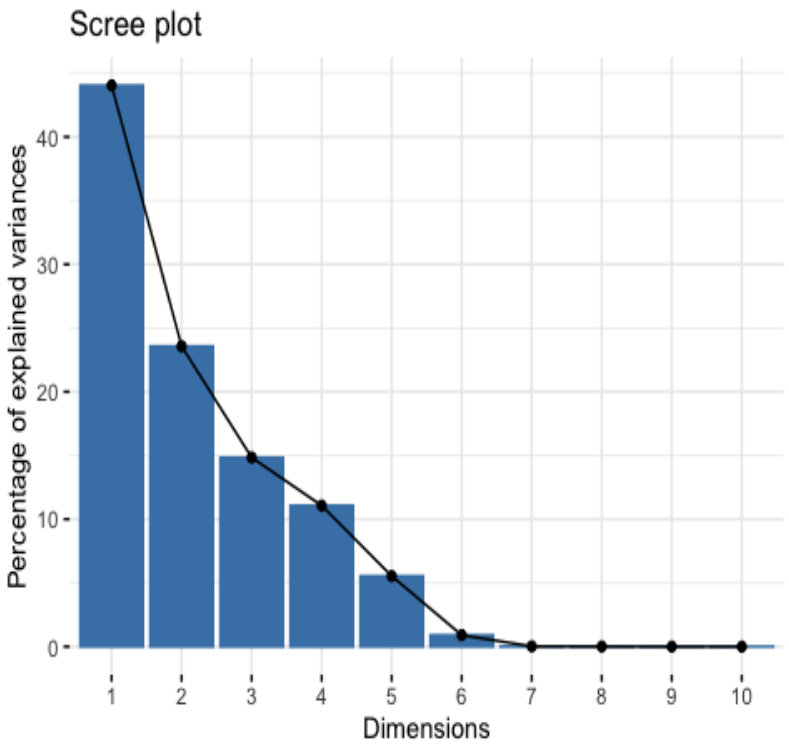
```
set.seed(123)
pca.transacciones <- prcomp(transacciones_pca_NOR)
summary(pca.transacciones)

## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     0.5379 0.3936 0.3123 0.2697 0.19077 0.07693 0.01222
## Proportion of Variance 0.4404 0.2357 0.1484 0.1107 0.05539 0.00901 0.00023
## Cumulative Proportion  0.4404 0.6762 0.8246 0.9352 0.99063 0.99963 0.99986
##                            PC8     PC9     PC10
## Standard deviation     0.008244 0.00460 0.001582
## Proportion of Variance 0.000100 0.00003 0.000000
## Cumulative Proportion  0.999960 1.00000 1.000000
```

Como puede observarse la función ***summary***, nos devuelve la proporción de varianza aplicada al conjunto total de cada atributo. Gracias a ello, el atributo 1 explica el 0.44 de variabilidad del total de datos; en cambio, el atributo 10 explica el 0.00000.

A continuación, se muestra un histograma para ver el peso de cada atributo sobre el conjunto total de datos:

```
library('factoextra')
#Los valores propios corresponden a la cantidad de variación explicada por cada c
omponente principal (PC).
ev= get_eig(pca.transacciones)
```

```
fviz_eig(pca.transacciones)
```



En este ejercicio se decidió utilizar el método de Kaiser para decidir cuál de las variables obtenidas serán escogidas. Este criterio mantendrá todas aquellas variables cuya varianza sea superior a 1.
Por tanto, el siguiente paso es escalar los datos y volver a calcular la varianza para ver qué datos seleccionamos.

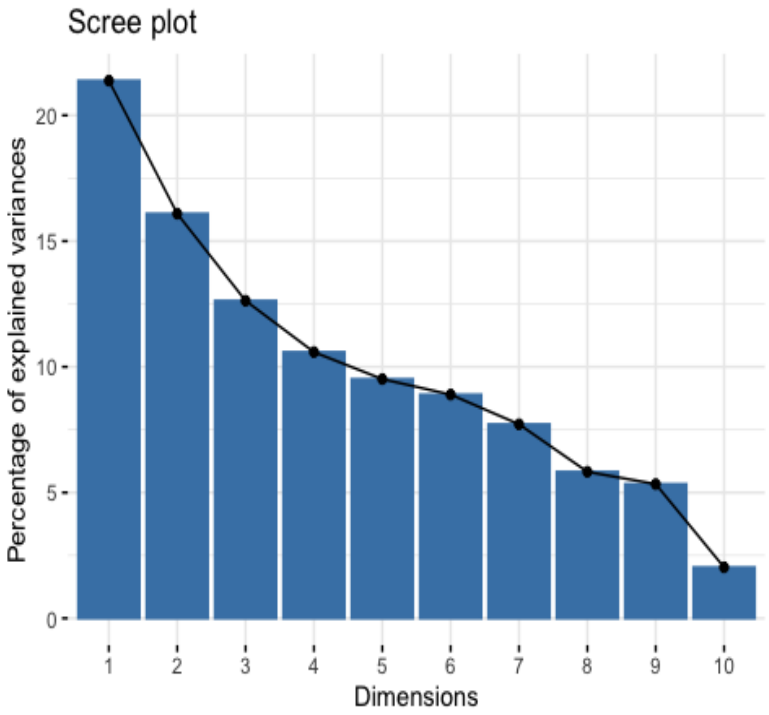
```
# Escalamos los datos
trans_scale <- scale(transacciones_pca_NOR)
# Calculamos las componentes principales
pca.trans_scale <- prcomp(trans_scale)
```

```
# Mostramos la varianza de dichas variables:
var_trans_scale <- pca.trans_scale$sdev^2
head(var_trans_scale)

## [1] 2.1381114 1.6090761 1.2628086 1.0586373 0.9512237 0.8896291
```

Después de analizar la varianza y aplicando el criterio de Káiser nos quedaremos con las principales componentes 1,2, 3 y 4 que son las superiores a 1. Este criterio tiene el problema de sobreestimar el número de factores, pero a pesar de ello es lo que aplicaremos para analizar los resultados.

Mostramos el histograma de porcentaje de varianza contado con los datos escalados:

```
set.seed(123)
fviz_eig(pca.trans_scale)
```



La suma de todos los valores propios da una varianza total de 10.

La proporción de variación explicada por cada valor propio se da en la segunda columna. Por ejemplo, 19.4 dividido por 10 es igual a 0,194, o alrededor del 19.4% de la variación se explica por ese primer valor propio. El porcentaje acumulado explicado se obtiene sumando las sucesivas proporciones de variación explicadas para obtener el total acumulado. Por ejemplo, el 19.4% más el 14.7% equivalen al 34%, y así sucesivamente. Por tanto, alrededor del 34% de la variación se explica por los dos primeros valores propios juntos.

Los valores propios se pueden utilizar para determinar el número de componentes principales a retener después de la PCA (Kaiser 1961):

- Un valor propio > 1 indica que los PCs representan más varianza de la que representa una de las variables originales de los datos estandarizados. Esto se utiliza habitualmente como punto de corte para el que se conservan los PCs. Esto sólo es cierto cuando los datos están estandarizados.

- También podemos limitar el número de componentes a ese número que representa una determinada fracción de la varianza total. Por ejemplo, si estamos satisfecho con el 70% de la varianza total explicada, utilizamos el número de componentes para conseguirlo que son las 4 componentes principales visto antes.

Continuamos con el análisis de los componentes principales. Después de aplicar el método Káiser se han seleccionado los 4 componentes principales.

Los componentes de get_pca_var() se pueden utilizar en el diagrama de variables de la siguiente manera:

- **var$coord**: coordenadas de variables para crear un diagrama de racimo
- **var$cos2**: representa la calidad de representación de las variables en el mapa de factores. Se calcula como las coordenadas al cuadrado: var.cos2 = var.coord * var.coord.
- **var$contrib**: contiene las contribuciones (en porcentaje) de las variables a los componentes principales. La contribución de una variable (var) a un determinado componente principal es (en porcentaje): (var.cos2 * 100) / (cos2 total del componente).

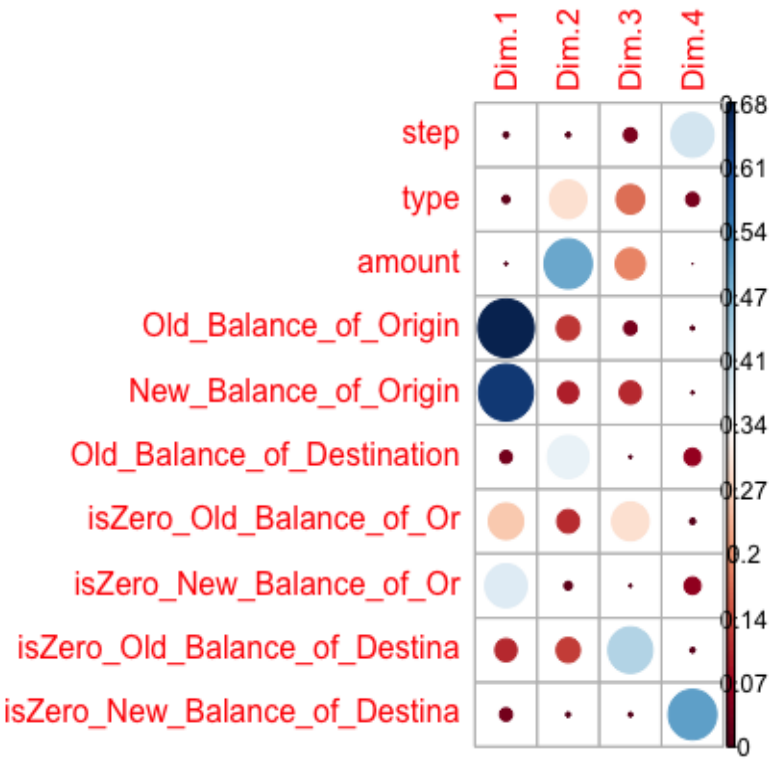
```
#Utilizamos las 4 componentes principales utilizadas anteriormente
head(var$coord[,1:4],10)
```

```
##                                  Dim.1       Dim.2       Dim.3        Dim.4
## step                          0.005155226  0.06310478  0.19487280 -0.624902576
## type                         -0.108434361 -0.54017638 -0.41388897 -0.192517136
## amount                        0.029537393  0.70204806  0.43682408  0.005833958
## Old_Balance_of_Origin        -0.822724354  0.34337996 -0.18912016  0.045831987
## New_Balance_of_Origin        -0.801373776  0.30462993 -0.32092722 -0.032380251
## Old_Balance_of_Destination    0.176663656  0.60247168  0.03240367 -0.241873364
## isZero_Old_Balance_of_Or      0.510910967  0.32817457 -0.54227676  0.079056526
## isZero_New_Balance_of_Or      0.614840804  0.11599466  0.03015624  0.239001559
## isZero_Old_Balance_of_Destina -0.320549707 -0.34968587  0.65196481  0.066526344
## isZero_New_Balance_of_Destina -0.182640698  0.05759375  0.05358623  0.708228757
```

4.7.1. Calidad de representación

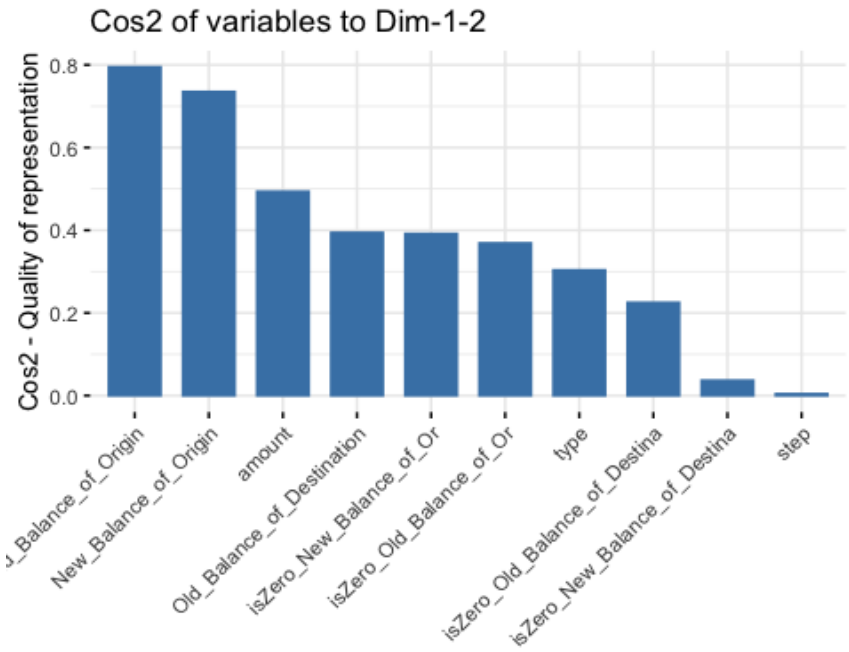
La calidad de representación de las variables en el mapa de factores se llama cos2 (coseno cuadrado, coordenadas cuadradas).

```
library(corrplot)

## corrplot 0.90 loaded

corrplot(var$cos2[,1:4], is.corr=FALSE)
```



También es posible crear un diagrama de barras de variables cos2 mediante la función fviz_cos2():

```
fviz_cos2(pca.trans_scale, choice = "var", axes = 1:2)
```



- Un cos2 elevado indica una buena representación de la variable en el componente principal. En ese caso, la variable se coloca cerca de la circunferencia del círculo de correlación.
- Un cos2 bajo indica que la variable no está perfectamente representada por los PC. En este caso, la variable se encuentra cerca del centro del círculo.

Para una variable dada, la suma del cos2 de todos los componentes principales es igual a uno. Si una variable está perfectamente representada por sólo dos componentes principales (Dim.1 y Dim.2), la suma del cos2 en estos dos PC es igual a uno. En ese caso las variables se colocarán en el círculo de correlaciones.

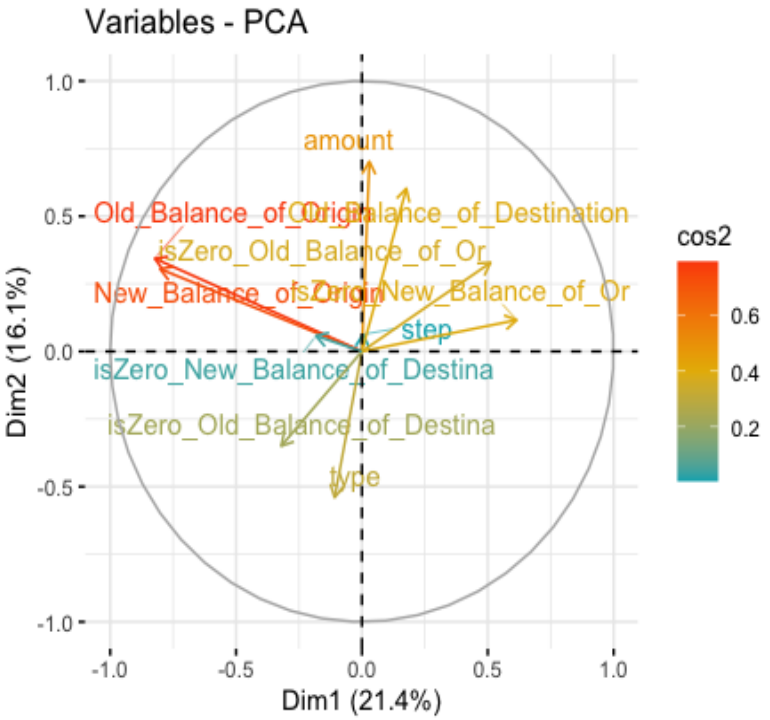
Para algunas variables, pueden ser necesarios más de 2 componentes para representar perfectamente los datos. En este caso, las variables se sitúan dentro del círculo de correlaciones.

En resumen:

- Los valores de cos2 se utilizan para estimar la calidad de la representación
- Cuanto más cercana esté una variable en el círculo de correlaciones, mejor será su representación en el mapa de factores (y más importante es interpretar estos componentes)
- Las variables que están cercanas al centro de la trama son menos importantes para los primeros componentes.

```
fviz_pca_var(pca.trans_scale,
             col.var = "cos2",
```

```
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE
              )
```



4.7.2. Contribución

Las contribuciones de las variables en la contabilización de la variabilidad de un determinado componente principal se expresan en porcentaje. Las variables que están correlacionadas con PC1 (es decir, Dim.1) y PC2 (es decir, Dim.2) son las más importantes para explicar la variabilidad en el conjunto de datos.

Las variables que no están correlacionadas con ningún PC o con las últimas dimensiones son variables con una contribución baja y pueden eliminarse para simplificar el análisis global. La contribución de las variables puede extraerse de la siguiente manera:

```
head(var$contrib[,1:4],13)
```

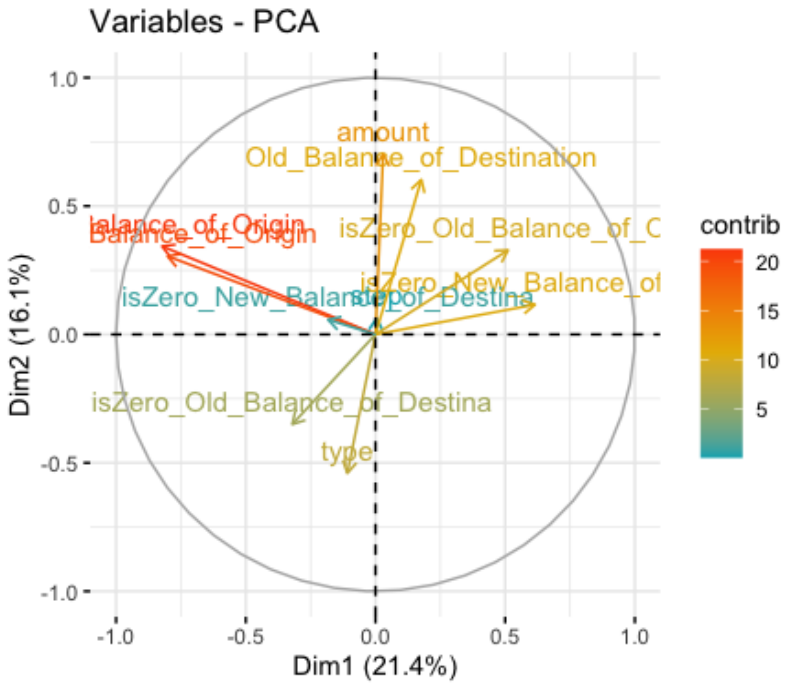
```
##                                 Dim.1      Dim.2       Dim.3        Dim.4
## step                       0.001242983  0.2474845  3.00721827 36.887348636
## type                       0.549925080 18.1340411 13.56532465  3.500995842
## amount                     0.040805058 30.6307132 15.11038805  0.003214989
## Old_Balance_of_Origin     31.657629129  7.3277951  2.83229282  0.198422167
## New_Balance_of_Origin     30.035850454  5.7672473  8.15596937  0.099040594
## Old_Balance_of_Destination 1.459701680 22.5577972  0.08314783  5.526229308
```

```
## isZero_Old_Balance_of_Or      12.208438826  6.6931918 23.28651321  0.590375410
## isZero_New_Balance_of_Or      17.680520427  0.8361793  0.07201397  5.395780568
## isZero_Old_Balance_of_Destina  4.805741967  7.5994050 33.65974309  0.418061456
## isZero_New_Balance_of_Destina  1.560144397  0.2061456  0.22738874 47.380531031
```

Cuanto mayor sea el vale de la contribución, mayor contribución habrá en el componente.

Las variables más importantes (o que contribuyen) pueden resaltarse en la gráfica de correlación de la siguiente manera:

```
fviz_pca_var(pca.trans_scale, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )
```



Las variables correlacionadas positivas apuntan al mismo lado de la trama. Las variables correlacionadas negativas apuntan a lados opuestos del gráfico. Por ejemplo, vemos que que **New_Balance_of_Origin y type** apuntan a direcciones opuestas por tanto no están nada correlaciones, además lo hemos visto antes ya que tienen un coeficiente de correlación de 0.014.

Se observa que la variable que más aporta a los componentes principales son **new y old balance of origin.** Esto se debe a que están altamente correlacionadas.

Por tanto, se puede afirmar que las técnicas aplicadas para saber si una transacción es fraudulenta o no, tiene un impacto fuerte sobre el new y old balance of origin sobre sus atributos correlacionados, mientras que otros atributos de las transacciones son los que

crean las diferencias, como pueden ser isRound o Type con las que existen correlaciones muy bajas. Por tanto, se podría afirmar que las características principales que permiten diferenciar, clasificar y llegar a predecir de qué tipo de vino se trata, son aquellas variables que tienen una gran correlación.

4.8. Submuestreo de datos previo al modelado

El submuestreo se refiere a un grupo de técnicas diseñadas para equilibrar la distribución de clases para un conjunto de datos de clasificación que tiene una distribución de clases sesgada.

Una distribución de clases desequilibrada tendrá una o más clases con pocos ejemplos (las clases minoritarias) y una o más clases con muchos ejemplos (las clases mayoritarias). Se entiende mejor en el contexto de un problema de clasificación binaria (dos clases) donde la clase 0 es la clase mayoritaria y la clase 1 es la clase minoritaria (en nuestro caso, fraude o no fraude).

Existen varias técnicas diferentes en la práctica para tratar con conjuntos de datos desequilibrados. La clase de técnicas más común es el muestreo: cambiar los datos presentados al modelo submuestreando clases comunes, sobremuestreando (duplicando) clases raras, o ambos.

La principal ventaja del submuestreo es que los científicos de datos pueden corregir datos no balanceados para reducir el riesgo de que su análisis o algoritmo de aprendizaje automático se desvíe hacia la mayoría (en este caso fraude). Sin volver a muestrear, los científicos podrían ejecutar un modelo de clasificación con un 90 % de precisión. Sin embargo, en una inspección más cercana, encontrarán que los resultados están en gran medida dentro de la clase mayoritaria. Esto se conoce como la paradoja de la precisión.

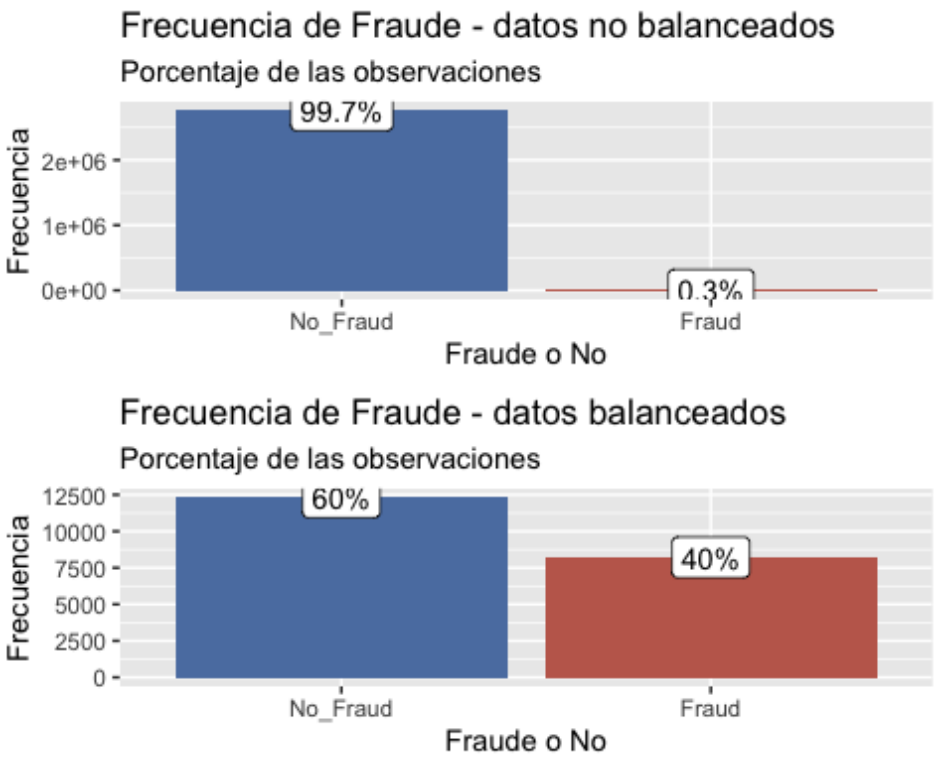
En pocas palabras, los eventos de las minorías son más difíciles de predecir porque hay menos (es decir las transacciones fraudulentas). Un algoritmo tiene menos información de la que aprender. Pero el remuestreo a través del submuestreo puede corregir este problema y hacer que la clase minoritaria sea igual a la clase mayoritaria a los efectos del análisis de datos.

Otras ventajas del submuestreo incluyen menos requisitos de almacenamiento y mejores tiempos de ejecución para el análisis. Menos datos significa que las empresas necesitan menos almacenamiento y tiempo para obtener información valiosa. En nuestro tenemos alrededor de 6M de transacciones que en la vida real no es nada del otro mundo, pero está claro que no disponemos de los mismos recursos para poder análisis tantos datos en unos tiempos razonables.

En la práctica, los métodos de muestreo se utilizan a menudo para equilibrar conjuntos de datos con distribuciones de clases sesgadas porque varios clasificadores han mostrado empíricamente un mejor rendimiento cuando se entrenó en un conjunto de datos equilibrado o balanceado. Sin embargo, estos estudios no implican que los clasificadores no puedan aprender de conjuntos de datos no balanceados, Por ejemplo, otros estudios también han demostrado que algunos clasificadores no mejoran sus actuaciones cuando el conjunto de datos de entrenamiento se equilibra utilizando técnicas de muestreo.

Cómo hemos visto anteriormente, tenemos un conjunto de datos para nada balanceado donde la gran mayoría de registros no son fraude. El objetivo es crear un conjunto de datos más balanceado. En mi opinión, bajar la prevalencia de la clase mayoritaria al 50% elimina totalmente el valor informativo de este desequilibrio. Es por eso que en este apartado crearemos una variable objetivo del 60% - 40%, las instancias minoritarias equivalen a 40% y el número de instancias mayoritarias muestreadas será el 60 %.

```
#encontramos las instancias de fraudes
minoria<- transacciones %>% filter(isFraud == "Fraud") %>% nrow()

mayoria <- transacciones %>% filter(isFraud == "No_Fraud") %>%
sample_n( minoria*6/4)

#creamos el nuevo data set balanceado donde la cantidad de filas
para fraude vendra dada por 6/4 partes de la clase minoritaria
transacciones_bal <- transacciones %>% filter(isFraud == "Fraud")
%>% rbind( mayoria ) %>% arrange(step)

p5 <- transacciones %>% ggplot(aes(x = isFraud, y = (..count..))) +
    geom_bar(fill =  c( "#5b7fb0", "#c2695b" ), stat = "count")+
    geom_label(stat='count',
    aes(label=  paste0(  round(((..count..)/sum(..count..)) ,4)*100 ,  "%" )))+
    labs(x = "Fraude o No", y = "Frecuencia",
    title = "Frecuencia de Fraude - datos no balanceados",
    subtitle = "Porcentaje de las observaciones")

p6 <- transacciones_bal %>% ggplot(aes(x = isFraud, y = (..count..))) +
    geom_bar(fill =  c( "#5b7fb0", "#c2695b" ), stat = "count")+
    geom_label(stat='count',
    aes(label=  paste0(round(((..count..)/sum(..count..)) ,4)*100 ,  "%" ) ) )+
    labs(x = "Fraude o No", y = "Frecuencia",
    title = "Frecuencia de Fraude - datos balanceados ",
    subtitle = "Porcentaje de las observaciones")

grid.arrange(p5,p6)
```



4.9. Conjunto de datos de entrenamiento y prueba

Para este apartado y los apartados futuros relacionados utilizaremos función del paquete CARET. El paquete caret (Classification And Regression Training) contiene funciones para agilizar el proceso de entrenamiento del modelo para problemas complejos de regresión y clasificación.

Después de dividir nuestros datos en entrenamiento (train) y conjunto de prueba (test), crearemos una rutina de preprocesamiento basada en los datos de entrenamiento (el conjunto de prueba se considera invisible).

Nuestra transformación de preproceso incluirá:

- centrado, (es decir, restando de cada columna su propia media)
- escalado, (es decir, dividiendo por la varianza de la columna)

Luego, lo usaremos para transformar ambos conjuntos.
El mismo proceso se aplicará tanto en el submuestreo como en el conjunto de datos original. Necesitamos evaluar nuestro algoritmo basado en la vida real y no en los datos balanceados.

4.9.1. Partición del dataset

Primero, dividimos los datos en dos grupos: un conjunto de entrenamiento y un conjunto de prueba. Para hacer esto, se usa la función **createDataPartition:**

```
#Cremos indices para las filas test/train
indx <- createDataPartition(y = transacciones$isFraud, p = .75, list = FALSE)
train_set <- transacciones[indx,]
test_set <- transacciones[-indx,]

## Centrar y escalar los predictores para el entrenamiento conjunto y
todas las muestras futuras.
pp <- preProcess( train_set[, sapply(train_set,is.numeric) ] ,
                            method = c("center", "scale"))

#Transformamos el conjunto de entrenamiento
train_set <- predict(pp, newdata = train_set)

#Transformamos el conjunto de prueba
test_set <- predict(pp, newdata = test_set)

train_set %>% str()
## 'data.frame':    2077807 obs. of  12 variables:
##  $ step                          : num  -1.7 -1.7 -1.7 -1.7 -1.7 ...
##  $ type                          : Factor w/ 5 levels "CASH_IN","CASH_OUT",
..: 5 2 2 5 2 2 2 5 2 2 ...
##  $ amount                        : num  -0.35913 -0.35913 -0.09962
-0.00606 -0.23418 ...
##  $ Old_Balance_of_Origin         : num  -0.193 -0.193 -0.1312 -0.1495
-0.0843 ...
##  $ New_Balance_of_Origin         : num  -0.11 -0.11 -0.11 -0.11 -0.11 ...
##  $ Old_Balance_of_Destination    : num  -0.405 -0.4 -0.403 -0.403 -0.336 ...
##  $ isFraud                       : Factor w/ 2 levels "No_Fraud","Fraud"
: 2 2 1 1 1 1 1 1 1 ...
##  $ isRound                       : Factor w/ 2 levels "float","round":
2 2 1 1 1 1 1 1 1 ...
##  $ isZero_Old_Balance_of_Or      : logi  FALSE FALSE FALSE FALSE FALSE
TRUE ...
##  $ isZero_New_Balance_of_Or      : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ isZero_Old_Balance_of_Destina: logi  TRUE FALSE FALSE FALSE FALSE
FALSE ...
##  $ isZero_New_Balance_of_Destina: logi  TRUE TRUE FALSE FALSE FALSE
FALSE ...
```

5. Modelado

En este apartado empezará el modelado y trataremos diversos algoritmos, tantos supervisados como no supervisados. Para encontrar los parámetros óptimos del modelo, utilizaremos la validación cruzada para entrenarlo.

Eso significa que cada vez que entrenamos nuestro algoritmo, dividimos los datos de nuestro entrenamiento en particiones, usamos una de ellas y luego lo evaluamos al predecir el resto de las particiones. En nuestro caso, medimos las predicciones resumiendo las probabilidades de cada clase con twoClassSummary. Esta función encaja bien ya que las opciones de predicción (los valores únicos de la variable de destino) son dos, Fraude o No Fraude.

Finalmente, para elegir diferentes medidas de rendimiento, se le dan argumentos adicionales a trainControl. El argumento summaryFunction se usa para pasar una función que toma los valores observados y predichos y estima alguna medida de rendimiento. Dos de estas funciones ya están incluidas en el paquete: defaultSummary y twoClassSummary. Este último calculará medidas específicas para problemas de dos clases, como el área bajo la curva ROC, la sensibilidad y la especificidad. Dado que la curva ROC se basa en las probabilidades de clase pronosticadas (que no se calculan automáticamente), se requiere otra opción. La opción classProbs = TRUE se utiliza para incluir estos cálculos.

Por último, la función seleccionará los parámetros de ajuste asociados con los mejores resultados. Dado que estamos utilizando medidas de rendimiento personalizadas, también se debe especificar el criterio que debe optimizarse. En la llamada para entrenar, podemos usar metric = “ROC” para hacer esto.

```
#Seteamos los parametros de entrenamiento
tr <- trainControl(method = "repeatedcv", #repeated cross validation
                           number = 9, #9 folds
                           repeats = 3, #3 veces
                           classProbs = TRUE,

#calculamos probabilidades para cada clase
                           summaryFunction = twoClassSummary)
```

5.1. Algoritmos supervisados

El **aprendizaje supervisado** es un enfoque de aprendizaje automático que se define mediante el uso de conjuntos de datos etiquetados. Estos conjuntos de datos están diseñados para entrenar o “supervisar” algoritmos para clasificar los datos o predecir los resultados con precisión. Usando entradas y salidas etiquetadas, el modelo puede medir su precisión y aprender con el tiempo.

El aprendizaje supervisado se puede separar en dos tipos de problemas en la minería de datos: ***clasificación y regresión***

5.1.1. Arboles de clasificación y regresión

CART (Classification And Regression Tree) es una variación del algoritmo del árbol de decisión (Decission tree). Puede manejar tareas de clasificación y regresión. CART fue producido por primera vez por Leo Breiman, Jerome Friedman, Richard Olshen y Charles Stone en 1984.

CART es un algoritmo predictivo utilizado en el aprendizaje automático y explica cómo se pueden predecir los valores de la variable objetivo en función de otras variables llamadas predictoras. Es un árbol de decisiones en el que cada bifurcación se divide en una variable predictora y cada nodo tiene una predicción para la variable objetivo al final.

El algoritmo **rpart** que utilizaremos a continuación por defecto la impureza de Gini para dividir el conjunto de datos en un árbol de decisión. Lo hace buscando la mejor homogeneidad para los subnodos, con la ayuda del criterio del índice de Gini.

El índice Gini es una métrica para las tareas de clasificación en CART. Almacena la suma de probabilidades al cuadrado de cada clase. Calcula el grado de probabilidad de que una variable específica esté mal clasificada cuando se elige aleatoriamente y una variación del coeficiente de Gini. Funciona en variables categóricas, proporciona resultados de “éxito” o “fracaso” en nuestro caso fraude o no fraude y, por lo tanto, solo realiza divisiones binarias.

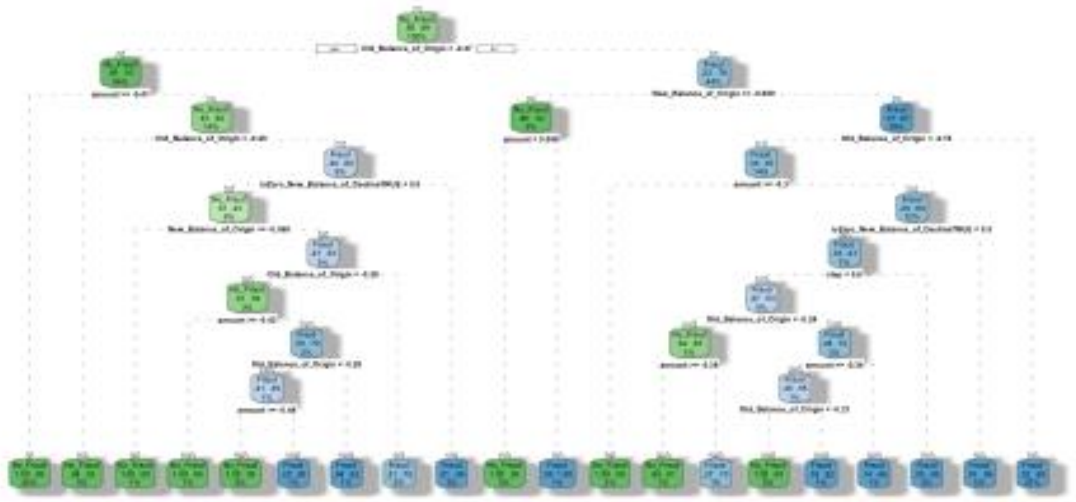
```
#Model Training - Regression Tree con CARET
start_time <- Sys.time()
set.seed(123)
rpart_model_b <- train(isFraud ~ .,
#queremos predecir si es fraude o no usando todo los predictores
data = train_set_b, #usamos el dataset balanceado
method = "rpart", # libreria que contiene el algoritmo del arbol
tuneLength = 10,
# El número de diferentes valores predeterminados utilizados
para optimizar el parámetro principal del modelo.
metric = "ROC", # estimar el éxito usando  área bajo la curva ROC
trControl = tr, #usamos cross validation definida arriba
parms=list(split='information'))

end_time <- Sys.time()

end_time - start_time
## Time difference of 9.87 secs
```

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
rpart_test_pred_b <- predict(rpart_model_b, test_set_b)

# prediccion en el conjunto de prueba
confusionMatrix(test_set_b$isFraud, rpart_test_pred_b)

# resultado de predicciones

## Confusion Matrix and Statistics
##
##           Reference
## Prediction No_Fraud Fraud
##   No_Fraud     2964   115
##   Fraud          12  2041
##
##                Accuracy : 0.9753
##                  95% CI : (0.9706, 0.9793)
##     No Information Rate : 0.5799
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9489
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9960
##             Specificity : 0.9467
##          Pos Pred Value : 0.9627
##          Neg Pred Value : 0.9942
##              Prevalence : 0.5799
##          Detection Rate : 0.5776
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9713
##
##        'Positive' Class : No_Fraud
##
```

Podemos ver que tenemos una precisión del 97%, lo cual es un porcentaje alto. Sin embargo, no podemos sacar de momento ninguna conclusión puesto que no hemos probado otros algoritmos con los cual comparar los resultados del modelo generado.

A continuación, mostraremos el plot del modelo creado previamente.



Rattle 2023-Jun-12 00:34:05 jespinlo10

Otro algoritmo que podemos utilizar en R es el **C5.0.** Este método se basa en la entropía para seleccionar características óptimas para dividir. La alta entropía significa que hay poca o ninguna homogeneidad dentro del conjunto de valores de clase. El algoritmo tiene como objetivo encontrar divisiones que reduzcan la entropía.

La variable por la que clasificaremos es el campo isFraud donde vemos si la transacción es fraudulenta o no. De esta forma, tendremos un conjunto de datos para el entrenamiento y uno para la validación.

```
set.seed(666)
y <- transacciones_bal[,7]
n = c(1,2,3,4,5,6,8,9,10,11,12)
X <- transacciones_bal[,n]
```

Podemos crear directamente una función de un parámetro, en este caso del "split_prop". Definimos un parámetro que controla el split de forma dinámica en la prueba.

```
split_prop <- 4
indexes = sample(1:nrow(transacciones_bal),
size=floor(((split_prop-1)/split_prop)*nrow(transacciones_bal)))
trainX<-X[indexes,]
trainy<-y[indexes]
testX<-X[-indexes,]
testy<-y[-indexes]
```

Después de una extracción aleatoria de casos, es altamente recomendable efectuar un análisis de datos mínimo para asegurarnos de no obtener clasificadores sesgados por los valores que contiene cada muestra. En este caso, verificaremos que la proporción de las transacciones es más o menos constante en ambos conjuntos:

```
summary(trainy)

## No_Fraud    Fraud
##     9236     6163

summary(testy)

## No_Fraud    Fraud
##     3083     2050
```

Se crea el árbol de decisión usando los datos de entrenamiento (no hay que olvidar que la variable outcome es de tipo factor). Además, tenemos que pasar algunas variables booleanas a enteros 1/0 ya que el algoritmo C50 no acepta variables booleanas.

```
trainX$isZero_Old_Balance_of_Or = as.numeric(trainX$isZero_Old_Balance_of_Or)
trainX$isZero_New_Balance_of_Or = as.numeric(trainX$isZero_New_Balance_of_Or)
trainX$isZero_Old_Balance_of_Destina =
as.numeric(trainX$isZero_Old_Balance_of_Destina)
trainX$isZero_New_Balance_of_Destina =
as.numeric(trainX$isZero_New_Balance_of_Destina)
```

A continuación, aplicamos el agoritmo C5.0:

```
trainy = as.factor(trainy)
C50_model <- C50::C5.0(trainX, trainy,rules=TRUE )
C50_model

##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
## Rule-Based Model
```

```
## Number of samples: 15399
## Number of predictors: 12
##
## Number of Rules: 30
##
## Evaluation on training data (15399 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      30   86( 0.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    9173    63    (a): class No_Fraud
##      23  6140    (b): class Fraud
##
##
##  Attribute usage:
##
##   99.86% isZero_New_Balance_of_Destina
##   74.84% Old_Balance_of_Origin
##   52.26% type
##   49.62% amount
##   39.36% isZero_New_Balance_of_Or
##   33.68% isZero_Old_Balance_of_Destina
##    1.16% step
##    0.75% isRound
##
##
## Time: 0.2 secs
```

El árbol obtenido tiene una tasa de error del 0.6%, con lo que podemos decir que se ha obtenido una muy buena clasificación.

Como contrapartida observamos que se han generado 44 reglas, por eso se explicarán las que se han considerado más relevantes y han obtenido una mayor validez:

Regla 23:

- type = TRANSFER
- isZero_Old_Balance_of_Destina > 0
- isZero_New_Balance_of_Destina > 0

Fraude 100%

Regla 1:

- step <= 405
- amount > 30548.46
- Old_Balance_of_Origin > 145
- Old_Balance_of_Origin <= 30827
- isZero_New_Balance_of_Destina <= 0

No Fraude 99%

Regla 2:

- amount > 38180.99
- Old_Balance_of_Origin > 145
- Old_Balance_of_Origin <= 37323.15

No Fraude 99%

Regla 3:

- amount > 51856.4
- Old_Balance_of_Origin > 145
- Old_Balance_of_Origin <= 51711

No Fraude 99%

Regla 4:

- amount > 22695.74
- Old_Balance_of_Origin > 145
- Old_Balance_of_Origin <= 22807

No Fraude 99%

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
testX$isZero_Old_Balance_of_Or = as.numeric(testX$isZero_Old_Balance_of_Or)
testX$isZero_New_Balance_of_Or = as.numeric(testX$isZero_New_Balance_of_Or)
testX$isZero_Old_Balance_of_Destina =
as.numeric(testX$isZero_Old_Balance_of_Destina)
testX$isZero_New_Balance_of_Destina =
as.numeric(testX$isZero_New_Balance_of_Destina)
predicted_model <- predict( model, testX, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",
```

```
100*sum(predicted_model == testy) /
                length(predicted_model)))
## [1] "La precisión del árbol es: 98.7 %"
```

Cuando existen pocas clases, la calidad de la predicción se puede analizar mediante una matriz de confusión que identifica los tipos de errores cometidos.

```
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf

##           Predicted
## testy      No_Fraud Fraud
##   No_Fraud     3058    25
##   Fraud          23  2027
```

Por lo tanto, vemos que el porcentaje de precisión del árbol C5.0 es ligeramente mejor que el árbol rpart utilizado anteriormente.

5.1.2. Random Forest

El algoritmo **Random Forest** consiste en una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de ellos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y después los promedia.

Bagging es una técnica usada para reducir la varianza de las predicciones a través de la combinación de los resultados de varios clasificadores, cada uno moldeados con diferentes subconjuntos presos de la misma población. Random forest permite trabajar en base de datos muy grandes, donde permite crear múltiples subconjuntos de datos, construir múltiples modelos y combinar estos modelos. Es muy útil para problemas de clasificación y regresión. Al igual que con el árbol, usaremos los mismos parámetros de validación cruzada.

Esta vez, en lugar de tuneLength que conecta los valores predeterminados para los hiperparámetros del modelo, usaremos un TuneGrid donde tenemos estos parámetros predeterminados.

Estos hiperparámetros incluyen:

- Ntree: Número de árboles por crear. Esta variable no debe ser un número demasiado pequeño, para garantizar que cada fila de entrada se predice al menos algunas veces.
- Mtry: número de variables mostradas al azar como candidatos a cada división.

Como hicimos con el árbol, entrenaremos nuestro bosque usando el conjunto de datos balanceado. Luego, evaluaremos su éxito en la misma muestra aleatoria que generamos anteriormente.

Generamos el modelo random forest con hiperparametros por defecto. Por defecto, el número de árboles es 500 (ntree) y el número de variables probadas en cada división es 3 (mtry) en ese caso.

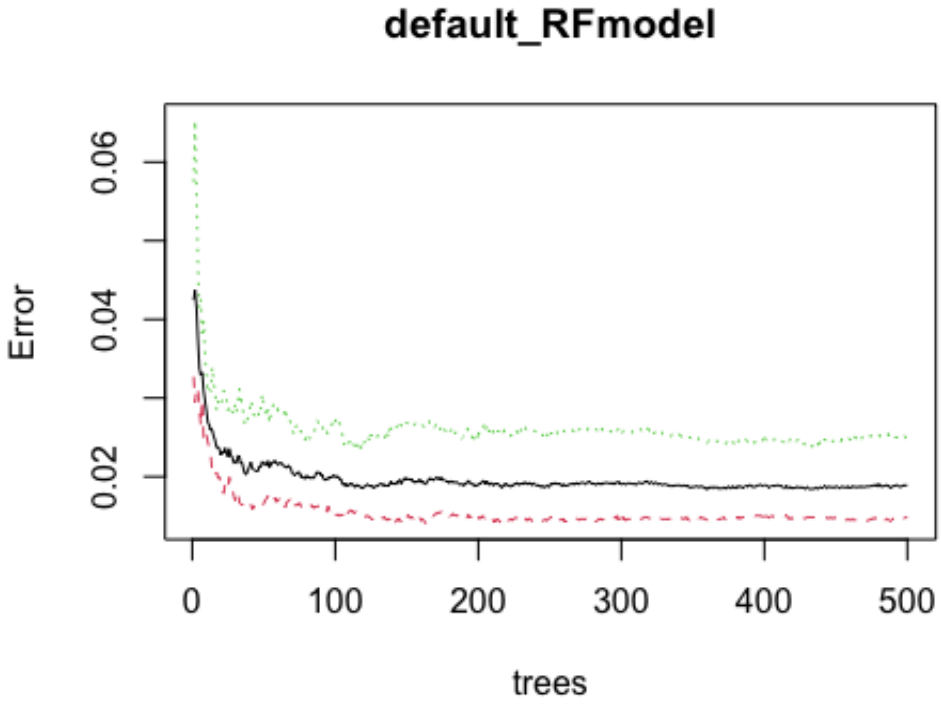
```
start_time <- Sys.time()
default_RFmodel <- randomForest(isFraud~ ., data = train_set_b[,1:12],
importance = TRUE)
end_time <- Sys.time()
end_time - start_time
## Time difference of 20.05605 secs

print(default_RFmodel)
##
## Call:
##  randomForest(formula = isFraud ~ ., data = train_set_b[, 1:12],
 importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.89%
## Confusion matrix:
##          No_Fraud Fraud class.error
## No_Fraud     9103   137  0.01482684
## Fraud         154  6006  0.02500000
```

Por defecto, el número de árboles es 500 y el número de variables probadas en cada división es 3 en ese caso. Se ha obtenido una tasa de error de 1.89%.

Al crearse distintos números de árboles (500) es muy difícil representar el resultado en un único árbol como se ha realizado en el caso anterior, por lo que en este caso se graficará en forma de gráfica.

A continuación, vamos a graficar el modelo generado previamente:

```
plot(default_RFmodel)
```



En este gráfico se observa la tasa de error por el número de árboles. Puede observarse que a medida que aumentan el número de árboles, la tasa de error disminuye.

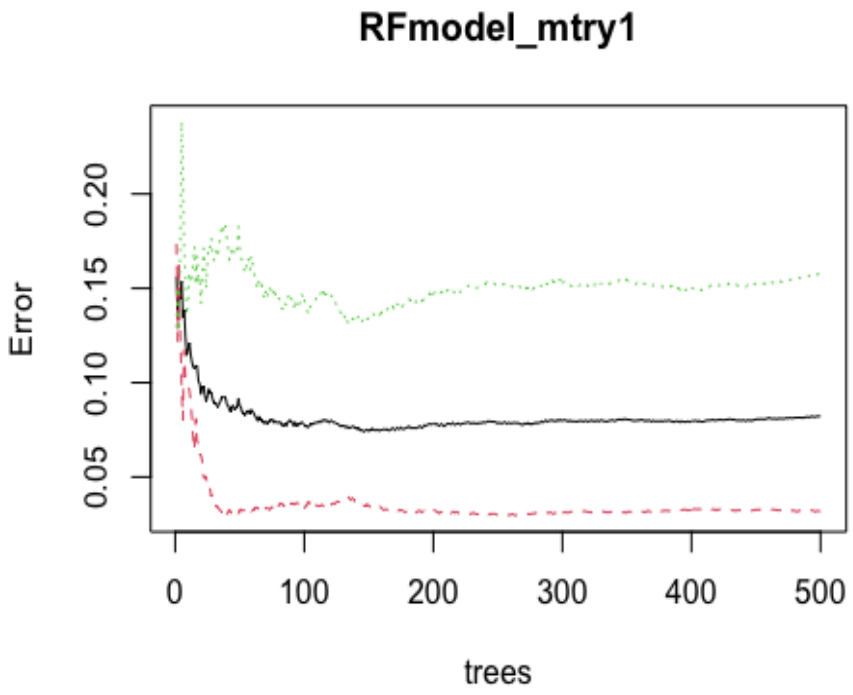
A continuación, se intenta mejorar la tasa de error modificando el parámetro mtry, dónde ahora será 1 y manteniendo el número de árboles utilizados para el caso anterior.

```
start_time <- Sys.time()
RFmodel_mtry1 <- randomForest(isFraud~ ., data = train_set_b[,1:11],

ntree = 500, mtry = 1 , importance = TRUE)
end_time <- Sys.time()
end_time - start_time

## Time difference of 8.606357 secs

print(RFmodel_mtry1)

##
## Call:
##  randomForest(formula = isFraud ~ ., data = train_set_b[, 1:11],

ntree = 500, mtry = 1, importance = TRUE)
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 1
```

```
##
##         OOB estimate of  error rate: 8.21%
## Confusion matrix:
##          No_Fraud Fraud class.error
## No_Fraud     8946   294  0.03181818
## Fraud         971  5189  0.15762987
```

Modificando el parámetro mtry a 1 no se ha logrado reducir la tasa de error, no obstante, el tiempo de ejecución ha disminuido notablemente a 8.6 segundos.

```
plot(RFmodel_mtry1)
```



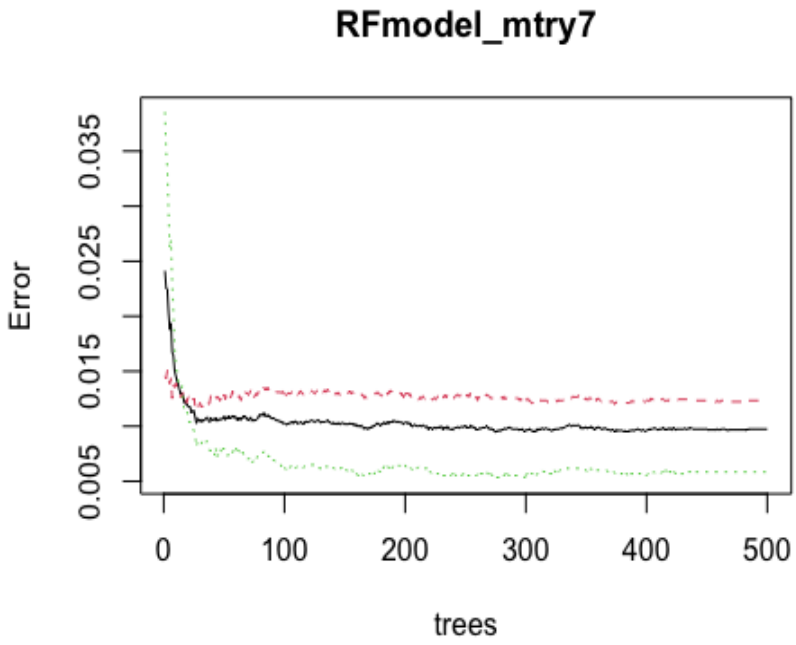
Finalmente, probaremos a aumentar el valor de mtry para ver cómo se comporta el algoritmo. El numero de predictores incluidos en la división de nodos será 7.

```
start_time <- Sys.time()
RFmodel_mtry7 <- randomForest(isFraud~ ., data = train_set_b[,1:11],
ntree = 500, mtry = 7 , importance = TRUE)
end_time <- Sys.time()
end_time - start_time
## Time difference of 16.7942 secs
```

```
print(RFmodel_mtry7)
##
## Call:
##  randomForest(formula = isFraud ~ ., data = train_set_b[, 1:11],      ntree = 5
00, mtry = 7, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.97%
## Confusion matrix:
##          No_Fraud Fraud class.error
## No_Fraud     9126   114 0.012337662
## Fraud          36  6124 0.005844156
```

Modificando el parámetro mtry a 7 se ha logrado reducir la tasa de error a 0.97%, no obstante, el tiempo de ejecución ha aumentado notablemente a 17 segundos. Lo cual es normal puesto que usando más variables predictoras en el cálculo.

```
plot(RFmodel_mtry7)
```



Cómo en el caso anterior, se observa que a medida que se incrementa el número de árboles, la tasa de error disminuye.

A continuación, obtendremos una tabla mostrando, en primer lugar, la clasificación real del dataset y en segundo lugar cómo la ha clasificado nuestro algoritmo. También se extraerá el porcentaje de fiabilidad de nuestro algoritmo y podremos comparar si ha tenido mejores resultados que con el algoritmo C5.0.

Obtenemos la tabla de clasificación de nuestro juego de datos

```
default_RFmodel$confusion
```

```
##          No_Fraud Fraud class.error
## No_Fraud     9103   137  0.01482684
## Fraud         154  6006  0.02500000
```

Lo aplicamos a nuestros datos de prueba y obtenemos la tabla de clasificación y la tasa de fiabilidad.

```
test_set_b$step <- as.numeric(as.character(test_set_b$step))
predValid <- predict(default_RFmodel, test_set_b, type = "class")
mean(predValid == test_set_b$isFraud)
```

```
## [1] 0.9814887
```

```
table(predValid,test_set_b$isFraud)
```

```
##
## predValid  No_Fraud Fraud
##   No_Fraud     3029    45
##   Fraud          50  2008
```

Observamos que hemos obtenido un porcentaje de fiabilidad del 98.1% en cuanto a la clasificación de transacciones.

Finalmente vamos a obtener la tabla de clasificación y la tasa de fiabilidad para el random forest creado cuyo mtry era 7 y daba mejor tasa de fiabilidad.

```
predValid7 <- predict(RFmodel_mtry7, test_set_b, type = "class")
mean(predValid7 == test_set_b$isFraud)
```

```
## [1] 0.9933749
```

```
table(predValid7,test_set_b$isFraud)
```

```
##
## predValid7 No_Fraud Fraud
##   No_Fraud     3052     7
##   Fraud          27  2046
```

Observamos que para mtry = 7, hemos obtenido un porcentaje de fiabilidad del 99% en cuanto a la clasificación de transacciones. Es ligeramente mejor a la precisión encontrada en el algoritmo C5.0 calculada antes, la cual era 98%.

5.2. Algoritmos no supervisados

El **aprendizaje no supervisado** utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de datos sin etiquetar. Estos algoritmos descubren patrones ocultos en los datos sin necesidad de intervención humana (por tanto, están “sin supervisar”).

Los modelos de aprendizaje no supervisado se utilizan para tres tareas principales: ***agrupación, asociación y reducción de la dimensionalidad.***

La principal diferencia es el uso de conjuntos de datos. Por decirlo simplemente, el aprendizaje supervisado utiliza datos de entrada y salida etiquetados, mientras que un algoritmo de aprendizaje no supervisado no.

5.2.1. K-Means

Durante este apartado deberemos aplicar un modelo no supervisado y basado en el concepto de distancia, sobre el juego de datos. El algoritmo escogido para realizar un modelo no supervisado es el K-Means basado en la métrica euclídea, cuyo objetivo es conocer y agrupar los datos en grupos (**clustering**).

Por eso lo primero que hay que hacer es un análisis de los posibles clusters a los que pueden pertenecer las transacciones. Por eso generaremos un nuevo juego de datos eliminando la variable isfraud. De esta forma conseguimos un juego de datos idóneo para aplicar un algoritmo no supervisado.

Utilizaremos un conjunto de datos normalizado donde también se tendrá que eliminar la variable isFraud.

```
#Comenzamos por pasar las variables booleanas a intengers con 0s y 1s
transacciones_bal$isZero_Old_Balance_of_Or =
as.numeric(transacciones_bal$isZero_Old_Balance_of_Or)
transacciones_bal$isZero_New_Balance_of_Or =
as.numeric(transacciones_bal$isZero_New_Balance_of_Or)
transacciones_bal$isZero_Old_Balance_of_Destina =
as.numeric(transacciones_bal$isZero_Old_Balance_of_Destina)
transacciones_bal$isZero_New_Balance_of_Destina =
```

```
as.numeric(transacciones_bal$isZero_New_Balance_of_Destina)
transacciones_bal$step <-
as.numeric(as.character(transacciones_bal$step))
```

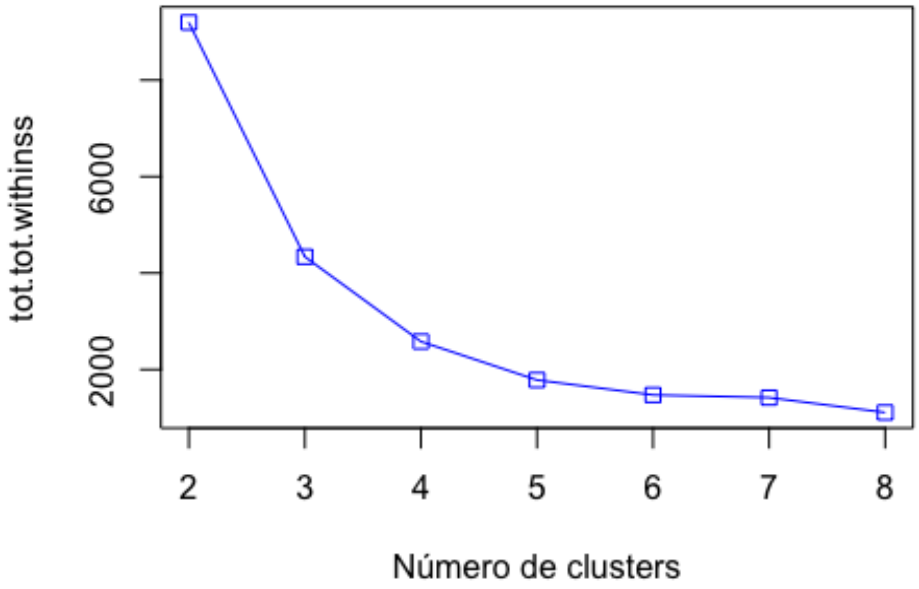
A continuación, vamos a normalizar las demás columnas para asegurarnos de que cada variable contribuye por igual a nuestro análisis.

```
# Definimos la función de normalización
 nor <-function(x) { (x -min(x))/(max(x)-min(x))}
# Guardamos un dataset nuevo normalizado
r = c(1,3,4,5,6,9,10,11,12)
transacciones_bal2<- transacciones_bal %>% select(all_of(r))
#transacciones_bal_NOR <- as.data.frame(lapply(transacciones_bal2[,
c(1,3,4,5,6,8,9,10,11)], nor))
transacciones_bal2$isFraud <- NULL
n = c(1,2,3,4,5,6,7,8,9)
```

A continuación, se aplicará la métrica Euclidiana que consiste en encontrar la menor suma de los cuadrados de las distancias de los puntos de cada grupo respecto a su centro (withinss), con la mayor separación entre centros de grupos (betweenss).

Este método es conocido como Elbow (codo), que no es más que la selección del número de clusters sobre la base de la inspección de la gráfica que se obtiene al iterar con el mismo conjunto de datos para diferentes valores del número de clusters . Se seleccionará el valor que se encuentra en el codo de la curva.

```
set.seed(666)
library(cluster)
library(factoextra)
library(NbClust)
d <- daisy(transacciones_bal_NOR)
resultados <- rep(0, 8)
for (i in c(2,3,4,5,6,7,8))
{
  fit           <- kmeans(transacciones_bal_NOR, i)
  resultados[i] <- fit$tot.withinss
}
plot(2:8,resultados[2:8],type="o",col="blue",pch=0,xlab="Número de clusters",
ylab="tot.tot.withinss")
```

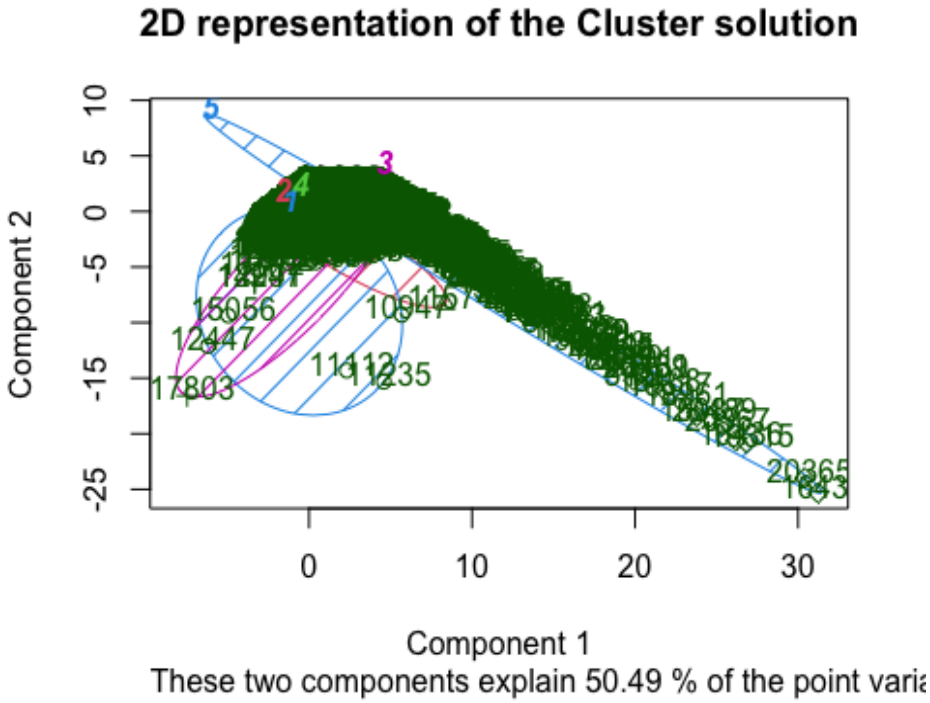


En este caso se observa que la curva comienza a estabilizarse a partir del valor 5, por lo que el valor óptimo para aplicar el algoritmo no supervisado K-means con esta métrica es 5.

```
##Para tener el mismo resultado cada vez que ejecutamos el código,
hacemos set.seed(4).
##Esto significa que R probará 4 tareas iniciales aleatorias diferentes y
#después seleccionará la que tenga la variación más baja dentro del clúster
set.seed(4)
transacciones5<- kmeans(transacciones_bal_NOR, 5)
```

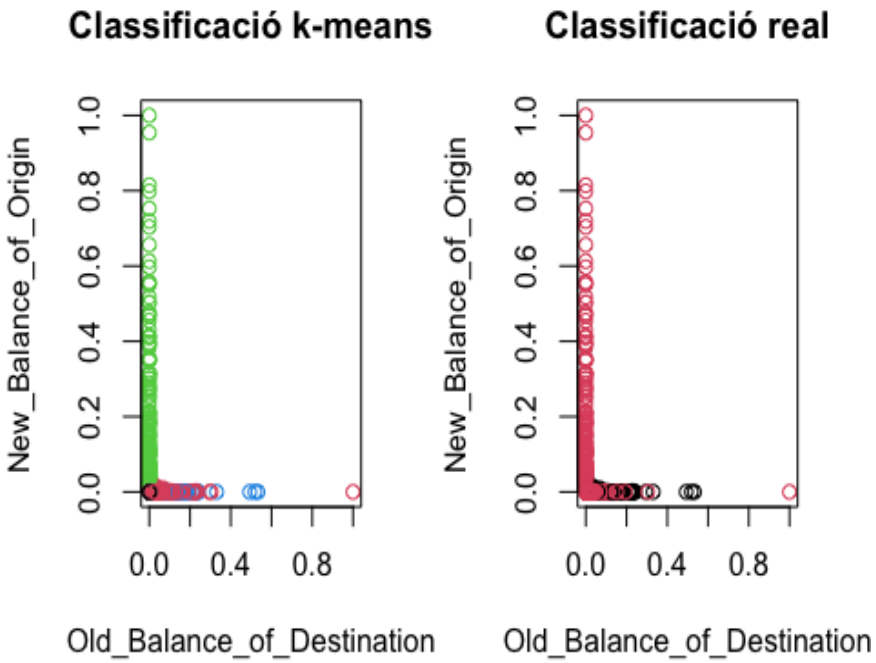
Para visualizar el clúster podemos utilizar la función clusplot. Miremos la agrupación con 5 clusters que son los que ha encontrado con el método elbow.

```
clusplot(transacciones_bal_NOR, transacciones5$cluster, main='2D representation of
the Cluster solution',
         color=TRUE, shade=TRUE,
         labels=2, lines=0)
```

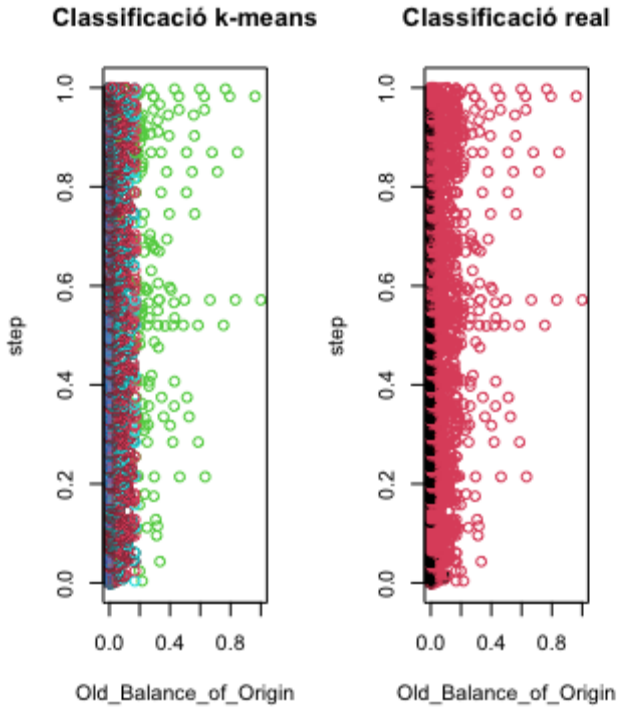


Utilizaremos ahora las variables que tienen mayor correlación como hemos visto en apartados anteriores y compararemos dos a dos:

```
# Old balance of destination y new balance of origin
old.par <- par(mfrow=c(1, 2))
plot(transacciones_bal_NOR[c(5,4)], col=transacciones5$cluster,

main="Classificació k-means")
plot(transacciones_bal_NOR[c(5,4)], col=as.factor(transacciones_bal$isFraud),

main="Classificació real")
```




```
# Step y old balance of origin
old.par <- par(mfrow=c(1, 3))
plot(transacciones_bal_NOR[c(3,1)], col=transacciones5$cluster,
main="Classificació k-means")
plot(transacciones_bal_NOR[c(3,1)], col=as.factor(transacciones_bal$isFraud),
main="Classificació real")
par(old.par)
```



A continuación, mostramos la matriz de confusión para cada cluster.

```
d=table(transacciones5$cluster,transacciones_bal$isFraud)
d
```

```
##
##      No_Fraud Fraud
##    1        9  3919
##    2     4772  2836
##    3      279   158
##    4     5812    35
##    5     1447  1265
```

Los resultados de la matriz nos dicen que el cluster 1 la gran mayoría de las transacciones ha clasificado cómo Fraude. En el cluster 4, la gran mayoría clasifica como No_fraude. Un comportamiento parecido tiene el cluster 3 y en los clusters restantes hay bastante semejanza en cuanto el numero de clasificaciones.

A partir de la matriz de confusión extraeremos el número de casos correctamente clasificados y el porcentaje de precisión del modelo.

```
cat("Total de casos correctamente clasificados: 3919+4775+280+5815+1440 =",
    3919+4775+280+5815+1440  , "\n")
```

```
## Total de casos correctamente clasificados: = 16229
```

```
cat("Total de casos incorrectamente clasificados: 9+2836+158+35+1265=",
    9+2836+158+35+1265, "\n")

## Total de casos incorrectamente clasificados: = 4303

cat("Precisión del modelo: (13232/(13232+7300))*100= ",
    (16229/(16229+4303))*100, "%\n")

## Precisión del modelo: (13232/(13232+7300))*100=  79%
```

Aplicando una métrica de distancia euclídea para calcular el número de clusters, se ha obtenido una precisión del modelo del 79% (k=5), por lo que se puede afirmar que esta métrica no es óptima para aplicar sobre nuestro juego de datos.

5.2.2. Distancia de Manhattan

La segunda métrica seleccionada para aplicar el modelo no supervisado es con la distancia de manhattan que consiste en calcular las diferencias absolutas entre coordenadas de par de objetos.

La distancia de gauss hemos decidido no usarla ya que esta distancia no tiene en cuenta la correlación entre variables. Además, también tenemos la distancia de mahalanobis es una métrica de distancia multivariante eficaz que mide la distancia entre un punto y una distribución.

Hemos escogido la distancia de manhattan para ver la diferencia con la distancia euclídea. En teoría, la distancia de Manhattan funcionaría bien con datos con una dimensionalidad grande, por tanto queremos ver la poca eficacia de este modelo dado que nuestro juego de datos tiene una dimensionalidad pequeña.

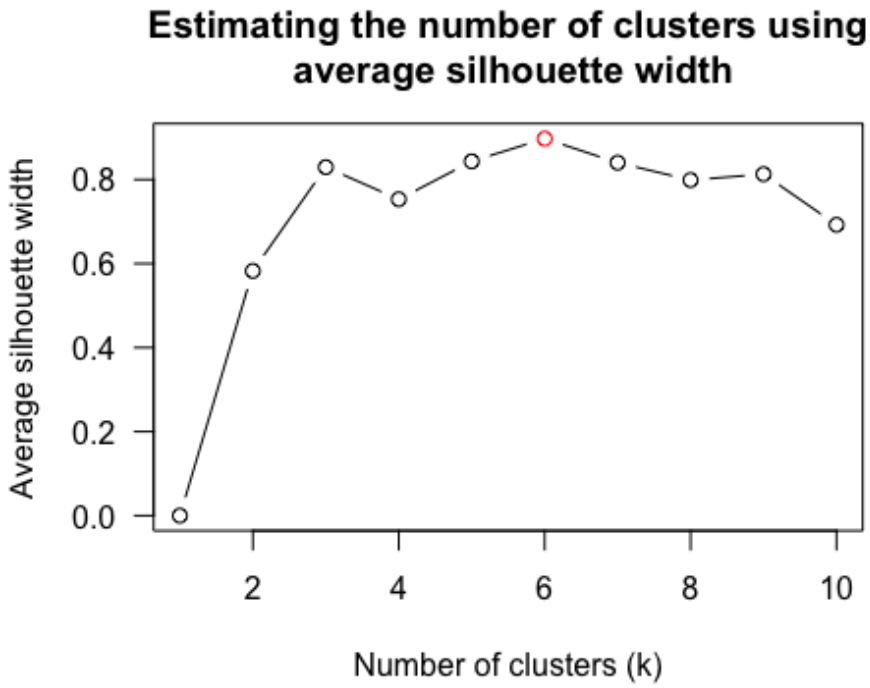
Para averiguar el número de clusters utilizaremos el algoritmo aglomerativo jerárquico. En este método, cada observación se asigna a nuestro cluster. Después, se calcula la similitud (o distancia) entre cada uno de los clusters y los dos clusters más similares se fusionan en uno.

Cada observación comienza en su propio cluster y los clusters se fusionan sucesivamente. Los criterios de enlace determinan la métrica utilizada para la estrategia de combinación.

Ward minimiza la suma de las diferencias al cuadrado dentro de todos los clusters. Es un enfoque que minimiza la varianza y, en este sentido, es similar a la función objetivo k-means, pero abordado con un enfoque jerárquico aglomerativo.

Para poder aplicar este algoritmo deberemos escalar los datos y posteriormente mostrar el dendograma para saber dónde cortarlo y poder escoger el número de clusters.

```
#Utilizaremos el dataset escalado previamente
#Calculamos la distancia manhattan del fecha set
d_manh <- dist(transacciones_scale, method = 'manhattan')

#Utilizaremos la variable d_manh
hc<- hclust(d_manh, method = "ward")

dend <- as.dendrogram(hc)
dend_k<- find_k(dend)
plot(dend_k)
```




```
Plot(color_branches(dend, k = dend_k$nc))
```

Obtenemos que el nombre de clusters será:

```
groups <- cutree(hc, k = 6)

table(groups, transacciones_bal$isFraud)

##
## groups No_Fraud Fraud
##      1     1147  2345
##      2     3913  2505
##      3     4422  1276
##      4     1572  1021
##      5     387   773
##      6     878   293
```

A partir de la matriz de confusión extraeremos el número de casos correctamente clasificados y el porcentaje de precisión del modelo.

```
cat("Total de casos correctamente clasificados:  1147+2505+4422+1572+773+878 =",
   1147+2505+4422+1572+773+878, "\n")

## Total de casos correctamente clasificados:  = 11927

cat("Total de casos incorrectamente clasificados2345+3913+1276+1021+387+293=",
    2345+3913+1276+1021+387+293, "\n")

## Total de casos incorrectamente clasificados: 9235

cat("Precisión del modelo: (11927/(1551+4912))*100= ",
    (11927/(11927+9235))*100, "%\n")

## Precisión del modelo: (11927/(11927+9235))*100=  55.02143 %
```

Para visualizar el clúster podemos utilizar la función clusplot. Miremos la agrupación con 6 clusters:

```
clusplot(transacciones_scale, groups,

main='2D representation of the Cluster solution',
         color=TRUE, shade=TRUE,
         labels=2, lines=0)
```

Como puede apreciarse a simple vista en la tabla, después de aplicar la métrica de manhattan a través del algoritmo de aglomeración se ha obtenido una precisión del 55%, la cual ha realizado una agrupación correcta pero inferior a la vista cuando se ha aplicado la distancia euclídea sobre el algoritmo de K-Means.

Esto se debe a que nuestro juego de datos presenta un mejor comportamiento cuando se calculan las distancias aplicando el cuadrado de la diferencia, donde permite realizar un

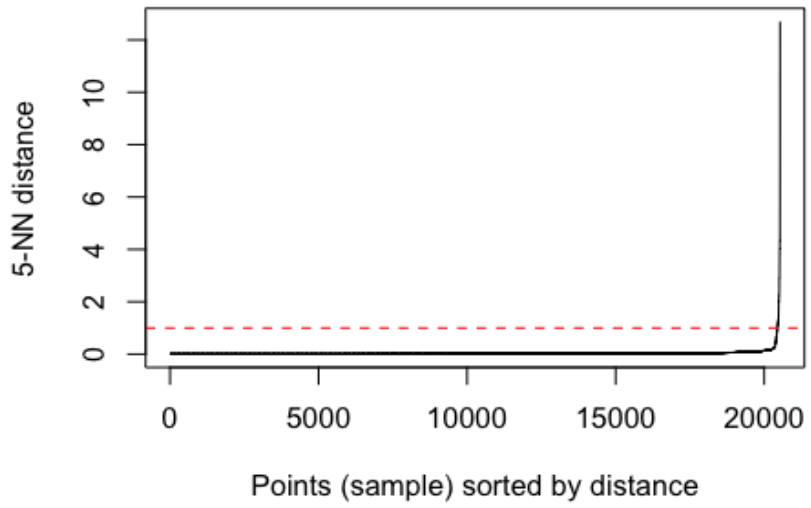
agrupamiento más exacto. Por tanto, en este caso, nos quedaríamos con la primera métrica aplicada para realizar una correcta clasificación de las diferentes transacciones.

5.2.3. DBSCAN

El método DBSCAN es un algoritmo de agrupación basado en la densidad que forma grupos de regiones densas de puntos de datos ignorando las áreas de baja densidad (considerándolas como ruido).

Por este modelo tomamos las 3 variables que tienen mejor correlación y que extraímos en los apartados anteriores.

```
n = c(1,4,5)
transaccionesScan= transacciones_bal %>% select(all_of(n))
transacciones_scan_nor <- as.data.frame(lapply(transaccionesScan, nor))
transacciones_scan_scale <- scale(transacciones_scan_nor)
kNNdistplot(transacciones_scan_scale, k = 5)
abline(h = 1, col = "red", lty = 2)
```

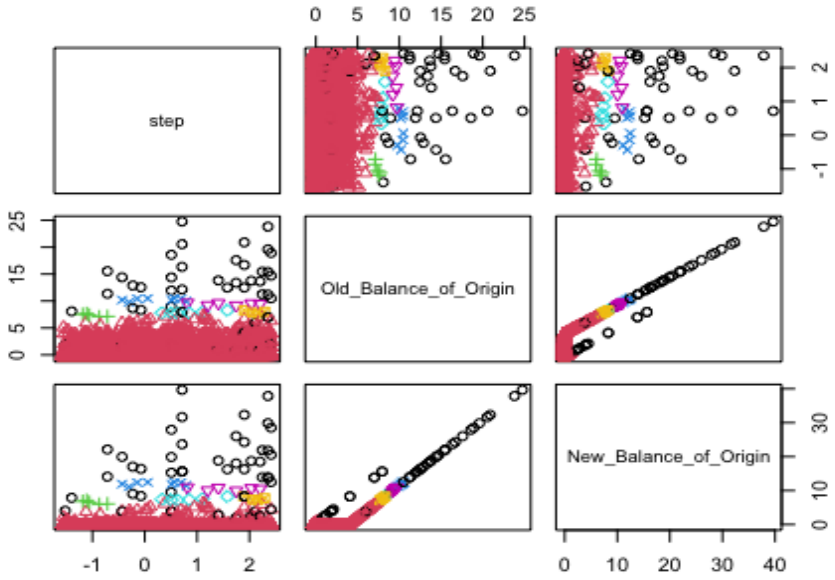



```
res_1<- dbscan(transacciones_scan_scale, eps = 1 , minPts = 5)
res_1

## DBSCAN clustering for 20532 objects.
## Parameters: eps = 1, minPts = 5
## The clustering contains 6 cluster(s) and 49 noise points.
```

```
##
##     0     1     2     3     4     5     6
##    49 20454     5     8     6     6     4
##
## Available fields: cluster, eps, minPts
```

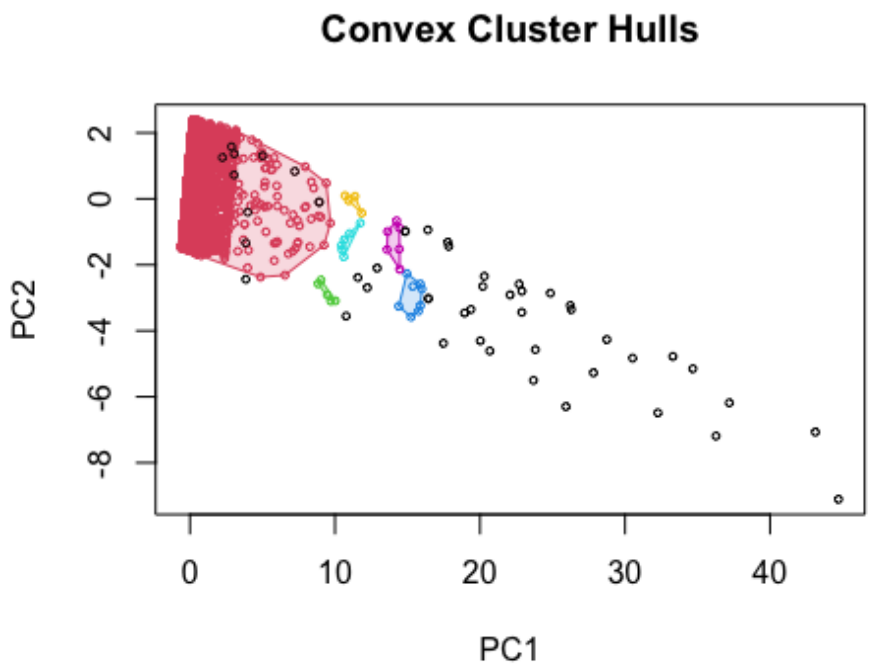
Hemos obtenido 6 clusters y 49 outliers que se encuentran en el cluster 0. Ahora mostraremos los clusters con los puntos ruidosos.

```
plot(as.data.frame(transacciones_scan_scale), col = res_1$cluster + 1L,
pch = res_1$cluster + 1L)
```



Seguimos adelante con una representación gráfica que nos muestra los clusters mediante formas convexas.

```
hullplot(transacciones_scan_scale, res_1)
```



Vemos que los clusters están bien identificados y separados. Ahora mostraremos la matriz de confusión para explicar el resultado de la agrupación.

```
d0=table(res_1$cluster,transacciones_bal$isFraud)
d0
```

```
##
##      No_Fraud Fraud
##   0         1    48
##   1     12318  8136
##   2         0     5
##   3         0     8
##   4         0     6
##   5         0     6
##   6         0     4
```

En el cluster 0 tenemos los 49 outliers, es decir, casos que no están asignados a ningún cluster. Tenemos 1 cluster identificado, en el primer cluster tenemos 12318 transacciones no fraudulentas y 8136 fraudulentas. En los otros 2-6 clusters las transacciones sólo clasifican como fraudes.

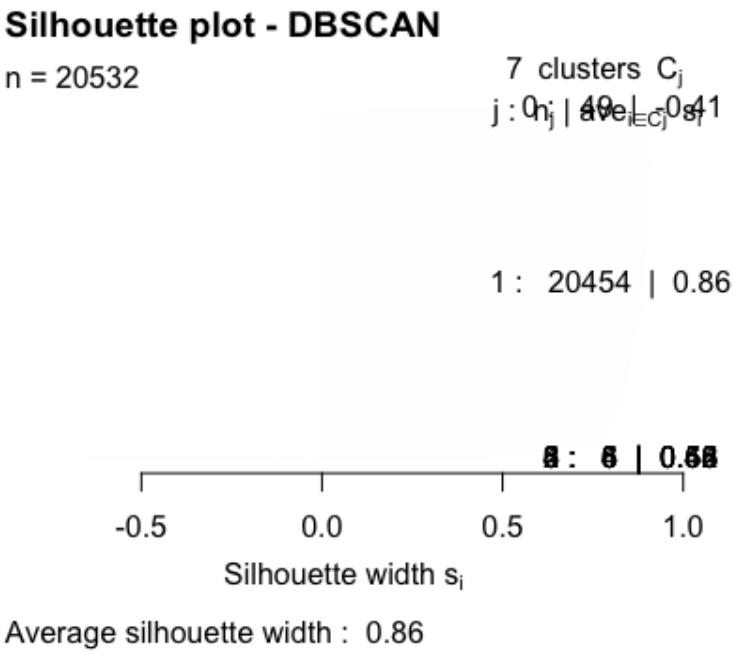
A partir de la matriz extraeremos el número de casos correctamente clasificados y el porcentaje de precisión.

```
cat("Total de casos correctamente classificados: 12318 =",
    12318, "\n")

## Total de casos correctamente classificados: 12318 = 12318

cat("Total de casos incorrectamente clasificados: 8136+5+8+6+6+4=",
    8136+5+8+6+6+4, "\n")

## Total de casos incorrectamente clasificados: 8136+5+8+6+6+4= 8165

cat("Precisión: (14844/(4844+1619))*100= ",
    ( 12318/(12318+8165))*100, "%\n")

## Precisión: (12318/(12318+8165))*100=  60.13768 %
```

El **coeficiente silhouette** se calcula utilizando la distancia media del cluster entre puntos y la distancia media del cluster más cercano. Por ejemplo, un clúster con muchos puntos de datos muy cerca unos de otros (alta densidad) y que está lejos del siguiente clúster más cercano, tendrá un coeficiente de silhouette cercano a 1.

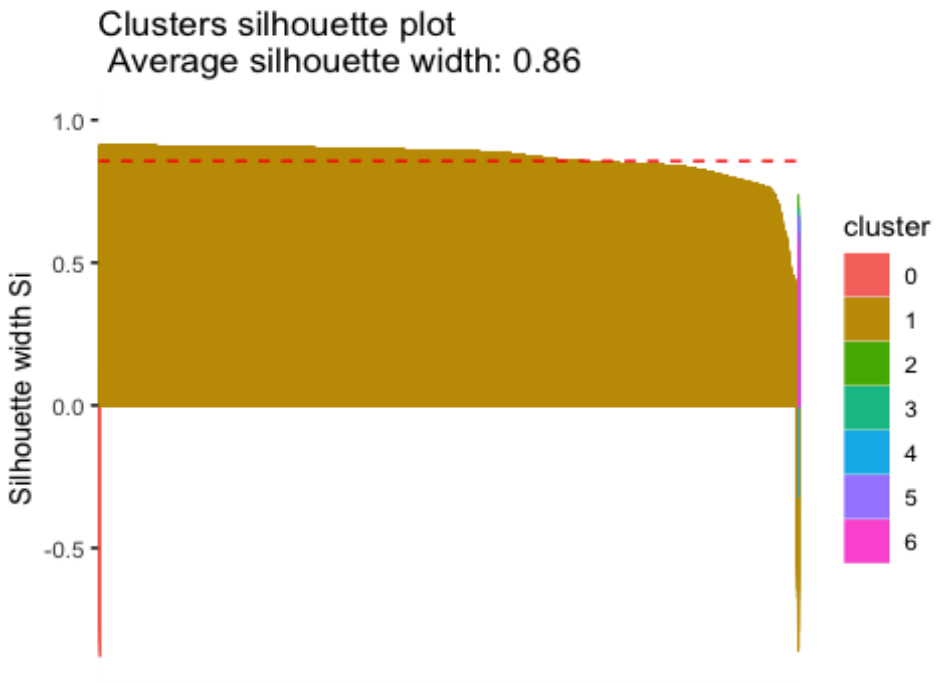
Por las gráficas de abajo vemos que el cluster 0 como era obvio tiene coeficientes negativos ya que se tratan de los outliers y por lo general los coeficientes de los otros clusters han obtenido buenos coeficientes.

```
aux = daisy(transacciones_scan_scale)
sil = silhouette(res_1$cluster, aux)
plot(sil, main = "Silhouette plot - DBSCAN")
```




```
fviz_silhouette(sil)
```

```
##   cluster  size ave.sil.width
## 0       0    49         -0.41
## 1       1 20454          0.86
## 2       2     5          0.66
## 3       3     8          0.54
## 4       4     6          0.42
## 5       5     6          0.55
## 6       6     4          0.48
```



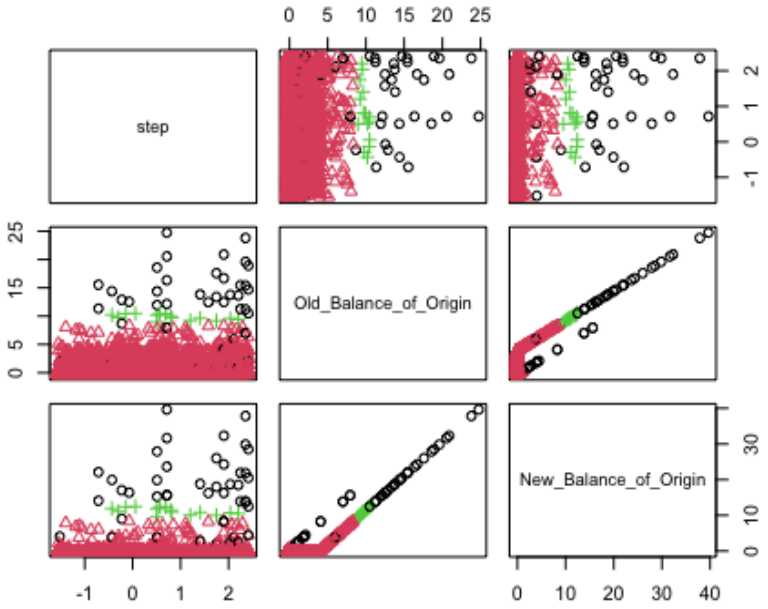
Vemos que obtenido un coeficiente de 0.86.

Qué pasaría si incrementamos el parámetro eps?

```
res_2<- dbscan(transacciones_scan_scale, eps = 1.2, minPts = 5)
res_2

## DBSCAN clustering for 20532 objects.
## Parameters: eps = 1.2, minPts = 5
## The clustering contains 2 cluster(s) and 46 noise points.
##
##     0     1     2
##    46 20471    15
##
## Available fields: cluster, eps, minPts
```

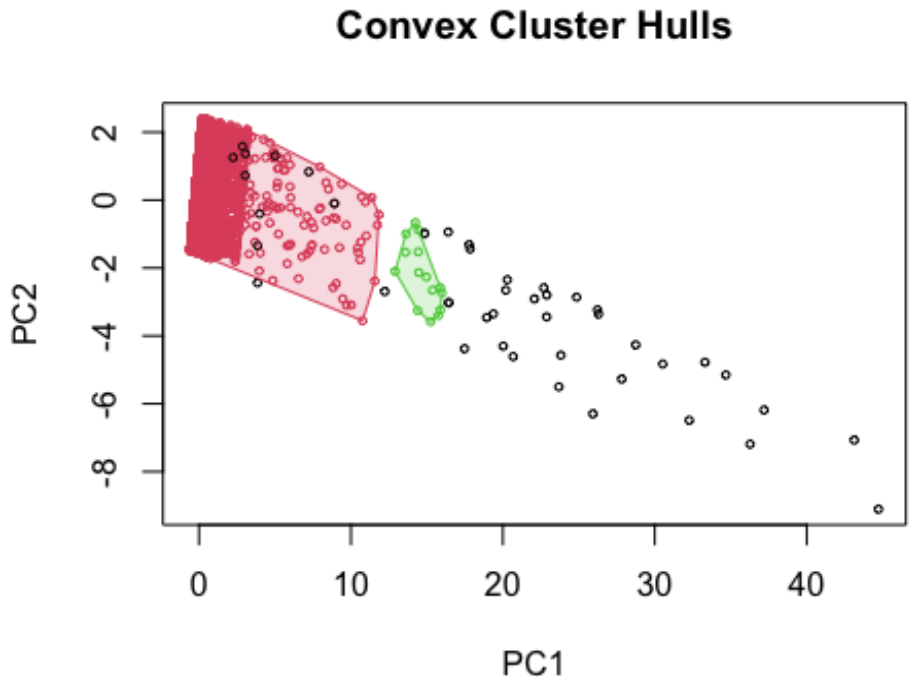
Hemos obtenido 2 clusters y 45 outliers que se encuentran en el cluster 0. Ahora mostramos los clusters con los puntos ruidososo.

```
plot(as.data.frame(transacciones_scan_scale), col = res_2$cluster + 1L,

pch = res_2$cluster + 1L)
```



Seguimos adelante con una representación gráfica que nos muestra los clusters mediante formas convexas.

```
hullplot(transacciones_scan_scale, res_2)
```



Vemos que los clusters

están bien identificados y separados. Ahora mostraremos la matriz de confusión para explicar el resultado de la agrupación.

```
d1=table(res_2$cluster,transacciones_bal$isFraud)
d1
```

```
##
##     No_Fraud Fraud
##   0        1    45
##   1    12318  8153
##   2        0    15
```

En el cluster 0 tenemos los 46 outliers, es decir, casos que no están asignados a ningún cluster. Tenemos 1-2 clusters identificados, en el primer cluster tenemos 12318 transacciones clasificados como No Fraude y 8153 como Fraude. En el otro clusters sólo clasifica como Fraude 15 transacciones.

A partir de la matriz extraeremos el número de casos correctamente clasificados y el porcentaje de precisión.

```
cat("Total de casos correctamente clasificados: 12319+15 =",
    12319+15, "\n")
```

```
## Total de casos correctamente clasificados: 12319+15 = 12334
```

```
cat("Total de casos incorrectamentee clasificados: 8153=",
    8153, "\n")
```

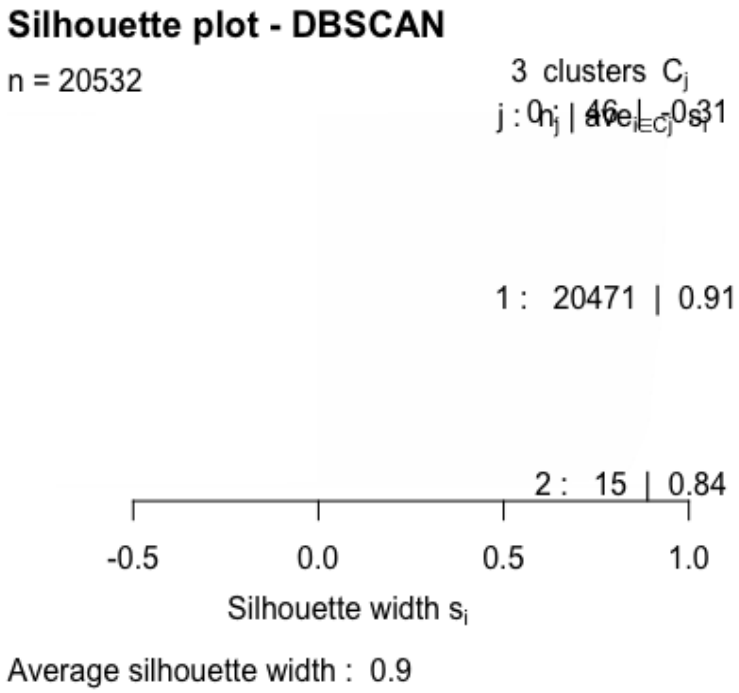
```
## Total de casos incorrectamentee clasificados: 8153= 8153
```

```
cat("Precissió: (12334/(12334+8153))*100= ",
    ( 12334/(12334+8153))*100, "%\n")
```

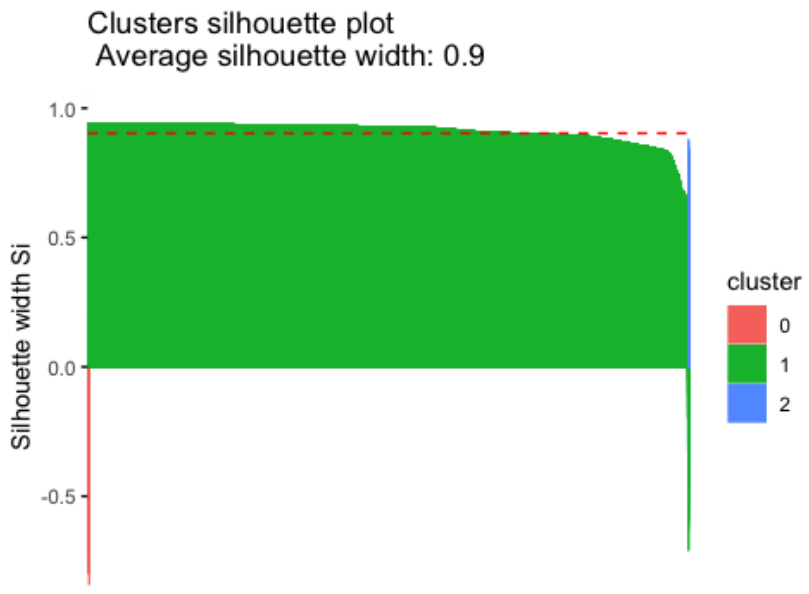
```
## Precissió: (12334/(12334+8153))*100=  60.20403 %
```

Por las gráficas de abajo vemos que el cluster 0 como era obvio tiene coeficientes negativos ya que se tratan de los outliers y por lo general los coeficientes de los otros clusters han obtenido buenos coeficientes.

```
aux = daisy(transacciones_scan_scale)
sil = silhouette(res_2$cluster, aux)
plot(sil, main = "Silhouette plot - DBSCAN")
```




```
fviz_silhouette(sil)

##   cluster  size ave.sil.width
## 0       0    46         -0.31
## 1       1 20471          0.91
## 2       2    15          0.84
```



Vemos que el coeficiente de silhouete por un eps = 1.2 no mejora y es casi el mismo.

Tenemos pues que con el algoritmo DBSCAN por diferentes eps, obteniendo 2 y 6 clusters, siendo la precisión la misma.

El porcentaje está alrededor de 60% y es mejor que el obtenido con k means con distancia euclídea es 79% y peor al modelo obtenido con el algoritmo de aglomeración y distancia de manhattan con 47%.

6. Resultados obtenidos

A continuación, mostraremos una tabla donde veremos los resultados obtenidos por los diferentes algoritmos a lo largo del proyecto.

	Algoritmo			Precisión	Tasa de error
Aprendizaje supervisado	CART (rpart)			97%	
	C5.0			98.7%	
	RF	Tree=500	Mtry=3	98.1%	1.89%
	RF	Tree=500	Mtry=1	91%	8.21%
	RF	Tree=500	Mtry=7	99%	0.97%
Aprendizaje no supervisado	K-Means			79%	
	Clustering Jerarquico			55%	
	DBSCAN	EPS=1		60.1%	
	DBSCAN	EPS=1.2		60.2%	

Cómo podemos ver los algoritmos supervisados tienen una excelente precisión en comparación con los algoritmos no supervisados. De esta manera vemos el papel que juega los datos con etiquetas y los que no.

De los algoritmos supervisados, el que tiene mejor precisión es el **Random Forest** con **Tree = 500 y Mtry = 7.** Podemos afirmar que el tener en cuenta más variables de cara a la división va suponer más precisión en el modelado. No obstante, el coste de tiempo aumentará significativamente.

De los algoritmos no supervisados, el que tiene mejor precisión es el **K-Means**, el cual tiene un 79%. Recordemos que obtuvimos 5 clusters, por el método del Codo.

7. Limitaciones en los modelos supervisados y no supervisados

El **aprendizaje supervisado** es un enfoque de aprendizaje automático que se define mediante el uso de conjuntos de datos etiquetados. Estos conjuntos de datos están diseñados para entrenar o "supervisar" algoritmos para clasificar los datos o predecir los resultados con precisión. Usando entradas y salidas etiquetadas, el modelo puede medir su precisión y aprender con el tiempo.

El aprendizaje supervisado se puede separar en dos tipos de problemas en la minería de datos: ***clasificación y regresión***

El **aprendizaje no supervisado** utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de datos sin etiquetar. Estos algoritmos descubren patrones ocultos en los datos sin necesidad de intervención humana (por tanto, están "sin supervisar").
Los modelos de aprendizaje no supervisado se utilizan para tres tareas principales: ***agrupación, asociación y reducción de la dimensionalidad.***

La principal diferencia es el uso de conjuntos de datos. Por decirlo simplemente, el aprendizaje supervisado utiliza datos de entrada y salida etiquetados, mientras que un algoritmo de aprendizaje no supervisado no.

En nuestro caso tenemos un conjunto de datos referente a las transacciones bancarias. Por cada fila es un data point, es decir, cada fila representa una transacción diferente que está definido por el conjunto de características como el balance de origen, la cantidad, el balance de destino, etc. Las características son las columnas del juego de datos. Para hablar de etiquetas, deberemos contextualizar nuestro problema. Si predecimos una feature o columna basada en las demás, entonces la etiqueta es la feature o columna. Si predecimos si nuestra transacción es fraude o no (en nuestro caso), basándonos en la información de la transacción, entonces ésta es la etiqueta.

Por tanto, nuestro juego de datos esta etiquetado ya que queremos predecir si la transacción es fraudulenta o no a partir del conjunto de features. Por tanto nuestro juego de datos está orientado al aprendizaje supervisado. Imaginemos que tenemos una nueva transacción es decir no sabemos si es fraudulenta o no, el modelo entonces haría una predicción si esta transacción es fraudulenta o no.

Supongamos que nuestro data set no está etiquetado, entonces nuestro modelo todavía podría decirnos si dos data points son similares entre ellos o diferentes. Los podría agrupar por similitud, aunque no sabemos que representa a cada grupo.

En nuestro data set no hay demasiadas limitaciones. Por un lado, la dimensionalidad es pequeña con 11 variables y alrededor de 6 millones de filas.

Estos son los **retos a los que se enfrenta el aprendizaje automático supervisado**:

- Los datos de entrenamiento irrelevantes de la función de entrada pueden dar resultados inexactos
- La preparación y el preprocesamiento de datos es siempre un reto
- La precisión sufre cuando se han introducido valores imposibles, improbables e incompletos cómo datos de entrenamiento
- Si el experto en cuestión no está disponible, el otro enfoque es “fuerza bruta”. Significa que debemos pensar las featuras adecuadas (variables de entrada) para entrenar la máquina. Puede ser inexacto.
- Clasificar gran cantidad de datos puede ser un gran reto
- Entrenar el modelo supervisado necesita tiempo computacional

Estos son los **retos a los que se enfrenta el aprendizaje automático no supervisado**:

- No podemos obtener información precisa sobre la ordenación de datos, y la salida como datos utilizados en el aprendizaje no supervisado está etiquetada y no se conoce.
- Menos precisión de los resultados se debe a que los datos de entrada no son conocidas y no etiquetados por la gente de antemano. Esto significa que la máquina debe hacerlo ella misma.
- Las clases no siempre corresponden a clases informativas.
- El usuario debe dedicar tiempo a interpretar y etiquetar las clases que siguen esta clasificación.

8. Líneas de trabajo futuras

Una vez concluido el proyecto se ha considerado que se podrían realizar las siguientes acciones para mejorarlo y/o seguir con su desarrollo:

- Mejorar la calidad de los datos. Una buena ETL siempre facilitará el trabajo posterior.

- El dataset debería ser más extenso. No solo el número de registros es importante sino obviamente las variables que hay en juego. Harían falta muchas más tablas con todo tipo de información.

- Aplicar técnicas de NLP para analizar textos, y poder así extraer más información
- Investigar configuraciones de hyperparámetros no utilizadas en el desarrollo del proyecto.
- Datos históricos para poder predecir con más exactitud en función del comportamiento pasado

Conclusión

Tras la realización del presente proyecto se han obtenido las siguientes conclusiones:

- Desde los atentados de 9/11 en Nueva York, se han impuesto medidas y leyes para prevenir el blanqueo de capitales o el financiamiento del terrorismo.
- Pese a haber muchas medidas de compliance, hay bancos que aún no cumplen con lo establecido por las organizaciones y por lo tanto son sancionados.
- La era tecnología para bien o para mal ha contribuido a que haya más delitos financieros (compras online, bizum, etc.)
- Las nuevas tecnologías han ayudado mucho a las instituciones financieras a poder detectar fraudes o delitos que puedan tener un impacto negativo económico y social.
- No obstante, el tiempo que invierten los analistas o investigadores en hacer una due diligence en posibles casos de fraude es demasiado grande. Además existen los falsos positivos, que pueden provocar un coste recursos enorme.
- Hay empresas que ya están a la vanguardia del software AML. Hoy en día estos softwares utilizan técnicas de machine learning para no solo reducir el coste de tiempo o recursos en analizar alertas que sean falsos positivos.
- Aún queda trabajo por hacer en esta industria, pues la introducción del bitcoins y la tecnología BlockChain está en auge y podría ser un canal del cual muchos infractores puedan beneficiarse.
- El mundo AML es bastante grande, puesto que no es suficiente con analizar transacciones sino analizar y hacer perfiles de clientes, por ejemplo.

Bibliografía

[1] HM Treasury, "Anti-money laundering strategy," Oct. 2004
Consulta: Marzo 2023

[2] Shijia Gao, Dongming Xu, Huaiqing Wang, and Yingfeng Wang. Intelligent Anti-Money Laundering System.
Consulta: marzo 2023

[3] Raúl Montoliu Colás, " **Models Supervisats**", Universitat Oberta de Catalunya, Barcelona, septiembre, 2021.
Consulta: marzo 2023

[4] Jordi Gironés Roig, " **Models no Supervisats**", Universitat Oberta de Catalunya, Barcelona, septiembre, 2021.
Consulta: marzo 2023

[5] Julià Minguillón Alfonso, Ramon Caihuelas Quiles, "**Procés de minería de dades",** Universitat Oberta de Catalunya, Barcelona, septiembre, 2021.
Consulta: marzo 2023

[6] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, CRISP-DM 1.0 Step-bystep data mining guide. IBM, Aug. 2000. [Online].
Consulta: marzo 2023

[7] https://www.undp.org/
Consulta: marzo 2023

[8] Peter Reuter and Edwin Truman, Chasing Dirty Money: The Fight Against Money Laundering (Institute for International Economics, 2004)
Consulta: marzo 2023

[9] International Federation of Accountants, "Anti-money laundering," Jan. 2002.
Consulta: marzo 2023

[10] R. C. Watkins, Exploring Data Mining technologies as Tool to Investigate Money Laundering. Journal of Policing Practice and Research.
Consulta: marzo 2023

[11] J. Tang, J. Yin, developing an intelligent data discriminating system of anti-money laundering based on SVM.

Consulta: marzo 2023

[12] Z. Zang, J.J. Salermo and P. S. Yu, Applying Data mining in Investigating Money Laundering Crimes.
Consulta: marzo 2023

[**13]** B. Scholkopf, A short tutorial on kernels. Microsoft Research.
Consulta: marzo 2023

[14] B. Scholkopf and J. Plattz, Estimating the support of a high dimensional distribution, Neural Computing.
Consulta: marzo 2023

[15] D.R Wilson and T. R. Martinez, Improved Heterogeneous distance functions. Journal of Artificial Intelligence Research.
Consulta: marzo 2023

[16] R. Jain, R. Kasturi and B.G. Schunck, Machine Vision, Prentice Hall, 1995
Consulta: marzo 2023

[17] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature.
Consulta: marzo 2023

[18] Han, Kamber, and Pei. Data Mining Concepts and Techniques.
Consulta: marzo 2023

[19] Paula, E.L, Ladeira, M., Carvalho, R.N., & Marzagão, T. (2016). Deep learning anomaly detection as support fraud investigation in Brazilian exports and anti-money laundering. Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA)

[20] J. Kingdon. AI Fights Money Laundering, IEEE Transactions on Intelligent Systems, 2004,

[21] T. Wicks. "Intelligent systems for money laundering prevention," Money Laundering Bulletin, Sep. 2001.

[22] I. Horobin. "Applying technology to fight money laundering," Money Laundering Bulletin, Apr. 2001.
Consulta: marzo 2023

[**23]** ingguang Han, Yuyun Huang, Sha Liu, Kieran Towey. Artifcial intelligence for anti-money laundering: a review and extension J
Consulta: marzo 2023

[24] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval
Consulta: marzo 2023

[25] Zhiyuan Chen, Le Dinh Van Khoa, Ee Na Teoh, Amril Nazir, Ettikan Kandasamy Karuppiah, Kim Sim Lam. Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review.
Consulta: marzo 2023

[26] Major AML & Sanctions Bank Fines & Penalties in the 21st Century.
https://thefinancialcrimenews.com/
Consulta: Mayo 2023

[27] PAYSIM: A FINANCIAL MOBILE MONEY SIMULATOR FOR FRAUD DETECTION Edgar Alonso, Ahmad Elmir y Stefan Axelsson
Consulta: Mayo 2023