Universidad de la República
Facultad de Ingeniería
Ingeniería en Computación

Detección de Patrones de Comportamiento Regulares y Fraudulentos Sobre un Conjunto de Transacciones Financieras

Proyecto de Grado

Integrantes: Lucía Adinolfi, Martín Ferreira, Lucía Ramos Muzio

Tutores: Msc. Diego Garat, Msc. Guillermo Moncecchi

Montevideo, Uruguay
Junio 2009

Resumen

Las entidades financieras de medios de pago sufren tanto pérdidas financieras como deterioro de imagen frente a sus clientes por fraude. Para reducir estas pérdidas, se busca detectar la actividad fraudulenta lo más pronto posible, destinándose personal especializado y sistemas informáticos de soporte para este fin. Generalmente estos sistemas son configurados basándose en el supuesto de que las transacciones fraudulentas siguen patrones particulares que difieren de las transacciones usuales, lo que permite su detección.

Entre los sistemas de detección de fraude se destacan aquellos basados en reglas, que deben ser configurados por especialistas, cuya principal debilidad es que suelen perder su efectividad con el tiempo a medida que se evoluciona en nuevos patrones de fraude. Por otro lado, existen sistemas completamente automatizados, capaces de aprender nuevos patrones a partir de transacciones clasificadas como legítimas o fraudulentas, cuya principal debilidad es que su aprendizaje no siempre es fácilmente interpretable por un especialista que pueda aprovechar su conocimiento para ser utilizado en otras formas de prevención fuera del alcance del sistema.

A partir de esta problemática, buscamos analizar y utilizar mecanismos de aprendizaje automático que permitan mejorar y mantener los sistemas de detección de fraude. Más específicamente utilizamos métodos de selección de atributos para poder entrenar modelos de forma eficiente; evaluamos métodos que permitan generar reglas que sean interpretables por especialistas y puedan ser incorporados en sistemas de detección basados en reglas y por último, analizamos métodos que generen modelos capaces de retornar un puntaje que determine el riesgo de que una transacción resulte fraudulenta.

Como resultado obtuvimos una herramienta de software que permite, a partir de un conjunto de transacciones clasificadas, entrenar modelos de árboles de decisión, redes neuronales y random forest. Esto incluye tomar transacciones de la base de datos, realizar las transformaciones necesarias y aplicar algoritmos de selección de atributos. A partir de los modelos generados es posible tanto obtener un conjunto de reglas como evaluar transacciones obteniendo puntajes de riesgo para cada transacción.

Palabras clave: detección de fraude, medios de pago, aprendizaje automático, árboles de decisión, redes neuronales, *random forest*, clasificación.

Índice general

Capítulo 1

Introducción

En la actualidad, el uso de medios de pago y en especial de la tarjeta de crédito como sustituto del dinero en efectivo se ha convertido no solo en un estilo de vida, sino en una necesidad clave para la economía de consumo.

Una problemática recurrente en el negocio de los medios de pago es el fraude. Sea tanto en los pagos realizados a través de Internet, como utilizando una tarjeta de banda magnética o chip, las organizaciones emisoras de medios de pago deben afrontar pérdidas financieras importantes debidas a fraude (transacciones realizadas en forma ilegal, que el emisor debe pagar pero no puede cobrar pues son repudiadas por el responsable del medio de pago) [23]. Una vez que los emisores de medios de pago establecen medidas de control y detección de fraude, las formas de realizar fraude evolucionan también con el fin de sobrepasar estos controles, generando así nuevos patrones de transacciones fraudulentas.

En general, el monitoreo de transacciones fraudulentas es realizado conjuntamente por una herramienta de software que genera alertas al detectar transacciones que se ajustan a ciertos perfiles de fraude, y un conjunto de analistas de riesgo que decide cuáles de esas alertas merecen ser investigadas y cuáles no. Lógicamente, es deseable que la efectividad (cuánto fraude del total real es detectado por el sistema) sea lo más alta posible y el falso positivo (cantidad de alertas que finalmente no resultan en fraude real) sea lo más bajo posible.

PayTrue solutions [38], cliente del proyecto, es una empresa de software dedicada al diseño de soluciones integrales para el negocio de medios de pago. Actualmente cuenta con un conjunto de soluciones específicamente diseñadas para el monitoreo de transacciones financieras y no financieras con el objetivo de detectar patrones predefinidos considerados como fraudulentos. La aparición de uno de estos patrones dentro del sistema deriva en la generación de una alerta, la cual es posteriormente analizada por un equipo de personas especializadas que determinarán si esta alerta es realmente un fraude o no.

El conjunto de productos orientado a detectar, prevenir, analizar y contener el fraude que ofrece *PayTrue solutions* se conoce como *Risk Center* y éste contiene un módulo llamado *Detection Engine* el cual se encarga de la evaluación de la transacción a medida que es recibida por la entidad financiera. El módulo

funciona en tres etapas: evaluación de reglas, cálculo de la probabilidad de que la transacción sea fraudulenta (puntaje de riesgo) y finalmente la comparación del puntaje calculado contra el umbral de activación definido. Si el puntaje de riesgo supera el umbral, entonces se genera una alerta. La configuración de las etapas mencionadas se realiza manualmente por un analista de riesgo.

Dado que la configuración de los productos de *PayTrue solutions* se realiza de forma manual, la efectividad y el falso positivo de las alertas generadas dependen en gran medida del talento y conocimiento de mercado que los analistas de negocio tengan en cuanto a la problemática de fraude que atraviesan y de un conjunto de reportes predefinidos que evidencian tendencias generales de fraude.

El objetivo de *PayTrue solutions* es incorporar un módulo adicional, denominado *Smart Analyzer*, capaz de sugerir configuraciones útiles al conjunto de analistas de riesgo, por ejemplo, cuáles reglas incorporar en la primera etapa. *PayTrue solutions* desea mejorar la detección de fraudes reales del sistema para incrementar la competitividad de su producto. Para lograr este último punto debe mejorar la precisión en la asignación del puntaje de riesgo a cada transacción.

El Aprendizaje Automático, disciplina que trata de programas capaces de aprender a partir de un conjunto de ejemplos, ha resultado ser de gran utilidad en diversas tareas, dentro de las cuales podemos distinguir tres grandes grupos: tareas difíciles de programar, aplicaciones auto adaptables y minería de datos. Por sus características, la tarea de detección de fraude pertenece a los dos últimos grupos, ya que para su correcta resolución es necesario desarrollar aplicaciones capaces de extraer conocimiento a partir de grandes conjuntos de ejemplos para descubrir patrones complejos y adaptarse a nuevos ejemplos. Por esta razón la detección de fraude ha sido ampliamente estudiada por esta disciplina [42].

El problema de detección de fraude mediante clasificación de una transacción de crédito es un tema de interés en el campo de Aprendizaje Automático [48]. Si bien existen varias empresas a nivel global dedicadas al desarrollo de software para detección de fraude, como por ejemplo FairIsacc con su sistema Falcon [13], las características de sus soluciones no son develadas. Sin embargo, es de conocimiento público que entre las técnicas utilizadas para obtener sus resultados se encuentran el uso de árboles de decisión, redes neuronales y regresión estadística entre otros.

Este proyecto busca entonces expandir las posibilidades del *Smart Analyzer* a través de la automatización de la configuración de reglas para obtener mayor efectividad y en consecuencia aumentar la detección de fraude.

1.1. Antecedentes

En el ámbito académico, el antecedente más directo de este proyecto es el trabajo realizado en **Análisis y Detección de Patrones de Fraude en Medios de Pago** [11], también impulsado por *PayTrue solutions*. En ese proyecto estudiaron las características del negocio de medios de pago y de patrones de fraude comunes en este negocio, propusieron métricas que permitan evaluar el

desempeño de una configuración del *Detection Engine*, definieron un enfoque para resolver el problema de automatización de la configuración del *Detection Engine* e implementaron un prototipo del enfoque elegido. Para el aprendizaje de reglas utilizaron árboles de decisión, en particular los programas `C4.5` y `QUEST`. Para el aprendizaje de *score* utilizaron un enfoque *Naive Bayesian*. Los resultados que obtuvieron con el algoritmo `C4.5` superaron los obtenidos con `QUEST` alcanzando con el `C4.5` el doble de precisión y recuperación que con el `QUEST`. Con el modelo final obtuvieron mejores resultados que utilizando un clasificador trivial pero concluyeron que la aplicación de *naive bayesian* al cálculo de puntaje de riesgo no resultó lo suficientemente provechosa.

Dentro de los resultados obtenidos se identificaron tres grandes carencias:

- Problemas para procesar los volúmenes de datos necesarios. Las implementaciones que utilizaron de algoritmos de árboles de decisión funcionan cargando el conjunto completo de ejemplos en memoria y permiten procesar conjuntos pequeños de autorizaciones.
- Imposibilidad de sugerir variables complejas (variables calculadas aplicando una función de agregación o variables que representan las tendencias en las transacciones cercanas en el tiempo). Estas variables pueden resultar de gran importancia en la detección de fraude.
- No se logra una mejora significativa en la detección de fraude con el algoritmo de asignación del puntaje de riesgo utilizado (*Naive Bayes*) respecto a la detección alcanzada en la etapa de aprendizaje de reglas.

1.2. Objetivos

El objetivo de este proyecto es lograr un sistema capaz de identificar patrones de fraude en las transacciones de crédito para mejorar la efectividad del sistema actual de *PayTrue solutions*.

Dentro de los objetivos específicos del proyecto se encuentra el analizar y utilizar algoritmos de aprendizaje automático para la generación de reglas y cálculo del puntaje de riesgo a partir de un conjunto de transacciones clasificadas como fraudulentas o legítimas. También es de interés determinar cuáles atributos son relevantes para la detección de fraude y en consecuencia merecen la pena calcularse a medida que llegan las transacciones dado el costo computacional que implica su cálculo y el costo de su almacenamiento. Para lograr los objetivos se dividimos el trabajo en tres tareas principales que se describen a continuación.

La primera tarea consiste en analizar y utilizar métodos de selección de atributos, utilizando como entrada los atributos disponibles en las transacciones y atributos complejos. Para esto es necesario definir un gran número de atributos calculados mediante funciones de agregación sobre los atributos de la transacción.

La segunda tarea es evaluar métodos de clasificación para la generación de reglas, que reciben como entrada distintos conjuntos de atributos seleccionados

en la etapa previa.

Como última tarea, debemos obtener funciones de evaluación para el cálculo de puntaje de riesgo (*scorer*) las cuales operan sobre los conjuntos de atributos calculados mencionados anteriormente.

1.3. Contribución

Contribuimos en todas las tareas realizadas para llevar a cabo el proyecto, trabajando en el marco de aprendizaje automático para alcanzar el objetivo en cada una de ellas.

En primer lugar, realizamos un estudio sobre *150* atributos calculados para determinar su relevancia ya que no encontramos antecedentes que indicasen cuáles atributos tomar en cuenta ni cuáles funciones de agregación calcular sobre ellos. Para lograr este fin, utilizamos algoritmos de selección de atributos.

En segundo lugar, construimos un conjunto de clasificadores con algoritmos de árboles de decisión, con el objetivo de obtener las reglas necesarias para la detección de fraude.

Finalmente, obtenemos dos *scorers* que intentan resolver la asignación de un puntaje de riesgo asociado a cada transacción para lo cual utilizamos algoritmos de redes neuronales y *random forests*.

Los tres puntos anteriores se encuentran capturados en una herramienta que permite ejecutar y visualizar los algoritmos estudiados así como clasificar nuevas transacciones. Esto incluye tomar transacciones de la base de datos, realizar las transformaciones necesarias, aplicar algoritmos de selección de atributos y entrenar modelos de árboles de decisión, redes neuronales y *random forest*.

1.4. Organización del documento

Este documento está organizado de la siguiente manera: en el capítulo 2 realizamos una introducción a los medios de pago y la problemática del fraude en el contexto de este negocio. En los capítulos 3, 4 y 5 detallamos las etapas fundamentales del proyecto: en el capítulo 3 introducimos el marco teórico de la generación de reglas utilizando árboles de decisión y de la función de evaluación buscada para medir el puntaje de riesgo de las transacciones utilizando redes neuronales y *random forests*, también introducimos las medidas utilizadas para evaluar y comparar los algoritmos utilizados; en el capítulo 4 analizamos el conjunto de datos de entrada y explicamos las modificaciones realizadas mediante el cálculo y selección de atributos; en el capítulo 5 nos enfocamos en las pruebas de los distintos algoritmos y en los resultados que con ellos obtuvimos. En el capítulo 6 presentamos la aplicación desarrollada a lo largo del proyecto y por último, en el capítulo 7 desarrollamos las conclusiones obtenidas.

Capítulo 2

Los medios de pago y la problemática del fraude

Este capítulo tiene como propósito describir el negocio de los medios de pago y la problemática del fraude. En la sección 2.1 describimos la mecánica del negocio incluyendo los actores principales y su interacción. En la sección 2.2 describimos las características y modalidades del fraude en medios de pago. Por último, en la sección 2.3 detallamos la suite de productos actual de *PayTrue solutions* y la interacción con la herramienta a desarrollar.

2.1. Mecánica del negocio

En un ciclo de negocio habitual un sujeto realiza una transacción en un comercio (*card acceptor*) para obtener un bien o servicio. El ciclo involucra a instituciones financieras (emisores) que son las encargadas de otorgar la tarjeta al solicitante (tarjetahabiente) y a otras instituciones (adquirentes) que establecen diferentes acuerdos con los comercios para que el tarjetahabiente pueda realizar sus transacciones; aparecen también otras figuras que interactuarán entre los emisores y los adquirentes brindándoles servicios de procesamiento (procesadoras). En el caso de transacciones internacionales aparecen las marcas internacionales como Visa o MasterCard que mediante la definición de estándares y protocolos permiten la interacción de tarjetahabientes y comercios de distintos países.

Para iniciar el ciclo, el comerciante solicita una autorización brindando varios datos de la tarjeta, del comercio y de la compra incluyendo número de tarjeta, fecha de vencimiento de la tarjeta, código del comercio, tipo de negocio o rubro del producto o servicio (conocido como *MCC* por sus siglas en inglés, *Merchant Category Code*) y monto de la transacción. La autorización no tiene efecto financiero y debe ser aprobada tanto por el adquirente como por el emisor. El emisor autoriza la transacción si los datos de la tarjeta son correctos y la tarjeta tiene dinero disponible para cubrir el monto de la transacción. El adquirente autoriza la transacción si los datos del comercio son correctos y el comercio está habilitado para realizar la transacción solicitada. Luego se efectúa la venta o confirmación de la transacción que sí tiene efecto financiero. Es posible que

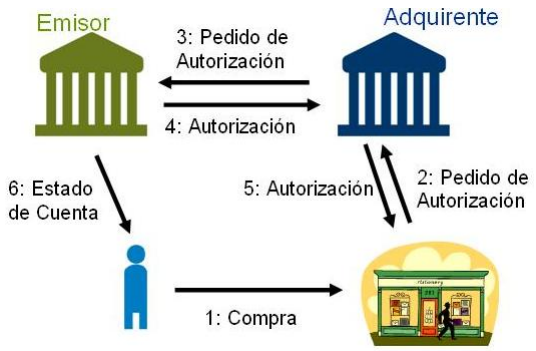

Figura 2.1: Ciclo de Negocio

una autorización nunca sea confirmada o que una confirmación no tenga una autorización previa. Para el intercambio de transacciones electrónicas de tarjetas de crédito se utiliza el estándar **ISO 8583** [25] (referirse al Apéndice A - ISO 8583 por una descripción del estándar).

Es posible que en un mercado existan varios emisores y adquirentes permitiendo que un comercio esté afiliado a más de un adquirente. Cuando un comercio acepta un cierto medio de pago, significa que alguno de los adquirentes a los que está afiliado tiene un acuerdo comercial con el emisor o con la marca internacional de dicho medio de pago. Algunos de los factores que afectan el negocio de los medios de pago y las relaciones entre los actores que intervienen son el tipo de contrato entre el emisor y el tarjetahabiente (producto) y el soporte físico del medio de pago (tarjeta). El producto determina el tipo de transacción que el tarjetahabiente puede realizar e incluye el tipo de tarjeta.

La tarjeta puede ser de crédito, débito y/o prepago. En el caso de las tarjetas de crédito los tarjetahabientes tienen límites con respecto a la cantidad que pueden cargar. Cada mes el tarjetahabiente puede pagar su saldo por completo o pagar solo una parte. El emisor establece el pago mínimo y los cargos de financiamiento para el saldo pendiente. Las tarjetas de crédito también pueden usarse en los cajeros automáticos o en un banco para servirse de un adelanto de efectivo aunque, a diferencia de las tarjetas de débito, se cobra un interés al tarjetahabiente.

En el caso de las tarjetas de débito el dinero que se usa no se toma a crédito sino que se debita directamente del que se disponga en la cuenta bancaria asociada a la tarjeta. Algunos emisores realizan acuerdos con sus clientes para permitirles extraer dinero en descubierto, generando un préstamo con sus respectivos intereses. Su cuota anual es más barata que la de crédito o incluso gratis.

En las tarjetas de prepago se anticipa el importe del consumo que se realizará con la tarjeta. Se efectúa una carga de dinero en la tarjeta y pueden realizarse operaciones hasta consumir el importe cargado.

La forma física que toma el medio de pago puede ser una tarjeta plástica de banda magnética, una tarjeta plástica con chip (conocidas como tarjetas EMV [12]), un teléfono celular u otros. Los distintos soportes físicos incluyen diferentes mecanismos para salvaguardar la información que contienen, por lo que algunos son más riesgosos que otros respecto al fraude.

2.2. Fraude en los medios de pago

El fraude se define en [39] como actividad no autorizada de una cuenta por una persona para quien la cuenta no fue creada. En la práctica son transacciones que son repudiadas por el tarjetahabiente y cuyo costo debe ser asumido por alguno de los actores que intervienen en el negocio de los medios de pago. Existe una intención de parte de todos los actores involucrados de tomar acciones e incorporar prácticas de administración de riesgo para prevenir y disminuir el progreso del fraude.

El fraude puede ser cometido por el tarjetahabiente, un comercio, un empleado de un emisor, un empleado de un adquirente o un tercero. Para que se produzca un fraude tiene que haber existido previamente un compromiso de la información de la tarjeta, que puede haber sido cometido o no por el mismo defraudador.

El fraude en tarjetas de crédito presenta las siguientes características [54, 1, 11]:

- El volumen y comportamiento de transacciones legales y fraudulentas varía dependiendo del contexto y en el tiempo.
- El fraude es significativamente menor a las ventas legítimas (en el entorno del 0,1 %).
- Se pueden observar múltiples patrones de fraude de distintas características simultáneamente. Luego de que se reconoce un patrón, los defraudadores modifican su comportamiento causando una evolución constante en los patrones de fraude.
- El reconocimiento de una transacción como fraude demora normalmente entre uno y tres meses.
- No todo el fraude es detectado y no todas las transacciones denunciadas como fraude lo son realmente.
- Existe una visión parcial de los datos. En general, los adquirentes sólo conocen aquellas transacciones realizadas en sus comercios, simétricamente, los emisores sólo conocen aquellas transacciones realizadas por sus tarjetahabientes. A su vez, los datos conocidos de cada transacción son los intercambiados con el protocolo ISO 8583.

2.2.1. Modalidades de fraude

El tipo de fraude que se puede realizar depende de la información que tenga el defraudador. Por ejemplo, si tiene una tarjeta física con la cual ir a un comercio

o si sólo tiene el número de tarjeta y fecha de vencimiento con lo cual basta para hacer una compra por Internet.

Existen diversas modalidades que emplean los defraudadores para obtener información de cuentas. A continuación detallamos las principales.

***Skimming*:** Cualquier situación en la cual se capturan datos de cuentas válidas, generalmente se trata de la copia de la información de la banda magnética de una tarjeta.

***Phishing*:** Se utiliza ingeniería social para obtener información confidencial como puede ser una contraseña o datos de una tarjeta de crédito. El defraudador, conocido como *phisher*, se comunica con un tarjetahabiente haciéndose pasar por un actor válido (por ejemplo un emisor o un adquirente) y lo convence de que le proporcione la información.

Intercepción de correo: Ocurre cuando el correo que contiene datos de una tarjeta válida es interceptado por un defraudador.

Robo o pérdida de tarjetas: El defraudador obtiene mediante un robo o pérdida una tarjeta válida. Es el modo de obtención de información más común.

Datos comprometidos por diversas fallas de seguridad: Incluye intrusiones en los sistemas informáticos de la entidad financiera, pérdida de paquetes con tarjetas a entregar en la empresa de transporte, acceso en forma indebida a archivos con datos de tarjetas, etc.

Existen diversas modalidades para cometer fraude con la información obtenida. A continuación detallamos las principales.

Solicitudes de tarjeta con datos falsos: Consiste en llenar una solicitud de tarjeta de crédito con datos falsos o datos válidos obtenidos previamente. Una vez obtenida la tarjeta, el defraudador la utiliza como si fuera el propietario legítimo.

Cambio de dirección: Solicitud de cambio de dirección haciéndose pasar por un tarjetahabiente, para luego solicitar una reimpresión de la tarjeta o emisión de una tarjeta adicional, que será enviada a la nueva dirección.

Autofraude: Se comete fraude con la propia cuenta del defraudador.

Clonación de tarjetas: Se crea una tarjeta con un número de tarjeta válido. Se puede codificar la información en una tarjeta de banda magnética en blanco o cambiar una robada.

Falsificación manual de tarjetas: Se crea un número de tarjeta válido usando un software que los genere. Se puede codificar la información en una tarjeta de banda magnética en blanco o cambiar una robada.

Prueba de cuenta: El defraudador pide autorización por montos bajos para probar que una tarjeta robada, clonada o falsa está activa y poder realizar luego compras hasta agotar el disponible.

Fraude con tarjeta no presente: Escenario conocido como *MOTO* (por las siglas en inglés de *mail order / telephone order*) en donde el comercio no puede ver la tarjeta. Se realizan transacciones con los datos de una tarjeta, sin necesidad de tener la tarjeta físicamente, por ejemplo, por teléfono, por mail o por Internet.

Comercio Fraudulento: Un comercio aparentemente legítimo que abre una cuenta válida con un adquirente y después de un período de actividad de venta normal deposita un gran número de transacciones fraudulentas por altos montos. Una vez recibido el pago de las transacciones el comercio desaparece. Los comercios fraudulentos con frecuencia presentan solicitudes de afiliación a varios adquirentes al mismo tiempo.

Existen algunos patrones de riesgo conocidos que pueden indicar que se trata de una actividad fraudulenta. Hay patrones que involucran parámetros de alto riesgo en la transacción como ser escenario de tarjeta no presente, modo de captura de los datos inconsistente con la capacidad de la terminal, indicador de entrega de dinero en efectivo, transacciones realizadas en comercios de *MCC* de fácil reventa (como ser joyerías, productos electrónicos, artículos de lujo), transacciones en el extranjero o transacciones por montos elevados.

Otros patrones involucran la repetición en cortos períodos de tiempo: transacciones hechas con una cuenta en distintos países; múltiples intentos de obtener autorizaciones en un comercio con tarjetas de un mismo emisor, caso conocido como ensayo de BIN (Bank Identification Number); varias transacciones rechazadas en un comercio por una única razón —número de cuenta no existente, fecha de vencimiento errónea, carencia de disponible—; varias transacciones de una tarjeta rechazadas por no coincidir el código de verificación; múltiples transacciones de una tarjeta por montos similares o realizadas en comercios de MCC en los que usualmente no hay repetición –restaurantes, hoteles, aerolíneas—.

Algunos patrones implican la desviación del comportamiento habitual como ser transacciones por montos superiores al promedio del comercio o la tarjeta, pagos fuera de las fechas normales o retiros en efectivo excesivos.

2.2.2. Control de fraude

Aunque se han desarrollado distintos métodos de detección y prevención de fraude, el control de fraude se realiza de formas muy variadas. Algunas entidades financieras (en general muy pequeñas) no dedican ningún esfuerzo a monitorear o prevenir el fraude. Otras hacen importantes inversiones. A continuación listamos algunas de las formas de detección y control de fraude [3]:

Sistema de verificación de direcciones: consiste en verificar que la dirección a la que se envían los bienes adquiridos coincida o se encuentre dentro de un rango aceptable respecto a la dirección donde se está enviando el estado de cuenta de la tarjeta.

Métodos de verificación de tarjetas: se trata de la verificación de un número de tres o cuatro dígitos impreso en la tarjeta, que no forma parte de la información que contiene la banda magnética.

Listas positivas y negativas: información almacenada en bases de datos donde se califican ciertas características como positivas o negativas para detectar transacciones de alto riesgo, un ejemplo de esto es la clasificación de los países según el volumen de fraudes cometidos, pero también se aplica a comercios, rangos de direcciones IP en compras por Internet, etc.

Autenticación del cliente: se basa en el concepto de los cajeros automáticos y el PIN (*Personal Identification Number*), el cual no está impreso ni codificado en la banda magnética de la tarjeta.

Biométricas: esta es una técnica que se apoya en una característica única del titular de la tarjeta, como por ejemplo la huella dactilar, de manera de tener la certeza de que quien está en poder de la tarjeta es efectivamente su propietario.

Monitoreo de comportamiento transaccional: consiste en realizar un seguimiento de las transacciones para detectar comportamientos inusuales o riesgosos para predecir transacciones fraudulentas. Existen variadas implementaciones, por ejemplo, sistemas de reglas, sistemas de votación, redes neuronales o revisiones manuales, en el cuadro 2.1 describimos estas implementaciones.

2.3. *Risk Center*

El *Risk Center* es un conjunto de soluciones desarrolladas por *PayTrue solutions* orientado a detectar, prevenir, analizar y contener el fraude en cualquier sistema de medios de pago. Para ello cuenta con un sistema integral e integrador que se integra e interactúa con otros sistemas existentes [38]. Actualmente el *Risk Center* engloba los sistemas *Advanced Console*, *CPP Finder*, *Suspicious Activity*, *Statistical Manager* y el *Detection Engine*. Se desea incorporar a los sistemas ya existentes el *Smart Analyzer*.

CPP Finder: Herramienta de detección e investigación sobre comercios donde se comprometió la información de una tarjeta.

Suspicious Activity: Monitorea las actividades de fraude sospechosas o excesivas de comercios.

Statistical Manager: Es un módulo de reportes estadísticos para el análisis del comportamiento de la actividad fraudulenta. Permite asistir a los analistas de riesgo en la comprensión de las situaciones de fraudes a partir de una herramienta de reportes que permite navegar sobre los datos.

Advanced Console: Consola de análisis de riesgo, que soporta alertas de múltiples fuentes, emisor y adquirente.

Detection Engine: Solución de monitoreo y generación de alertas para transacciones, emisor y adquirente. Al *Detection Engine* llegan todas las autorizaciones en forma inmediata o casi inmediata (*online*). Las transacciones recibidas son evaluadas y la herramienta decide si se genera o no una alerta asociada a la transacción. La decisión se toma a partir de un conjunto de reglas, un puntaje de riesgo y un umbral. Mostramos el proceso de

Sistema de reglas: es un sistema de reglas del estilo *If ... Then* para filtrar las transacciones entrantes. Estas reglas son diseñadas por un experto en el dominio. La eficacia del sistema depende del conocimiento del experto y de la buena definición de las reglas.

Sistema de votación (*Scorer*): herramientas basadas en modelos estadísticos diseñados para reconocer transacciones fraudulentas de acuerdo con un número de indicadores derivados de las características de la transacción. La eficacia del sistema depende de la cantidad y calidad de las transacciones históricas disponibles; se precisa grandes volúmenes de datos para obtener buenos resultados.

Redes Neuronales: las redes se basan en el reconocimiento de patrones de fraude. Se apoyan en el conocimiento de transacciones históricas válidas y fraudulentas para su entrenamiento (aprendizaje), dando como resultado un sistema que identifica el grado de riesgo o fraude de una transacción en base a los patrones aprendidos. Al igual que en los sistemas de votación, la eficacia del sistema depende de la cantidad y calidad de las transacciones históricas disponibles. El sistema obtenido no es fácilmente interpretable por un analista ni modificable para incluir conocimiento previo. Es capaz de adaptarse y aprender sin necesidad de hacer asunciones estadísticas sobre el modelo de datos.

Revisión manual: Consiste en la revisión manual de las transacciones sospechosas. Este método es extremadamente caro, lento y poco efectivo.

Cuadro 2.1: Monitoreo del comportamiento transaccional

decisión en el cuadro 2.3. Este proceso requiere una configuración (definición de reglas, forma de cálculo del puntaje de riesgo, umbrales) que debe realizarse en forma manual.

Evaluación de reglas

Esta fase consiste en evaluar un conjunto de reglas sobre cada transacción. Para mostrar una regla nos limitaremos a dar sus condiciones, ya que todas siguen la forma:

If **condiciones** *Then* **fraude**.

Por ejemplo, una regla podría ser:

$$MCC = 6010 \wedge MONTO \geq 10000$$

Si una transacción es clasificada como fraudulenta por alguna regla se realizan las fases posteriores. Si no es clasificada por ninguna regla no se genera una alerta para ella. Las reglas expresan condiciones sobre las propiedades de cada transacción, la historia y otros factores. Existen tres niveles de propiedades de acuerdo a si son propiedades derivables de los

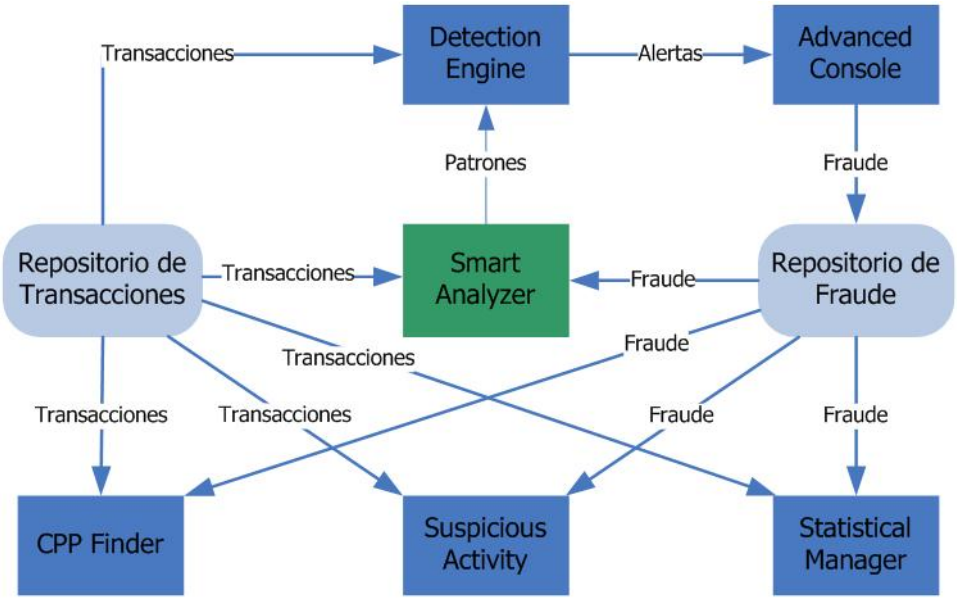

Figura 2.2: Risk Center

datos disponibles en la transacción (Nivel 1), si son propiedades agregadas a partir de un histórico de transacciones (Nivel 2) o si son propiedades no agregadas que involucran transacciones anteriores (Nivel 3).

Nivel 1
En un primer nivel, las propiedades disponibles se corresponden con los datos intercambiados en el protocolo ISO 8583. Además, si la entidad financiera es un emisor o un adquirente, es posible disponer de datos adicionales como nombre del tarjetahabiente, dirección del comercio y otros. Para conocer las propiedades de nivel 1 basta con conocer los datos asociados a una transacción. Las propiedades en los niveles mayores a 1 son calculadas y requieren conocer datos asociados a varias transacciones. Para que las propiedades de los niveles mayores a 1 estén disponibles deben ser indicadas al *Detection Engine*, que debe calcularlas a medida que las transacciones llegan.

Nivel 2
Las propiedades de Nivel 2 son aquellas calculadas aplicando una función de agregación a una propiedad del conjunto de transacciones anteriores que cumplen alguna condición de filtrado. Las condiciones de filtrado más comunes son: por tarjeta, producto, comercio, o por comercio y tarjeta, siempre respecto a la transacción para la cual se está calculando la propiedad. De esta forma se puede contar con nuevas propiedades tales como suma de los montos de las transacciones realizadas por la tarjeta en el último mes o cantidad de transacciones rechazadas en el mismo día, tarjeta y comercio. A su vez es posible componer varias funciones de agregación,

1. **Evaluación de reglas:** En el *Detection Engine* se deben configurar un conjunto de reglas de la forma *If* **condiciones** *Then* **fraude**. Si una transacción cumple con al menos una regla pasa a la siguiente etapa y si no cumple con ninguna de las reglas definidas no se generará una alerta para esta transacción.
2. **Cálculo de puntaje de riesgo:** Esta fase, también llamada *scoring*, consiste en calcular un puntaje de riesgo para aquellas transacciones que pasaron la primera etapa. Para realizarlo en el *Detection Engine* también se debe configurar un scorer.
3. **Umbral:** Si el puntaje de riesgo supera el umbral definido, se genera una alerta para esa transacción.

Cuadro 2.2: Proceso de decisión del *Detection Engine*

por ejemplo, para calcular el máximo del gasto mensual de una tarjeta, o máximo de la suma de los montos de las transacciones por mes de la tarjeta durante el último año.

Nivel 3
Las propiedades de Nivel 3 representan las tendencias en las transacciones cercanas en el tiempo. Para su cálculo se consideran los mismos tipos de filtrado que para las transacciones de Nivel 2, pero no se aplican funciones de agregación. Por ejemplo se puede calcular si las últimas tres transacciones de la tarjeta fueron por montos crecientes o realizadas en el mismo comercio, si se realizaron varias transacciones con una tarjeta con pocos segundos de diferencia entre cada una (*velocity check*) o si en las últimas transacciones de algún comercio aparecen números de tarjeta consecutivos.

Para poder configurar este tipo de reglas el *Detection Engine* permite acceder a cualquier propiedad de alguna de las últimas N transacciones que cumplen alguna de las condiciones de filtrado disponibles para el cálculo de atributos agregados (agrupados por tarjeta, producto, comercio, o por comercio y tarjeta). Para saber si las últimas 3 transacciones de una tarjeta fueron por montos crecientes, se podría escribir la regla:

$$MONTO > TX_1.MONTO \wedge TX_1.MONTO > TX_2.MONTO$$

donde TX_N es la N-ésima transacción hacia atrás en el tiempo.

Puntaje de riesgo y umbral
Sólo aquellas transacciones clasificadas como fraudulentas por alguna regla se procesan en esta fase. El *Detection Engine* calcula para cada transacción un puntaje de riesgo (*risk score*) que toma valores en el rango entre

cero y uno y genera alertas sólo para aquellas transacciones cuyo puntaje de riesgo supera un cierto umbral. La forma de hacer el cálculo puede variar, se puede calcular como una combinación lineal de las propiedades de la transacción, definir por extensión en forma de matriz N-dimensional (con N igual a la cantidad de propiedades de la transacción), o mediante cualquier forma que se desee, por ejemplo mediante una red neuronal.

Se pueden calcular varios puntajes de riesgo distintos para una transacción. A partir de los distintos puntajes de riesgo y umbrales, el *Detection Engine* genera una alerta si al menos una de las predicciones indica que la transacción es fraudulenta.

Smart Analyzer: El *Smart Analyzer* es un producto que *PayTrue solutions* desea desarrollar para asistir a mantener la configuración del *Detection Engine* sugiriendo reglas y *scorers* que puedan ser configurados en el *Detection Engine* para mejorar la detección de fraude.

Será utilizado de forma periódica (por ejemplo semanal o mensualmente) y no debe operar en forma *online*, por esto se admite que su ejecución insuma varias horas de ser necesario. Para poder utilizarlo se deberá contar con una base de datos de transacciones clasificadas como fraudulentas o legítimas. Usualmente esta base de datos es una copia de la base de datos operativa de una entidad financiera.

Nuestro proyecto tiene como fin desarrollar el núcleo de este producto. Por lo tanto, se estudiarán técnicas de aprendizaje automático para determinar el o los algoritmos más adecuados para deducir reglas (en especial reglas de nivel 2) y calcular un *score* de acuerdo al dominio particular del problema.

Capítulo 3

Marco teórico de la solución

Parte de los objetivos del proyecto es el desarrollo de una herramienta capaz de sugerir configuraciones posibles del *Detection Engine*. Debe poder sugerir reglas de clasificación y funciones de evaluación para ser utilizadas por el *Detection Engine* en su proceso de evaluación. Para cumplir este objetivo utilizamos técnicas del área de aprendizaje automático que detallamos dentro de este capítulo, junto con medidas de evaluación que permiten evaluar su efectividad.

En la sección 3.1 presentamos el método de aprendizaje de árboles de decisión utilizado para la obtención de reglas. En la sección 3.2 introducimos los métodos utilizados para la obtención del puntaje de riesgo, las redes neuronales artificiales y *random forest*. En la sección 3.3 explicamos brevemente las medidas utilizadas para la evaluación de los distintos métodos.

3.1. Aprendizaje de reglas

Uno de los principales objetivos del proyecto es la evaluación de métodos de clasificación que permitan obtener reglas que generen alertas de fraude a partir de transacciones reales. Para esto utilizamos distintos algoritmos de aprendizaje automático que construyen, a partir de un conjunto de entrenamiento, árboles de decisión capaces de clasificar una transacción como fraudulenta o legítima.

El método de aprendizaje de árboles de decisión se encuentra entre los más populares de inferencia inductiva y ha sido aplicado exitosamente a un amplio rango de tareas de aprendizaje como ser diagnósticos médicos, asesoramiento para riesgo crediticio en préstamos bancarios y fraude en medios de pago [44, 40, 41, 21]. Es un método de aproximación de funciones, robusto frente a la presencia de datos erróneos y capaz de aprender expresiones disyuntivas. Existe toda una familia de algoritmos de aprendizaje de árboles de decisión que incluye a algoritmos como ID3 y C4.5 [26].

3.1.1. Árboles de decisión

La figura 3.1 muestra un árbol de decisión, para el concepto *fraude*. Los nodos internos involucran una prueba del valor de un atributo de la instancia a clasificar y las ramas que descienden de este nodo son rotuladas con los valores posibles de la prueba. Generalmente la prueba es la comparación del valor del atributo con una constante. Los nodos hoja indican el valor retornado en caso de que dicho nodo sea alcanzado. Estos valores corresponden a posibles clasificaciones de una instancia [22].

Una instancia se clasifica comenzando en el nodo raíz del árbol, testeando el atributo especificado por este nodo y moviéndose hacia abajo por la rama del árbol que corresponde al valor del atributo en la instancia a clasificar. Este proceso se repite para el subárbol cuya raíz es el nuevo nodo, y así sucesivamente hasta alcanzar un nodo hoja, en cuyo caso se retorna la clasificación asociada con este nodo. Por ejemplo una instancia con monto 800, MCC 6010 y 5 transacciones es clasificada como positiva por el árbol de decisión de la figura 3.1.

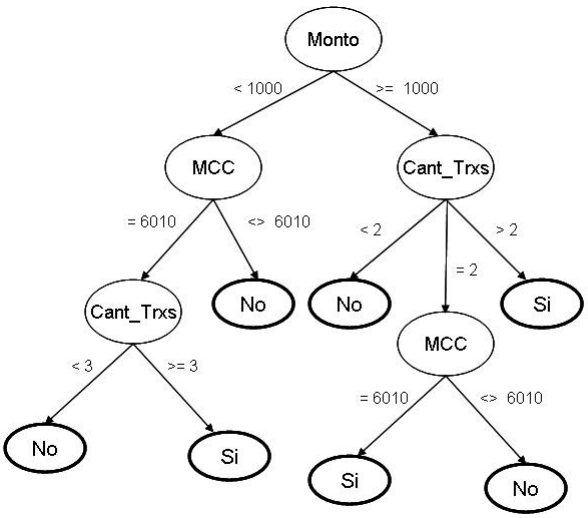

Figura 3.1: Árbol de decisión

Generalmente, un árbol de decisión representa una disyunción de conjunciones de restricciones sobre los atributos. Más precisamente, cada camino desde la raíz hacia las hojas corresponde con una conjunción de pruebas sobre atributos y el árbol en sí a una disyunción de estas conjunciones. Estos árboles pueden ser convertidos en conjuntos de reglas *if-then-else* para facilitar la interpretación por parte de una persona. Esto es una gran ventaja respecto a otros métodos, como por ejemplo redes neuronales, ya que permite, además de comprender el clasificador, modificarlo para incluir conocimiento adquirido por fuera del entrenamiento.

Entre los problemas en los que se puede aplicar árboles de decisión como método

ID3(*Ejemplos*, *AtributoObjetivo*, *Atributos*)

1. Se crea el nodo raíz
2. Si todos los ejemplos son positivos, se etiqueta la raíz como positiva
3. Si todos los ejemplos son negativos, se etiqueta la raíz como negativa
4. Si el conjunto *Atributos* es vacío, se etiqueta la raíz con el valor de *AtributoObjetivo* más común en *Ejemplos*
5. En caso contrario,
   a) A = Atributo del conjunto *Atributos* que mejor clasifica los *Ejemplos*
   b) Para cada valor posible v_i de A:
      1) Agregar una nueva rama debajo de A correspondiente a la prueba $A = v_i$
      2) Sea $Ejemplos_{v_i}$ el subconjunto de *Ejemplos* donde $A = v_i$
      3) Si el conjunto $Ejemplos_{v_i}$ es vacío, se etiqueta con el valor de *AtributoObjetivo* más común en $Ejemplos_{v_i}$
      4) Si el conjunto $Ejemplos_{v_i}$ no es vacío, se etiqueta con el valor de $ID3(Ejemplos_{v_i}, AtrbutoObjetivo, Atributos - A)$

Cuadro 3.1: Seudocódigo del algoritmo ID3

de aprendizaje, se encuentran aquellos en los cuales las instancias se representan como pares atributo-valor, la función objetivo tiene valores de salida discretos, pueden requerirse descripciones disyuntivas o cuando el conjunto de entrenamiento puede presentar errores o valores ausentes.

La mayoría de los algoritmos de aprendizaje de árboles de decisión se basan en un algoritmo que realiza una búsqueda *greedy-top-down* a través del espacio de posibles árboles de decisión. Se selecciona el atributo a colocar en la raíz y se dividen las ramas según los posibles valores que éste pueda tomar. Así, se divide el conjunto de ejemplos en subconjuntos, uno por cada valor posible del atributo. El proceso se repite recursivamente para cada rama, utilizando únicamente las instancias que las alcanzan. Cuando todas las instancias en un nodo tienen la misma clasificación o ya se utilizaron todos los atributos se finaliza el proceso. El cuadro 3.1 contiene un seudocódigo del algoritmo ID3 obtenido de [48].

Lo que resta por definir es cómo determinar el atributo por el cual particionar dado un conjunto de ejemplos con diferentes clases. Para definirlo cada instancia de atributo es evaluada estadísticamente para determinar qué tan bien clasifica el conjunto de entrenamiento. Una medida de evaluación utilizada, definida en el cuadro 3.2, es la ganancia de información. Por su definición al utilizar ganancia

de información se favorece a los atributos con muchos valores sobre aquellos con pocos valores. Para evitarlo se podría tratar de ajustar la medida de ganancia de información de manera tal de penalizar aquellos atributos que tengan un número muy grande de valores posibles. Esta idea originó el concepto de *gain ratio*, definido en el cuadro 3.3.

$Entropy(S) = -\sum_{i=1}^{k} p_i * \log_2 p_i$

$Entropy_A(S) = -\sum_{i=1}^{v} p_i * Entropy(S_i)\ ,\ p_i = \frac{|S_i|}{|S|}$

$Gain_A(S) = Entropy(S) - Entropy_A(S)$

siendo p_i la proporción de ejemplos pertenecientes a la clase i-ésima

---

Cuadro 3.2: Ganancia de información

$SplitInformation_A(S) = -\sum_{i=1}^{v} p_i * \log_2 p_i\ ,\ p_i = \frac{|S_i|}{|S|}$

$GainRatio_A(S) = \frac{Gain_A(S)}{SplitInformation_A(S)}$

Cuadro 3.3: *Gain ratio*

Cuando hay ruido en los datos o no se tienen suficientes ejemplos, puede ocurrir que el árbol de decisión aprendido se sobreajuste al conjunto de entrenamiento, es decir, clasifique correctamente el conjunto de entrenamiento pero no clasifique satisfactoriamente las instancias de un nuevo conjunto. Existen varias técnicas para evitar el sobreajuste que se pueden agrupar en dos clases, las que detienen el crecimiento del árbol antes de que clasifique perfectamente a todas las instancias del conjunto de entrenamiento y las que permiten crecer al árbol y luego lo podan. Como se indica en [48], la segunda clase ha demostrado ser más efectiva en la práctica y consiste en eliminar subárboles completos, remplazándolos por una hoja. La nueva hoja se rotula con el valor más frecuente dentro de las instancias que alcanzan el subárbol eliminado. Luego se evalúa si la poda realmente mejoró el resultado utilizando un nuevo conjunto de validación o un test estadístico sobre el conjunto de entrenamiento.

Para tratar atributos numéricos continuos es necesaria la definición dinámica de nuevos atributos que particionan los atributos continuos en un conjunto discreto de intervalos, utilizando un umbral para particionar. Para elegir el umbral se puede ordenar los ejemplos de entrenamiento de acuerdo al atributo numérico para identificar los ejemplos adyacentes en que se producen los cambios en la clasificación. Un conjunto de umbrales candidatos esta dado por los valores

intermedios del atributo en que dichos cambios se producen. Una aproximación para encontrar el umbral óptimo consiste en, dados los posibles puntos intermedios para el umbral, seleccionar aquel que produce la mayor ganancia de información. Para tratar atributos faltantes se puede considerar el valor más común del atributo entre los ejemplos en el nodo o utilizar algún procedimiento más complejo como el utilizado en el algoritmo C4.5 [27].

El método que utiliza la ganancia de información en los nodos es conocido como ID3. Se realizaron mejoras posteriores en el sistema C4.5 y el posterior C5.0, que incluyen el uso de la tasa de ganancia, manejo de atributos numéricos, valores ausentes y ruido en los datos [48, 22].

3.2. Aprendizaje del puntaje de riesgo

La asignación de un puntaje de riesgo a una autorización es uno de los objetivos del proyecto. Los modelos de asignación de puntaje o *scorers* son muy utilizados para el estudio del comportamiento de clientes, tanto en ambientes financieros para la concesión de créditos como en *marketing* para la construcción de aplicaciones CRM (*Customer Relationship Management*).

En el contexto de estudio, luego de que un modelo de predicción es creado a partir de datos históricos y algún algoritmo de aprendizaje automático, el siguiente paso es utilizarlo para predecir la probabilidad de que una autorización no clasificada sea fraudulenta o no. Esta probabilidad es llamada *behavioural score* de la autorización. El *scorer* o *scoring engine* a construir recibirá como entrada un conjunto de datos y un modelo dando como resultado la predicción de clase y el puntaje para cada elemento del conjunto, como podemos ver en la figura 3.2.

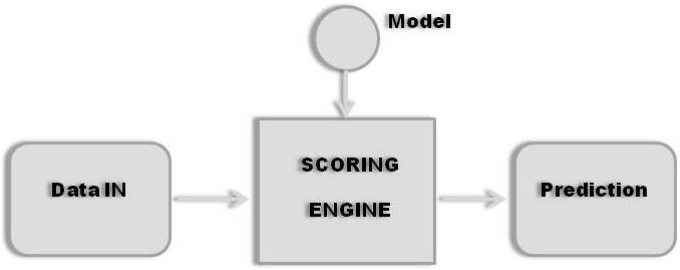

Figura 3.2: *Scoring Engine*

Elegimos dos tipos de algoritmos para la construcción del modelo de predicción: un paradigma de aprendizaje automático inspirado en el funcionamiento del sistema nervioso animal, denominado red neuronal [36], y *random forest* [34], el cual se basa en un conjunto de clasificadores de árboles de decisión.

3.2.1. Redes Neuronales Artificiales

Una red neuronal artificial (a partir de ahora utilizaremos las siglas en inglés *ANN*) puede ser vista como un conjunto de nodos o neuronas interconectadas, donde cada conexión tiene asociada un peso y cada neurona tiene internamente un sesgo que altera la salida que será enviada a los siguientes nodos.

Existen tres tipos de neuronas, las de entrada, que reciben los datos desde fuera de la red neuronal (no realizan modificaciones sobre los datos), las de salida, que envían datos hacia fuera y las ocultas (en inglés *hidden*), cuyas señales de entrada y salida quedan siempre dentro de la red. La salida de una neurona es conformada por una función de propagación, la cual normalmente consiste en la sumatoria de las entradas multiplicadas por el peso de su conexión, una función de activación que modifica la salida anterior y una función de transferencia que acota la salida (por ejemplo la tangente hiperbólica para valores en el intervalo [-1,1] y la función sigmoidea para valores en el intervalo [0,1]).

El tipo de redes utilizadas en este proyecto son *Feed-Foward*, donde los datos fluyen desde los nodos de entrada (en inglés *input*) hacia los nodos de salida (en inglés *output*) en un camino directo (hacia adelante). Se mantiene una estructura de capas, donde los nodos de una misma capa no se interconectan entre sí. En este tipo de red, los nodos de entrada no realizan cómputos, por lo que, como convención, la capa de entrada no se cuenta. De esta forma una red de dos capas es aquella que tiene una capa de entrada, una oculta y una de salida [5].

La versión más sencilla de una red neuronal *Feed-Foward* es el perceptrón, un método iterativo que ajusta los pesos y predisposiciones de la red a partir de un conjunto de aprendizaje. La figura 3.3 muestra una red con dos nodos de entrada y una única salida [5].

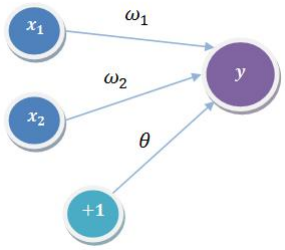

Figura 3.3: Red con dos nodos de entrada y una única salida

Siendo ω_i el peso del i-ésimo conector y θ el valor de predisposición del nodo, el problema a resolver se puede ver como buscar obtener en cada iteración los $\Delta\omega_i(t)$ y $\Delta\theta(t)$ que corrijan el valor anterior de forma de permitir clasificar correctamente las instancias de entrenamiento:

- $\omega_i(t+1) = \omega_i(t) + \Delta\omega_i(t)$
- $\theta(t+1) = \theta(t) + \Delta\theta(t)$

Perceptrón

1. Iniciar con pesos cualesquiera en las conexiones
2. Seleccionar un vector x de entre los ejemplos de entrenamiento
3. Si la clasificación del perceptrón (llamémosla y) difiere de la clasificación esperada (llamémosla $d(x)$), se deben modificar todas las conexiones ω_i de la siguiente forma:
   $\Delta\omega_i = d(x)x_i$ siendo x_i el valor del nodo de donde proviene el conector.
4. Ejecutar 2 nuevamente mientras existan más ejemplos para procesar.

Cuadro 3.4: Seudocódigo del algoritmo perceptrón

Siendo entonces x un vector del conjunto de entrenamiento y $d(x)$ su clasificación, el perceptón se comporta de la forma explicada en el cuadro 3.4. Lo mismo aplica para $\Delta\theta(t)$, solo que como es un valor interno al nodo (no tiene conector), x_i vale siempre 1. Observando que los valores son modificados únicamente cuando la instancia no es clasificada correctamente, $\Delta\theta(t)$ toma el valor 0 cuando la instancia x es correctamente clasificada y $d(x)$ cuando no lo es.

Se puede demostrar que de existir un conjunto correcto de pesos de conexiones capaz de realizar la transformación buscada, el perceptrón converge a alguna solución en un número finito de pasos, cualquiera sea la selección inicial de pesos [5]. El perceptrón en su forma más simple consiste en una capa de entrada de N elementos que alimenta una capa de M elementos y un único nodo de salida.

El perceptrón tiene limitaciones de representación que se solucionan al agregar una capa nueva de predicados, los cuales deben estar todos conectados a la capa de entrada. Esta clase de red neuronal es llamada perceptrón multicapa y es la utilizada en este proyecto.

Para ajustar los pesos de las conexiones de un perceptrón multicapa durante el entrenamiento de la red, se propagan los valores de activación de cada ejemplo de entrenamiento por la red hasta las unidades de salida, donde se compara el resultado con el resultado esperado. Esta comparación permite obtener un error para cada nodo de salida producto de la salida esperada y la obtenida. Para poder llevar el error en cada nodo de salida a cero se utiliza la regla delta (en inglés *Least Mean Square*). Esta es una generalización del algoritmo de entrenamiento del perceptrón [6] que ya hemos mencionado donde la principal diferencia es que se utiliza directamente la salida de la red sin transformación adicional (el perceptrón mapea la salida en valores -1,+1). Al igual que el algoritmo del perceptrón, no modifica los pesos de los conectores que van desde las capas de entrada hasta las ocultas.

Para ajustar los conectores que van desde las capas de entrada hasta las ocultas,

se utiliza la regla de la cadena, que consiste en distribuir el error de un nodo de salida hacia todos los nodos ocultos a los que está conectado, en función al peso de su conexión. Su funcionamiento consta de dos fases, la primera propaga la entrada hacia adelante hasta la salida, de forma de poder evaluar el resultado y compararlo con el deseado, obteniendo para cada nodo de salida una señal de error. La segunda fase implica recorrer la red en sentido inverso, enviando una señal de error que ajusta los pesos.

Una de las características más interesantes de una *ANN* es su capacidad de aprender y ajustarse directamente a partir de conjuntos de entrenamiento [46]. Este método de aprendizaje ha sido aplicado exitosamente a un amplio rango de tareas de aprendizaje como ser fraude en medios de pago, fraude en comunicaciones móviles, detección de texto, procesamiento de imágenes y robótica [45, 56, 31, 18].

3.2.2. *Random Forest*

El clasificador *random forest* [34], o bosque aleatorio, es un *meta-learner* [1] que consiste de muchos árboles de decisión aleatorios individuales, los cuales son combinados para producir un resultado en común. Cada árbol vota por un resultado con igual peso, dando como clasificación a aquella que logre mayor cantidad de votos [17]. Este algoritmo fue diseñado originalmente para operar en grandes conjuntos de datos y utiliza muestras al azar sobre los atributos para construir los árboles. El cuadro 3.5 muestra el seudocódigo para construir cada árbol.

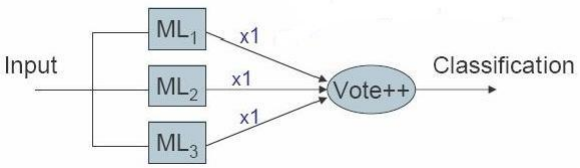

Figura 3.4: *Meta Learner* [30]

La diversidad de los árboles se obtiene como consecuencia de que (a) el árbol se construye con un subconjunto aleatorio de atributos, y (b) en cada paso se selecciona al atributo que provee el mayor nivel de aprendizaje sobre un subconjunto de instancias también seleccionado aleatoriamente. La correlación entre los árboles que componen el bosque aleatorio tiene un fuerte impacto en el rendimiento del algoritmo: si aumenta la correlación, entonces el rendimiento disminuye. La relación entre fortaleza y correlación es utilizada para establecer un límite superior para el error de generalización:

- Incrementando la correlación aumenta la tasa de error.

[1] Por definición, describe la aplicación de técnicas de aprendizaje con el objetivo de aprender sobre el proceso de aprendizaje en sí.

1. Formar un conjunto de datos [*inbag*] utilizando muestreo con reposición (*bootstrapping* [2]) en el conjunto de entrenamiento. Al utilizar esta técnica, un tercio del conjunto de entrenamiento no se encuentra presente en el [*inbag*]. Este conjunto restante se conoce como el conjunto de datos [*out-of-bag*].
2. Elegir un subconjunto de m atributos al azar (*random*) para particionar en cada árbol (en contraste a la elección del atributo que mejor particiona en la práctica usual de árboles de decisión). Estos atributos forman los nodos del árbol.
3. Construir el árbol aleatorio sin realizar poda. Este árbol se construye como se describió previamente en el cuadro, con la particularidad de que los atributos por los cuales particiona en cada nodo son los seleccionados previamente al azar.
4. Calcular la tasa de error *out of bag* procesando el conjunto de datos [*out-of-bag*] en el árbol generado previamente.

Cuadro 3.5: Seudocódigo de la construcción de árboles de un bosque aleatorio [17]

- Un árbol con baja tasa de error se considera un clasificador fuerte. Al aumentar las fortalezas individuales de los clasificadores, disminuye la tasa de error en el bosque aleatorio.

Como ventajas se puede mencionar que es un algoritmo que mantiene la precisión cuando existen datos faltantes, provee información sobre la relación entre los atributos y la clase, así como su importancia.

El error de generalización global del modelo se obtiene promediando las tasas de error, lo cual tiene varias ventajas:

1. Se utilizan todas las instancias del conjunto para construir el modelo.
2. El *testing* es rápido ya que se realiza durante la construcción del modelo y no posteriormente.

La selección y el número de atributos seleccionados durante la construcción de un árbol afecta proporcionalmente tanto a la correlación como a la tasa de error. Por último, aumentar la cantidad de árboles en el bosque provee un clasificador más inteligente; resulta análogo a contar con un amplio y diverso grupo de personas al momento de tomar una decisión.

El bosque aleatorio se ha utilizado para resolver diferentes problemas de clasificación, como diagnósticos médicos [15], clasificación de imágenes [4] y aplicaciones de predicción climática [28]. Es un método efectivo computacionalmente y

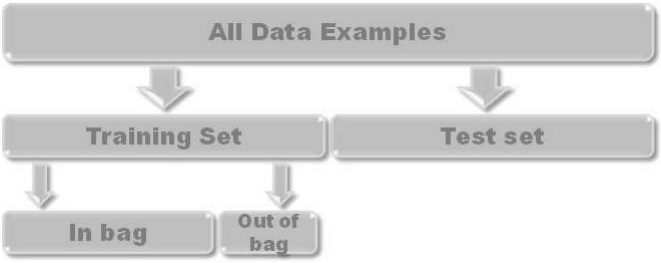

Figura 3.5: Datos usados en la construcción de un árbol

ofrece un buen rendimiento de predicción. Puede demostrarse que este clasificador no sufre de sobreajuste a medida que más árboles son agregados al bosque aleatorio [34].

3.3. Medidas de evaluación

Para evaluar los resultados obtenidos utilizamos algunas medidas de evaluación comunes a todos los algoritmos y algunas específicas del algoritmo particular. Más precisamente, las medidas de evaluación comunes consideradas son:

- Cantidad de instancias positivas clasificadas positivas, TP.
- Cantidad de instancias negativas clasificadas positivas, Falso Positivo o FP.
- Cantidad de instancias negativas clasificadas negativas, TN.
- Cantidad de instancias positivias clasificadas negativas, Falso Negativo o FN.
- Precisión, cantidad de instancias correctamente etiquetadas / Cantidad total de instancias etiquetadas *TP/TP+FP*.
- Recuperación(*Recall*), cantidad de instancias correctamente etiquetadas / Cantidad total de instancias *TP/TP+FN*.

Normalmente existe una relación inversa entre la Precisión y la Recuperación, cuando una aumenta, la otra disminuye. Usualmente se utiliza la medida-f que combina ambas medidas según un parámetro β. Utilizamos la medida correspondiente a $\beta = 1$ que prioriza de igual manera la precisión que la cobertura. La medida-f$_\beta$ se define de la siguiente forma [29]:

$$\text{medida-f}_\beta = \frac{(1+\beta) * recuperación * precisión}{recuperación + \beta * precisión}$$

Para facilitar la visualización de las medidas FP, FN, TP y TN utilizaremos la

matriz de confusión mostrada en la figura 3.6. La matriz de confusión está formada por tantas filas y columnas como clases estén involucradas en la clasificación. En cada celda aparece la cantidad de instancias de la clase correspondiente a la fila clasificada como la clase correspondiente a la columna, de esta manera, las instancias clasificadas correctamente corresponden a la diagonal de la matriz.

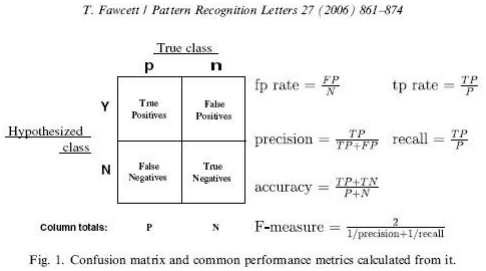

Fig. 1. Confusion matrix and common performance metrics calculated from it.

Figura 3.6: Matriz de confusión

Además para cada regla obtenida se tiene la cantidad de instancias para la cual aplica y la cantidad de instancias mal clasificadas por ella.

En *random forest* utilizamos la medida de evaluación *out of bag error*. Esta medida nos proporciona el error de generalización del bosque aleatorio, computando el error de cada árbol y luego promediando [33].

Para medir el rendimiento de los *scorers* obtenidos utilizamos otra medida, la curva de ROC (*Receiver Operating Characteristic*). La curva de ROC es una técnica para visualizar, organizar y seleccionar clasificadores basada en su rendimiento. El denominado análisis de ROC es muy utilizado en la toma de decisiones médicas y en años recientes se ha introducido su uso en aprendizaje automático y *data mining* [47]. Particularmente, utilizamos como medida al área bajo la curva de ROC.

Los gráficos de ROC como en la figura 3.7, son gráficos bidimensionales en los cuales la recuperación se representa en el eje Y y la tasa FP (cantidad de instancias negativas clasificadas como positivas sobre el total de instancias negativas) se representa en el eje X. Para el caso de clasificadores discretos, como en nuestro caso, cada clasificador produce una pareja (*tasa FP, recuperación*) que se corresponde con un punto en la curva. El punto (0,0) representa a un clasificador que nunca clasifica positivo ya que no comete falsos positivos (FP) pero tampoco genera verdaderos positivos (TP). Por otra parte el punto (1,1) representa lo opuesto, es decir, un clasificador que siempre clasifica positivo. Es de notar que el punto (0,1) representa una clasificación perfecta porque no se obtienen falsos positivos y todos los clasificados como positivos lo son realmente.

Para crear la curva de ROC, es necesario tener el *score* del clasificador. Para los algoritmos elegidos, esto corresponde a la proporción con la cual se predice

determinada clase (*fraude, no fraude*), es decir el grado con el cual una instancia es miembro de una clase.

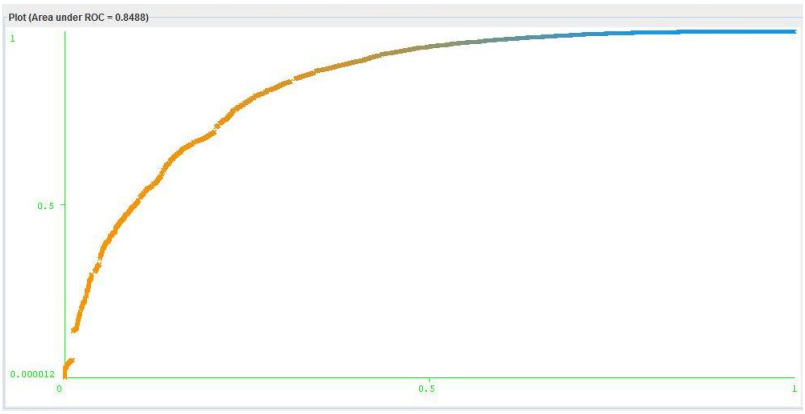

Figura 3.7: Ejemplo de Gráfica de ROC con true positive representado en eje y, false positive en el eje x .

Dado que la curva de ROC es bidimensional, el área bajo su curva permite reducirse a un único valor escalar para representar el rendimiento esperado de un clasificador. Como el valor del área es una porción de una unidad cuadrada varía entre 0 y 1 y dado que un clasificador *random* que predice *positivo* la mitad del tiempo y *negativo* la otra mitad produce una diagonal entre (0,0) y (1,1), con un área de 0.5, ningún clasificador *realista* debería tener un área menor a 0.5.

Capítulo 4

Conjunto de datos de entrada

Para cumplir con los objetivos del proyecto es necesario contar con un conjunto de autorizaciones clasificadas como legítimas o fraudulentas. *PayTrue solutions* proporcionó una base de datos con todas las autorizaciones y fraudes correspondientes a un año de actividad de un cliente.

Necesitamos optimizar el conjunto de datos de entrada para poder utilizarlo en el aprendizaje. Para lograrlo, debimos reducir la cantidad de transacciones a tomar en cuenta, deducir atributos mediante funciones de agregación sobre los atributos presentes en la transacción y por último seleccionar los atributos que resultaran más relevantes para la detección de fraude.

El capítulo se divide de la siguiente forma: en la sección 4.1 mostramos las características básicas del conjunto proporcionado originalmente, en la sección 4.2 detallamos los atributos calculados para obtener reglas de Nivel 2 en el *Detection Engine*. En la sección 4.3 tratamos la reducción del conjunto de datos, de forma que podamos utilizarlo como entrada en el entrenamiento y evaluación de los algoritmos. Por último, en la sección 4.4 detallamos las técnicas de selección de atributos y los resultados obtenidos.

4.1. Descripción del conjunto

PayTrue solutions proporcionó una base de datos con una tabla con los registros de las autorizaciones y otra tabla con los registros de los fraudes reportados, en donde los números de tarjeta fueron previamente ofuscados por privacidad. El tamaño de la tabla de transacciones superaba los 43 millones de registros y la tabla de fraude superaba los 30 000 aproximadamente.

Debido a que los algoritmos de aprendizaje seleccionados precisan que las transacciones estén clasificadas como fraudulentas o legítimas, nuestro primer paso consiste en el etiquetado de cada transacción.

Las tablas provistas no se encuentran explícitamente relacionadas mediante

ningún campo, y no es posible relacionar cada denuncia de fraude con su respectiva autorización de forma simple. Para clasificar cada autorización debimos detectar si fue reportada como fraude relacionando ambas tablas por algunos de los campos disponibles, como por ejemplo, el número de tarjeta, el monto y la fecha de la transacción. La relación entre estos campos no es directa ya que pueden existir diferencias entre el monto y fecha del fraude reportado y la autorización que originó la venta.

Luego de aplicado el proceso de etiquetado, obtuvimos un conjunto de autorizaciones de las cuales 0,0435 % son fraudulentas. Los campos de las instancias son los siguientes:

Atributo	***Descripción***
CARDID	Número de tarjeta
AMOUNT	Monto de la autorización en dólares americanos
TRANDATE	Fecha de la autorización en formato día, mes y año
MERCHANTCODE	Código del comercio
MCC	Tipo de negocio o rubro del servicio
ACQUIRER	Número identificador del adquirente
FRAUDE	Clasificación de la autorización

Cuadro 4.1: Atributos en el conjunto de datos de entrada

Como podemos ver en la tabla, no contamos con el atributo *hora* ya que no figuraba en los datos proporcionados. Este atributo resulta de interés para la detección de fraude porque permite distinguir una cantidad de transacciones realizadas en un corto período de tiempo. Por una descripción completa de los atributos originales referirse al Apéndice B - Conjunto de Datos.

En el Apéndice C - Etiquetado del Fraude describimos en detalle el proceso de etiquetado.

4.2. Atributos calculados

Parte de los objetivos de este trabajo es descubrir propiedades de Nivel 2 que resulten útiles para la determinación del fraude. Para cada autorización calculamos la cantidad de autorizaciones y la suma, mínimo, máximo y promedio del monto considerando distintas particiones del conjunto de autorizaciones con igual *día, semana y mes*. Los atributos de partición que tomamos fueron:

1. Misma tarjeta (CARDID2).

2. Misma tarjeta y mismo MCC (CARDID2 - MCC).

3. Misma tarjeta y mismo comercio (CARDID2 - MERCHANTCODE).

4. Mismo comercio (MERCHANTCODE).

Para las particiones 1 y 4 calculamos atributos adicionales tomando en cuenta únicamente las transacciones mayores a un *monto* y menores a un *monto*. Tomamos los valores 500 y 10 respectivamente. Realizamos la selección de estos valores empíricamente intentando detectar transacciones riesgosas por montos elevados y transacciones por muy bajo monto que se correspondan a patrones de testeo de cuenta. Para la partición 1, además, realizamos un filtrado por $MCC = 6010$ y $MCC = 6011$ y *monto mayor* y *menor* a los valores mencionados, con las mismas funciones de agregación anteriores pero con el fin de diferenciar si la autorización fue realizada por cajero automático (identificado por MCC=6011 o MCC=6010) ya que los patrones detectados para este tipo de servicio presentan características particulares con respecto al resto.

La intención de los cálculos considerados es detectar patrones de fraude según la frecuencia de compra, montos de las compras fraudulentas y cantidad de transacciones fraudulentas seguidas para la misma tarjeta entre otros. Realizamos la selección de los atributos considerados para las particiones de forma empírica, dado que no encontramos antecedentes que describiesen indicadores efectivos para la detección de fraude. El objetivo es poder determinar cuáles atributos tienen influencia sobre el fraude.

En el Apéndice D - Cálculo de atributos detallamos los atributos calculados. En total agregamos a nuestro conjunto inicial 150 nuevos atributos, aumentando el tamaño de la base de datos veintidós veces el tamaño original.

4.3. Selección de instancias

Como se desprende de las secciones anteriores, el volumen del conjunto de datos de entrada es extremadamente grande y además, las autorizaciones fraudulentas luego del etiquetado no superan el 0,0435 % del total. Ya sea por la diferencia entre la cantidad de instancias de las dos clases (*fraude / no fraude*) como por el tamaño de la base de datos, nos vimos obligados a reducir el conjunto de instancias para poder ejecutar los algoritmos de aprendizaje automático.

Consideramos que dado que realizamos los cálculos de la sección anterior tomando en cuenta todo el conjunto, no perdemos demasiada información al reducirlo, por contener dichos atributos información de todo el conjunto original de datos. Para el muestreo tomamos todas las transacciones marcadas como fraudulentas (aproximadamente 18 500) y 200 000 no fraudulentas tomadas al azar del resto del conjunto. Con esta decisión transformamos el universo en uno más fraudulento que el original, pero optamos por tomar una actitud más conservadora, prefiriendo obtener más falsos positivos a no detectar positivos verdaderos.

De este conjunto, con el fin de reducir el posible sesgo, utilizamos el 12 % exclusivamente para la selección de atributos a describirse en 4.4, el 48 % como conjunto de entrenamiento para la generación de reglas y la función de evaluación y reservamos el restante 40 % como conjunto de evaluación para las reglas y función de evaluación obtenidas previamente.

4.4. Selección de atributos

Con frecuencia la selección de atributos es aplicada a datos de gran volumen como paso previo al aprendizaje de clasificación. La dimensión es equivalente a la cantidad de atributos que describe una instancia. Si bien muchos atributos pueden resultar en mayor poder discriminativo, en la práctica, conjuntos con excesiva cantidad de atributos pueden enlentecer el proceso de aprendizaje y sufrir sobreajuste [22]. El objetivo de este proceso es extraer la información más relevante de una cantidad significativa de datos observados para facilitar el análisis de los datos, eliminando aquellos atributos que sean redundantes (correlacionados con otros) o irrelevantes (no aportan a la predicción).

La motivación puede considerarse tanto computacional como teórica [49]. Dentro de las razones computacionales, encontramos que al reducirse la dimensión de los datos se reduce el tamaño del espacio de hipótesis lo cual se traduce en menor tiempo de ejecución de los algoritmos de clasificación y en menores costos computacionales de memoria. Dentro de las razones teóricas podemos mencionar que ciertos algoritmos poseen un pobre desempeño en presencia de atributos irrelevantes o redundantes con lo cual se obtiene una mejora de la precisión en las predicciones.

Los métodos de selección de atributos para clasificación se dividen según Guyon *et al.* [24] en tres categorías. La primer categoría se conoce como *filter* en la cual la selección de atributos es realizada en una etapa de preprocesamiento que es independiente a la clasificación, donde los atributos son elegidos de acuerdo a un puntaje calculado para cada uno de ellos. En la segunda categoría tenemos los métodos *wrapper* [43], que utilizan un sistema de aprendizaje del estilo caja negra para asignar un puntaje a un conjunto de atributos. Por último se encuentran los *embedded methods* en los cuales la selección de atributos se realiza dentro del proceso de entrenamiento [22].

Puede observarse que alcanzar una selección óptima requiere de una búsqueda exhaustiva de todos los posibles subconjuntos de atributos para una cardinalidad dada, lo cual resulta poco práctico para grandes cantidades de atributos. Es por esto que en la práctica, los diferentes algoritmos por lo general buscan un conjunto que sea satisfactorio, en vez del óptimo. Dentro de los algoritmos de selección de atributos encontramos: *Feature Ranking* y *Subset Selection* [22]. El primero ordena jerárquicamente los atributos utilizando una métrica determinada y elimina todos aquellos que no alcanzan un determinado umbral, mientras que el segundo busca el subconjunto de atributos que más se aproxime a un óptimo.

Mediante este proceso determinamos la importancia de los atributos originales y cuáles atributos calculados son realmente relevantes y por lo tanto vale la pena calcular. Los atributos continuos son discretizados mediante el método de Fayyad e Irani [16] para ejecutar los algoritmos de selección a continuación. Las implementaciones de los algoritmos utilizados, mencionados a continuación, pertenecen a la plataforma WEKA [53].

4.4.1. *Feature ranking*

Los algoritmos de este tipo poseen la ventaja de tener un mayor rendimiento respecto de otros métodos, ya que cada atributo se evalúa por separado, independientes unos de otros, utilizando una medida de la información que aporta el atributo para determinar la clase a la cual pertenece la instancia [52]. Estos algoritmos pertenecen a la categoría *filter* dado que normalmente son utilizados en la etapa de preprocesamiento de los datos para filtrar atributos. Además, utilizan una función de evaluación para construir el *ranking* y no se valen de un método de aprendizaje. Una característica importante a destacar es que al evaluar los atributos de forma individual, las relaciones entre estos no son tomadas en cuenta. A modo de ejemplo si se utilizan tres atributos como ser día, mes y año, serán evaluados de forma individual, sin considerar la información que puedan aportar en conjunto. Otro inconveniente es que tampoco se toma en consideración si existen atributos que tienen un alto *ranking* pero aportan exactamente la misma información. El cuadro 4.2 contiene el seudocódigo de un algoritmo de ranking típico.

Existe una gran variedad de medidas de evaluación de atributos, muchas de las cuales están basadas en la entropía. Estudios de comparación de métodos de filtro realizados por Yang *et al.* [55] y Forman [19] concluyen que la ganancia de información y χ-*square* (ambos medidas de información de un atributo) se encuentran entre los métodos más efectivos de selección de atributos para clasificación. Por lo tanto, decidimos realizar pruebas utilizando los mencionados esquemas así como la medida *gain ratio*. Elegimos la última dado que la clasificación utilizará árboles de decisión y es una medida común en algoritmos exitosos como ser *ID3* [26].

1. Se inicializa Γ con todos los atributos. $\Omega = \varnothing$
2. Para cada atributo f de Γ, se computa $\rho(f)$
3. Encontrar f que maximiza $\rho(f)$. $\Omega \leftarrow \Omega \cup \{f\}$, $\Gamma \leftarrow \Gamma \backslash \{f\}$
4. Repetir hasta que el cardinal de Ω sea p

Cuadro 4.2: Seudocódigo de una algoritmo de *ranking* [51]

En el cuadro 3.2 se observa la fórmula matemática de ganancia de información, que se define como la reducción esperada de la entropía al particionar los ejemplos de acuerdo a un atributo. Las bases teóricas del concepto de entropía se encuentran en el campo de la teoría de la información, a partir de la definición de entropía desarrollada por Shannon [7]. Una interpretación de la entropía tomada del libro de Mitchell [48] es que especifica el mínimo número de bits de información necesarios para codificar la clase de una instancia arbitraria del conjunto.

Existe un sesgo natural al utilizar la ganancia de información como medida de un atributo, dado que favorece a los atributos con múltiples valores sobre aquellos que tienen pocos valores [48]. Si se estuviese construyendo un árbol de decisión, el resultado sería un árbol ancho con muchas ramas pero de poca profundidad. El problema con este enfoque es que si bien presenta mucha ganancia de información para el conjunto de entrenamiento, se convierte en un predictor pobre para las instancias nunca vistas. Una alternativa para evitar este problema es la medida de *gain ratio* [26], definido en el cuadro 3.3. Se incorpora un término a la medida ganancia de información que indica la información de la partición (*split information*) que es sensible en cuanto a la uniformidad con la que un atributo separa los datos.

Por otra parte, χ^2 (*chi-cuadrado*) es una medida estadística utilizada para probar la independencia de dos eventos. Dos eventos A y B son independientes si $P(AB) = P(A)P(B)$, lo cual equivale a decir $P(A|B) = P(A)$ y $P(B|A) = P(B)$ [8]. En selección de atributos, los eventos son la ocurrencia de un término y una clase. Los atributos numéricos deben ser discretizados en intervalos previamente. Cuanto mayor es el valor obtenido con χ^2, más importante resulta el atributo. Definimos el valor χ^2 de un atributo en el cuadro 4.3 , donde m es el número de intervalos, k el número de clases, A_{ij} el número de ejemplos en el intervalo i, clase j, R_i el número de ejemplos en el intervalo i, C_j es el número de ejemplos de la clase j, N es el total de ejemplos (instancias) y E_{ij} es la frecuencia esperada de A_{ij} ($E_{ij} = R_i * C_j/N$) [20].

$$\chi^2 = \sum_{i=1}^{m} \sum_{i=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

Cuadro 4.3: χ_2

En los cuadros 4.4, 4.5 y 4.6 mostramos los resultados obtenidos en cada caso, para los primeros 15 atributos. Observamos que la intersección de los tres conjuntos resulta en los atributos {`TAR_MAYOR_DIA, TAR_MAYOR_CANT_DIA`} [2]. Además, el conjunto de atributos obtenidos con ganancia de información y con χ^2 es muy similar, resultando su intersección en {`TAR_SIMPLE_DIA, TAR_SIMPLE_SEMANA, TAR_MAYOR_SEMANA, TARMCC_SIMPLE_DIA, TARMCC_SIMPLE_SEMANA, TARCOM_SIMPLE_DIA, TARMCC_SIMPLE_MAXIMO_SEMANA, TARCOM_SIMPLE_SEMANA, TAR_SIMPLE_MAXIMO_SEMANA, TAR_MAYOR_MAXIMO_SEMANA, TARCOM_SIMPLE_MAXIMO_SEMANA`}.

4.4.2. *Subset selection*

Un enfoque alternativo para la reducción del conjunto de atributos es el de *subset selection*, donde en lugar de ordenar jerárquicamente los atributos, se busca el subconjunto de atributos que más se aproxima a un óptimo. Dados N atributos tenemos 2^N subconjuntos posibles, por lo que la generación de subconjuntos

[2] Por una descripción de los atributos mencionados en esta seccción referirse al Apéndice D

Ganancia	***Atributo***
0,0826	TAR_SIMPLE_DIA
0,0809	TAR_SIMPLE_SEMANA
0,0783	TAR_MAYOR_SEMANA
0,0751	TARMCC_SIMPLE_DIA
0,0739	TAR_MAYOR_DIA
0,0723	TARMCC_SIMPLE_SEMANA
0,0717	TAR_MAYOR_CANT_DIA
0,0703	TAR_SIMPLE_MAXIMO_SEMANA
0,0696	TARCOM_SIMPLE_DIA
0,0683	TARCOM_SIMPLE_SEMANA
0,0681	TARMCC_SIMPLE_MAXIMO_SEMANA
0,0680	TARCOM_SIMPLE_MAXIMO_SEMANA
0,0672	TAR_MAYOR_MAXIMO_SEMANA
0,0671	TAR_SIMPLE_MAXIMO_DIA

Cuadro 4.4: *Information Gain - Primeros 15 atributos en el ranking*

requiere una estrategia de búsqueda determinada. Una vez que se conforma un subconjunto, debe evaluarse su mérito.

Como heurística para la búsqueda de subconjunto elegimos la de *best first*, que realiza la búsqueda en el espacio de atributos utilizando *backtracking* [35]. Primero se mueve en el espacio de de búsqueda realizando cambios locales al subconjunto actual pero a diferencia del método *hill climbing* [3] si el camino comienza a parecer menos prometedor puede realizar *backtracking* a un mejor subconjunto anterior y continuar la búsqueda desde allí. Si se cuenta con el tiempo y los recursos necesarios, *best first* explora todo el espacio de búsqueda utilizando un grafo por lo que se toma como condición de parada que no haya mejora luego de un determinado número de nodos consecutivos. La forma de selección de atributos puede ser *forward*, *backward* o *bi-directional*. En *forward selection* se incorporan progresivamente las variables comenzando con un conjunto vacío y es computacionalmente el método más eficiente. *Backward selection* consiste en ir eliminando atributos progresivamente y si bien es más caro computacionalmente puede considerar variables débiles individualmente pero fuertes en conjunto. Por último, en *bi-directional selection* se eliminan o agregan atributos partiendo de un subconjunto inicial.

Para evaluar cada subconjunto, se puede optar por métodos *filter* o *wrapper*. El problema de elegir un método *wrapper*, es decir, utilizar la performance de un algoritmo de aprendizaje para la evaluación del subconjunto, es que el costo computacional requerido es mucho mayor [37]. Por este motivo no se utiliza cuando el número de atributos es demasiado extenso como en el problema que se está resolviendo[4]. Tomamos como evaluador del subconjunto el algoritmo *correlation based feature selection* [35], de tipo *filter*, que evalúa el mérito de

[3] Método matemático de optimización utilizado en búsquedas locales

[4] No encontramos un algoritmo de aprendizaje que finalizara exitosamente su ejecución con un método *wrapper* para la evaluación del subconjunto

Gain Ratio	***Atributo***
0,085	TAR_MENOR_MINIMO_SEMANA
0,0815	TAR_MENOR_AVERAGE_SEMANA
0,0792	TAR_MENOR_MAXIMO_SEMANA
0,0776	TAR_MAYOR_CAJ_MES
0,0759	TAR_MENOR_AVERAGE_MES
0,075	TAR_MENOR_MINIMO_MES
0,0659	TAR_MENOR_AVERAGE_DIA
0,0659	TAR_MENOR_MINIMO_DIA
0,0653	TAR_MENOR_MAXIMO_DIA
0,0653	TAR_MENOR_DIA
0,0594	TAR_MENOR_MES
0,0565	TAR_MAYOR_DIA
0,0549	TAR_MAYOR_CANT_DIA
0,0549	TAR_MENOR_MAXIMO_MES
0,0538	TAR_MAYOR_CAJ_CANT_DIA

Cuadro 4.5: *Gain Ratio - Primeros 15 atributos en el ranking*

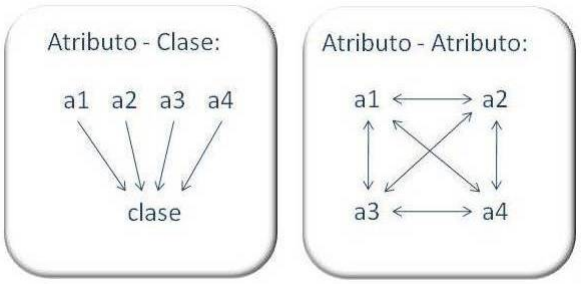

Figura 4.1: Correlación de atributos

un subconjunto de atributos considerando la habilidad predictiva individual de cada atributo junto al grado de redundancia entre ellos. Se prefieren los subconjuntos de atributos fuertemente correlacionados con la clase pero débilmente interrelacionados entre ellos.

Los resultados obtenidos se muestran en los cuadros 4.8 y 4.9. Los subconjuntos utilizando *forward* y *bi-directional selection* son exactamente iguales, mientras que con *backward selection* existe un atributo en cada subconjunto que no se encuentra en la unión {`TAR_SIMPLE_SEMANA, TAR_MAYOR_SEMANA`}.

Si comparamos con los resultados obtenidos con algoritmos de *ranking*, observamos que se introducen tres nuevos atributos, ausentes previamente {`COM_MENOR_SEMANA, TAR_MAYOR_MAXIMO_MES, TARMCC_SIMPLE_CANT_DIA`}.

χ^2	***Atributo***
5196,734	TAR_MAYOR_CANT_DIA
5083,387	TAR_SIMPLE_DIA
4785,616	TAR_MAYOR_DIA
4699,485	TAR_SIMPLE_SEMANA
4680,495	TAR_MAYOR_SEMANA
4312,774	TARMCC_SIMPLE_DIA
4274,379	TAR_SIMPLE_CANT_DIA
4198,279	TARMCC_SIMPLE_SEMANA
3961,591	TAR_MAYOR_CANT_SEMANA
3806,912	TARCOM_SIMPLE_DIA
3705,709	TARMCC_SIMPLE_MAXIMO_SEMANA
3664,393	TARCOM_SIMPLE_SEMANA
3659,583	TAR_SIMPLE_MAXIMO_SEMANA
3615,671	TAR_MAYOR_MAXIMO_SEMANA
3598,116	TARCOM_SIMPLE_MAXIMO_SEMANA

Cuadro 4.6: χ^2 - *Primeros 15 atributos en el ranking*

4.4.3. Conclusiones

Si consideramos todos los cuadros de los resultados obtenidos y tomamos aquellos atributos presentes en dos o más cuadros, obtenemos el conjunto de la tabla 4.10.

En este conjunto observamos por ejemplo, que para las particiones por tarjeta (`TAR_SIMPLE_DIA, TAR_SIMPLE_MAXIMO_SEMANA, TAR_SIMPLE_SEMANA`), por tarjeta y por comercio (`TARCOM_SIMPLE_DIA, TARCOM_SIMPLE_MAXIMO_SEMANA, TARCOM_SIMPLE_SEMANA`), y por tarjeta filtrado por cajero, (`TARMCC_SIMPLE_DIA, TARMCC_SIMPLE_MAXIMO_SEMANA, TARMCC_SIMPLE_SEMANA`) obtuvimos los mismos atributos: monto acumulado por día, por semana y máximo monto por semana. Estos resultados se encuentran dentro de lo esperado, ya que esperábamos que el monto acumulado jugara un rol importante. Además el monto acumulado por semana podría indicar que el fraude se encuentra distribuido en la semana de forma de pasar el control detectado por los atributos diarios. Por otra parte, el máximo monto por semana detectaría actividad fuera del flujo normal de transacciones.

En la partición por tarjeta se encuentran también los atributos `TAR_MAYOR_DIA, TAR_SIMPLE_CANT_DIA, TAR_MAYOR_CANT_DIA, TAR_MAYOR_MAXIMO_SEMANA, TAR_MAYOR_SEMANA`. Los dos primeros acumulan la cantidad de transacciones por tarjeta y por día y por tarjeta y por semana, mientras que los dos últimos acumula los montos mayores a 500 así como el máximo monto mayor a ese valor para una misma semana.

El atributo que calcula el promedio de montos de transacciones menores a 10 dólares en principio podría parecer poco importante, pero fue retornado como un atributo de interés. La razón de esto posiblemente esté vinculada a que el

1. Comenzar con la lista OPEN conteniendo el estado inicial y la lista CLOSED vacía. $BEST \leftarrow$ estado inicial
2. Sea $s = argmaxe(x)$. (Tomar el estado de OPEN con la mayor evaluación)
3. Remover s de OPEN y agregarlo a CLOSED
4. Si $e(s) \geq e(BEST)$, entonces $BEST \leftarrow s$
5. Para cada hijo t de s que no se encuentra en la lista OPEN o CLOSED, evaluarlo y agregarlo a OPEN
6. Si BEST cambia en las últimas expansiones, ir a 2
7. Retornar BEST

Cuadro 4.7: Algoritmo de búsqueda *best first* [35]

Atributo
COM_MENOR_SEMANA
TAR_SIMPLE_CANT_DIA
TAR_SIMPLE_SEMANA
TAR_MENOR_AVERAGE_MES
TAR_MAYOR_CANT_DIA
TAR_MAYOR_DIA
TAR_MAYOR_MAXIMO_MES
TARMCC_SIMPLE_CANT_DIA

Cuadro 4.8: Resultado *best first - forward* utilizando *correlation based feature selection*

atributo sea de utilidad cuando se combina con otros atributos. Por ejemplo, un patrón podría darse cuando hay compras de valores que superan los 500 dólares pero no se han hecho compras por valores bajos, en cuyo caso el promedio de montos de transacciones menores a 10 dólares sería cero.

Atributo
COM_MENOR_SEMANA
TAR_SIMPLE_CANT_DIA
TAR_MENOR_AVERAGE_MES
TAR_MAYOR_CANT_DIA
TAR_MAYOR_DIA
TAR_MAYOR_SEMANA
TAR_MAYOR_MAXIMO_MES
TARMCC_SIMPLE_CANT_DIA

Cuadro 4.9: Resultado *best first - backward* utilizando *correlation based feature selection*

Atributo
TAR_MAYOR_CANT_DIA
TAR_MAYOR_DIA
TAR_MAYOR_MAXIMO_SEMANA
TAR_MAYOR_SEMANA
TAR_MENOR_AVERAGE_MES
TAR_SIMPLE_CANT_DIA
TAR_SIMPLE_DIA
TAR_SIMPLE_MAXIMO_SEMANA
TAR_SIMPLE_SEMANA
TARCOM_SIMPLE_DIA
TARCOM_SIMPLE_MAXIMO_SEMANA
TARCOM_SIMPLE_SEMANA
TARMCC_SIMPLE_DIA
TARMCC_SIMPLE_MAXIMO_SEMANA
TARMCC_SIMPLE_SEMANA

Cuadro 4.10: Atributos presentes en más de un resultado

Capítulo 5

Entrenamiento

Para realizar el entrenamiento primero etiquetamos las transacciones y calculamos los atributos complejos, ver sección 4.2 junto con apéndices C y D para mayor información. Posteriormente, realizamos *sampling* sobre este conjunto para poder trabajar con los algoritmos de clasificación, como describimos en la sección 4.3.

Partimos de forma aleatoria el conjunto obtenido del *sampling* en tres subconjuntos. Utilizamos el 12 % de las instancias para la selección de atributos, detallada en la sección 4.4, el 48 % como conjunto de entrenamiento para la generación de reglas y aprendizaje del puntaje de riesgo y reservamos el restante 40 % como conjunto de evaluación. Se puede observar una diferencia significativa en la proporción de transacciones fraudulentas entre ambos conjuntos, lo que es causado por realizar el muestreo aleatorio sobre un conjunto sumamente desbalanceado.

El conjunto de entrenamiento contiene 104 775 instancias de las cuales 14 394 están etiquetadas como fraude (un 13,74 %). El conjunto de evaluación contiene 88 000 instancias de las cuales 2895 están etiquetadas como fraude (un 3,29 %).

Como entrada de los distintos algoritmos utilizamos los conjuntos de entrenamiento y evaluación descritos anteriormente tomando distintos subconjuntos de atributos. En primer lugar utilizamos el conjunto de atributos completo y luego los subconjuntos obtenidos en la etapa de selección de atributos, para poder comparar los resultados del aprendizaje entre los conjuntos reducidos y el conjunto original. Buscamos determinar cómo influye la variación de los atributos de entrada en el proceso de aprendizaje. Utilizar los mismos conjuntos para los distintos algoritmos facilita comparar los resultados entre sí. Para cada conjunto ejecutamos los algoritmos con distintos parámetros con el objetivo de encontrar la mejor configuración de éstos para la resolución el problema en concreto.

La implementación de los algoritmos utilizados pertenecen a la plataforma *WEKA (Waikato Environment for Knowledge Analysis)*, versión 3.5.8 [22]. WEKA es una colección de algoritmos de aprendizaje automático y minería de datos escritos en Java, desarrollada en la Universidad de Waikato. Es un software libre distribuido bajo licencia GNU-GPL que contiene herramientas para pre-

procesamiento de datos, clasificación, regresión, *clustering*, reglas de asociación y visualización de resultados [53]. Por más información sobre WEKA referirse al Apéndice G.

Es de interés aclarar que en el contexto de este proyecto, un clasificador trivial que simplemente clasificara como negativas a todas las instancias clasificaría correctamente el 96,71 % del conjunto de evaluación por lo que las instancias clasificadas correctamente no es una medida que por sí misma indique un buen clasificador, por otra parte, obtendría un 0 % de precisión considerando únicamente las transacciones fraudulentas y un 0 % de falso positivo. Debido a esta realidad, mostraremos solamente las medidas obtenidas para la clase *Fraude*.

Una precisión baja significaría que una gran cantidad de transacciones de fraude pasarían sin ser detectadas. Por otro lado un falso positivo elevado implica que se generen alertas por un gran número de transacciones legítimas. Por esta causa, para el propósito de evaluar la efectividad de los diferentes modelos y configuraciones, nos apoyamos en la medida-f, permitiendo mantener una relación equilibrada entre precisión y recuperación. Para el caso particular de los algoritmos con clasificadores de *scorer*, utilizamos medidas de curva de ROC y su área debajo de la curva.

El resto del capítulo está organizado de la siguiente manera: las secciones 5.1, 5.2 y 5.3 documentan las pruebas realizadas y resultados obtenidos con los árboles de decisión, las redes neuronales y *random forest* respectivamente; la sección 5.4 documenta el estudio del umbral de activación y la cobertura del *scorer* construido. En las conclusiones del capítulo 7 analizamos los resultados obtenidos en cada caso.

5.1. Árboles de decisión

Utilizamos dos implementaciones de árboles disponibles en Weka, el *J48* y el *RandomTree* utilizando distintas configuraciones [22].

El J48 es una implementación del C4.5, cuyo algoritmo soporta datos nominales y continuos de entrada y no puede ser entrenado incrementalmente. Se puede optar por utilizar poda o no, se puede variar el umbral de confianza para la poda y el número mínimo de instancias permitido en cada hoja para obtener árboles más eficientes. También se puede optar por particionar los atributos nominales en forma binaria.

El RandomTree considera un subconjunto de K atributos elegidos al azar para particionar en cada nodo; utilizamos K=3 para nuestras pruebas. En el cuadro 5.1 mostramos las distintas configuraciones utilizadas en las pruebas.

Realizamos varias pruebas combinando los distintos conjuntos de entrada con las distintas configuraciones de árboles de decisión.

Nombre	Opciones
J48	Opciones por defecto
J48 Unpruned	No se realiza *prunning*
J48 Split	Se utiliza para atributos nominales
Random Tree	Se consideran 3 atributos al azar para particionar

Cuadro 5.1: Opciones de árboles utilizadas en las pruebas

5.1.1. Resultados obtenidos

Teniendo en cuenta la clase positiva obtuvimos una precisión entre el 17,93 % y el 47,91 % y una recuperación entre 32,02 % y 46,98 %, siendo todos los resultados variados entre los clasificadores obtenidos y claramente superiores al clasificador trivial que obtiene 0 % de recuperación, ya que no clasifica ninguna instancia como positiva.

Como se observa en el cuadro 5.2 la mejor medida-f obtenida para la clase positiva es de 42 %, esta medida es el doble de la obtenida por el proyecto anterior [11].

Árbol	Atributos	Prec.	Rec.	Medida-f
J48	Chi Sqr	0,403	0,394	0,398
J48	Gain Ratio	0,475	0,365	**0,413**
J48	Info Gain	0,405	0,394	0,399
J48	Subset	0,389	0,397	0,393
J48	Total	0,346	0,391	0,367
J48 Split	Chi Sqr	0,403	0,394	0,398
J48 Split	Gain Ratio	0,475	0,365	**0,413**
J48 Split	Info Gain	0,405	0,394	0,399
J48 Split	Subset	0,389	0,398	0,393
J48 Split	Total	0,316	**0,469**	0,378
J48 Unpruned	Chi Sqr	0,356	0,406	0,379
J48 Unpruned	Gain Ratio	**0,479**	0,356	0,408
J48 Unpruned	Info Gain	0,416	0,392	0,403
J48 Unpruned	Subset	0,400	0,400	0,400
J48 Unpruned	Total	0,194	0,395	0,260
Random	Chi Sqr	0,201	0,450	0,278
Random	Gain Ratio	0,268	0,320	0,292
Random	Info Gain	0,201	0,428	0,273
Random	Subset	0,179	0,426	0,252
Random	Total	0,212	**0,469**	0,292

Cuadro 5.2: Precisión, Recuperación y Medida-f para la clase positiva

Del cuadro 5.3 se puede deducir que al realizar poda para evitar el sobreajuste obtuvimos árboles significativamente más pequeños (con menor cantidad de nodos) sin reducir demasiado su efectividad. Por ejemplo, para el conjunto formado con la totalidad de atributos obtuvimos un árbol con 24 326 nodos el cual tiene una medida-f para la clase positiva de 26,07 %. Con el mismo conjunto de atributos al aplicar poda en el entrenamiento obtuvimos un árbol con 7811

nodos el cual tiene una medida-f de 36,78 %, en este caso, todas las medidas de evaluación mejoran en el árbol podado. También se puede observar que el entrenamiento con el subconjunto *Gain Ratio* también produce los árboles más pequeños mejorando la medida-f, seguido por los subconjuntos *Info Gain* y *Chi Sqr* y que para todos los tipos de árboles el entrenamiento con el conjunto total de atributos produce árboles significativamente más grandes.

Atributos	Árbol	Nodos	Nodos Hojas	Medida-f
Total	J48	7 811	6 309	0,368
Total	J48 Split	4 533	2 267	0,378
Total	J48 Unpruned	24 326	20 398	0,261
Total	Random	74 169	62 414	0,292
Chi Sqr	J48	721	361	0,399
Chi Sqr	J48 Split	721	361	0,399
Chi Sqr	J48 Unpruned	1 475	738	0,380
Chi Sqr	Random	24 333	12 167	0,278
Gain Ratio	J48	**59**	**30**	**0,413**
Gain Ratio	J48 Split	**59**	**30**	**0,413**
Gain Ratio	J48 Unpruned	261	131	0,410
Gain Ratio	Random	20 573	10 287	0,292
Info Gain	J48	437	219	0,400
Info Gain	J48 Split	437	219	0,400
Info Gain	J48 Unpruned	599	300	0,404
Info Gain	Random	23 871	11 936	0,274
Subset	J48	581	291	0,393
Subset	J48 Split	581	291	0,393
Subset	J48 Unpruned	1 231	616	0,386
Subset	Random	28 223	14 112	0,252

Cuadro 5.3: Tamaño de árboles obtenidos

En el apéndice F detallamos para cada árbol la matriz de confusión obtenida. El mayor TP obtenido es 1360 y corresponde al árbol *J48 Split* con atributos *Total* o al *Random* con atributos *Subset*. El mayor TN obtenido es 83 983 y corresponde al árbol *J48 Unpruned* con atributos *Gain Ratio*. El árbol *Random* con atributos *Subset* produce la mayor cantidad de falsas alarmas (5641) y es seguido por el mismo algoritmo con los atributos *Chi Sqr*, *Total* e *Info Gain*. Los árboles entrenados con el conjunto de atributos *Gain Ratio* producen la mayor cantidad de falsos negativos, es decir, cantidad de fraudes clasificados como no fraudes. Al mismo tiempo el conjunto presenta la menor cantidad de falsas alarmas, lo que genera los más altos niveles de precisión, capaces de compensar sus bajos niveles en recuperación.

Reglas obtenidas

Transformamos los árboles obtenidos en las pruebas en conjuntos de reglas *if-then-else*. De las reglas obtenidas se muestran en el cuadro 5.4 las cinco reglas que clasifican instancias como Fraude con mayor medida-f. Para cada regla se especifica la prueba a la que pertenece (árbol y conjunto de atributos de entrada), la cantidad de instancias alcanzadas, la cantidad de instancias mal clasificadas, la precisión, la recuperación y la medida-f de la regla.

Regla						
Árbol	Atributos	Total	FP	Precisión	Recuperación	Medida-f
TAR_MAYOR_CANT_DIA > 1.0 AND TAR_MAYOR_DIA > 4830.0 AND TAR_MAYOR_CAJ_CANT_DIA <= 0.0 AND TAR_MAYOR_CANT_DIA <= 3.0 AND TAR_MENOR_MINIMO_DIA <= 0.0 AND TAR_MAYOR_DIA <= 42119.59						
J48 Split	Gain Ratio	3556	953	0,732	0,181	0,290
TAR_MAYOR_CANT_DIA > 1.0 AND TAR_SIMPLE_SEMANA > 6002.35 AND TAR_SIMPLE_CANT_DIA <= 4.0 AND TAR_MAYOR_DIA > 7216.98 AND TAR_MAYOR_MAXIMO_MES <= 31545.78 AND TAR_MENOR_AVERAGE_MES <= 0.01						
J48 Split	Subset	2533	575	0,773	0,136	0,231
TAR_MAYOR_CANT_DIA > 1 AND TAR_MAYOR_DIA > 4830 AND TAR_MAYOR_CAJ_CANT_DIA <= 0 AND TAR_MAYOR_CANT_DIA <= 3 AND TAR_MENOR_MINIMO_DIA <= 0 AND TAR_MAYOR_DIA > 7216.98 AND TAR_MAYOR_DIA <= 42119.59						
J48 Unpruned	Gain Ratio	2381	527	0,779	0,129	0,221
TAR_MAYOR_CANT_DIA > 1.0 AND TAR_MAYOR_DIA > 4830.0 AND TAR_MAYOR_CAJ_CANT_DIA <= 0.0 AND TAR_MAYOR_CANT_DIA > 3.0 AND TAR_MAYOR_CANT_DIA <= 12.0						
J48 Split	Gain Ratio	1781	249	0,860	0,106	0,189
TAR_MAYOR_CANT_DIA > 1.0 AND MCC != 5968 AND TAR_SIMPLE_SEMANA > 6277.13 AND COM_SIMPLE_CANT_SEMANA <= 671.0 AND TAR_MAYOR_CAJ_CANT_DIA <= 0.0 AND MCC != 5960 AND MCC != 3501 AND TARCOM_SIMPLE_MAXIMO_MES <= 31527.39 AND COM_SIMPLE_CANT_DIA <= 139.0 AND COM_SIMPLE_MAXIMO_MES <= 35385.42 AND TAR_MAYOR_CANT_DIA > 2.0 AND MCC != 3517 AND MCC != 5950 AND MCC != 6300 AND MCC != 7512 AND MCC != 8220 AND TARCOM_SIMPLE_MES <= 43868.42 AND MCC != 3007 AND MCC != 3366 AND MCC != 3387 AND MCC != 3513 AND MCC != 7997 AND MCC != 9311 AND MCC != 7011 AND MCC != 3357 AND MCC != 4900 AND MCC != 8071 AND MCC != 3076 AND MCC != 5511 AND MCC != 5969 AND TAR_SIMPLE_DIA > 7639.28 AND MCC != 8062 AND TAR_MAYOR_CAJ_MINIMO_DIA <= 0.0 AND MCC != 5611 AND MCC != 5621 AND MCC != 5300 AND MCC != 5812 AND TAR_MENOR_MINIMO_DIA <= 0.5 AND COM_MAYOR_CANT_DIA <= 15.0						
J48 Unpruned	Gain Ratio	2381	527	0,779	0,129	0,221

Cuadro 5.4: Reglas que clasifican instancias como Fraude con mayor medida-f

En el cuadro podemos observar que todas las reglas comienzan con la condición TAR_MAYOR_CANT_DIA > 1.0 lo que identifica autorizaciones cuyas tarjetas hayan realizado en el día más de una compra de 500 dólares.

En la primer regla se valida específicamente que la transacción de mayor monto del día supere los 4830 dólares, se hayan realizado en total entre dos y tres transacciones en la cuenta por más de 500 dólares pero que ninguna de ellas sea realizada mediante cajero automático o retiro en efectivo y que no existan movimientos menores a 10 dólares. De esta forma, se identifican tarjetas con pocos movimientos y de montos altos sin utilizar un cajero automático o realizar retiros en efectivo. El excluir cajero y efectivo posiblemente esté relacionado con la necesidad de un pin u otro mecanismo de autenticación que dificulta que la transacción sea fraudulenta. Esta regla se relaciona con el patrón de fraude de compras por montos excesivos.

En la segunda regla, se valida además de que exista más de una transacción

por monto mayor a 500 dólares en la tarjeta, que la máxima transacción en el día supere los 7216 dólares en un máximo de cuatro transacciones diarias. Además que en la semana el monto total supere los 6002,35 dólares (lo que es redundante con la condición del monto de una transacción en el día). Se valida además, que la tarjeta no haya tenido movimientos por menos de 10 dólares en el mes. Nuevamente se apunta al mismo objetivo que en la regla anterior, con distintos umbrales, períodos de tiempo y cantidad de transacciones, pero nuevamente detectando un patrón de movimientos altos en pocas transacciones. Esto situación se repite nuevamente en la tercera regla.

La cuarta regla mantiene umbrales de monto y períodos de tiempo como las anteriores, pero tiene la peculiaridad de ampliar notoriamente la cantidad de movimientos, generando alerta para transacciones que tengan de cuatro a doce movimientos en la tarjeta que superen los 500 dólares. En este caso, las transacciones no pueden ser de cajero automático o efectivo. Aquí se apunta a montos totales altos, en muchas transacciones que no sean efectivo o cajero, detectando un patrón donde se busca obtener dinero rápidamente pero intentando pasar desapercibido haciéndolo en muchas transacciones en un mismo día. El valor doce posiblemente denote simplemente que el sistema encontró ejemplos de este patrón de fraude de hasta 12 movimientos y no más, por lo que esta restricción podría ser eliminada por un analista a la hora de incorporar la regla al *Detection Engine*.

La última regla es la más compleja de interpretar, nuevamente validándose umbrales en montos semanales y diarios distintos de cajeros automáticos o efectivo, pero con la particularidad que aplica solamente a determinados MCC, filtrando cerca del 30 % del total de transacciones. Los MCC filtrados abarcan rubros en hotelería, venta de seguros, alquiler de autos, universidades privadas, etc, donde posiblemente el patrón ocurra de forma significativa en transacciones no fraudulentas y por lo tanto sea excluido de la alerta de fraude.

Dentro de las reglas obtenidas se pueden observar condiciones redundantes o innecesarias que pueden ser analizadas y eliminadas por un analista antes de ingresarlas en el *Detection Engine*. A su vez, existen colisiones entre las reglas obtenidas en los distintos modelos. Por ejemplo, reglas que identifican tarjetas con más de una autorización y reglas que identifican tarjetas que tienen entre cuatro y doce autorizaciones. Un análisis y reestructuración de las reglas puede resultar en mayor efectividad y más fácil mantenimiento del modelo de detección.

5.2. Redes Neuronales Artificiales

El perceptrón multicapa puede ser implementado de forma *Batch* u *Online*. Cuando es *Batch* todos los patrones son presentados a la red, el error total es calculado y los pesos son actualizados en una etapa posterior llamada un *epoch* de entrenamiento. Cuando es *Online*, los pesos son actualizados luego de cada presentación individual. La ventaja de este último sobre el primero es que converge más rápidamente (empíricamente), pero existe la posibilidad teórica de entrar en un ciclo de cambios repetidos (esto se intenta evitar presentando los

ejemplos de entrenamiento de forma aleatoria). En cambio el algoritmo *Batch* no presenta estos problemas, siendo más preciso [9].

El algoritmo que utilizamos es *Batch*, lo que da una mayor estabilidad. La cantidad de *epochs* por defecto es de 500, por lo que realizamos entrenamientos con este valor y con uno seis veces mayor (3.000 *epochs*) para estudiar en que grado esta variable afecta el resultado del algoritmo.

Para mejorar la velocidad de convergencia se puede modificar la tasa de aprendizaje en cada iteración. Si la tasa es demasiado grande, el algoritmo puede volverse inestable oscilando sin alcanzar la solución. Para evitar esto, se utiliza un término en el ajuste de pesos llamado momento. Con este término se logra que el ajuste de pesos sea más grande si es en la dirección de los cambios previos y más pequeño si es en una dirección distinta, buscando evitar que el algoritmo quede atascado en oscilaciones. Se recomienda partir de una tasa de aprendizaje pequeña, en el orden de 0,1 [9].

El valor por defecto de la tasa de aprendizaje es de 0,3 y del momento es 0,2. Además de éstos valores, utilizamos una tasa de aprendizaje de 0,06 y un momento de 0,04 (cinco veces menor manteniendo la proporción).

En largos entrenamientos los perceptrones multicapa pueden adolecer de parálisis de red. Esto ocurre cuando los pesos de la red toman valores muy altos (positivos o negativos) ocasionando que el total entrante a un nodo oculto o de salida también sea muy alto. En esas situaciones la función de activación toma valores muy cercanos a 0 o a 1 ocasionando que los ajustes sean desproporcionalmente cercanos a cero en comparación a los pesos, por lo que el proceso de aprendizaje queda virtualmente detenido [5].

Otro riesgo ocurre cuando la gráfica de error de una red compleja presenta muchas oscilaciones, en este caso, la red puede quedar atrapada en un mínimo local. Una posibilidad para evitar esta situación es incrementar el número de capas ocultas, aumentando el tamaño del espacio de error.

Utilizar demasiadas capas ocultas puede llevar a un sobreajuste donde se aprende a encajar exactamente sobre el conjunto de entrenamiento y no el patrón que se puede deducir a partir del conjunto, obteniéndose poco error en el conjunto de entrenamiento pero un error mucho mayor en el conjunto de prueba.

Ya que la complejidad interna de la red neuronal a entrenar depende en gran medida de los datos, no hay un método universal para seleccionar la cantidad de neuronas de las capas ocultas. El criterio general es que demasiadas neuronas y el sistema memoriza en vez de aprender, demasiado pocas y no será suficiente. Partiendo de que los atributos de entrada a utilizar pasaron por un proceso previo de selección de atributos, podemos asumir que cada uno aporta información, por lo que optamos porque la cantidad de nodos en las capas ocultas mantenga relación con la cantidad de atributos y clases de clasificación (que en este caso son dos). Además, para evitar tanto mínimos locales como sobreajuste variamos las cardinalidades y dimensiones, realizando entrenamientos con una capa oculta de (atributos + clases)/2 nodos, una capa oculta de (atributos + clases)

Variable	Valores
Epoch	500 3000
Tasa de Aprendizaje y Momento	0,3 y 0,2 0,06 y 0,04
Capas y nodos	(atributos + clases)/2 (atributos + clases) [(atributos + clases),(atributos + clases)/2]

Cuadro 5.5: Variables y valores utilizados en el entrenamiento del Perceptrón Multicapa

y dos capas ocultas de (atributos + clases) nodos y (atributos + clases)/2 nodos.

La tabla 5.5 resume las variables y conjuntos de entrada que utilizamos.

5.2.1. Resultados obtenidos

Dado que el objetivo es detectar fraudes, consideramos la clase positiva de especial interés. Para esta clase, obtuvimos una precisión entre el 33,9 % y el 54,6 % y una recuperación entre 25,7 % y 42,5 %, siendo todos los resultados superiores al clasificador trivial y variados entre los clasificadores obtenidos.

El cuadro 5.6 muestra las mejores estadísticas obtenidas, pudiéndose observar que la diferencia de área bajo la curva de ROC entre las distintas configuraciones es pequeña, con valores entre el 81 % y 85 %. En el apéndice F se listan cuadros completos para las diferentes configuraciones utilizadas.

Atributos	Neurs.	Epochs	LR	AUC	Prec.	Rec.	Med.-f
ChiSQ	t,a	3000	0,3	**0,852**	0,362	0,399	0,379
GainRatio	t,a	3000	0,3	0,824	0,491	0,347	**0,407**
GainRatio	t,a	500	0,3	0,82	**0,546**	0,257	0,349
SubSetBF	t	3000	0,3	0,848	0,344	**0,425**	0,38
SubSetBF	t,a	3000	0,06	**0,852**	0,366	0,399	0,382

Cuadro 5.6: Mejores resultados de Perceptrón Multicapa para cada medida
a =(atributos + clases)/2, t = (atributos + clases), LR = Learning Rate
AUC = Area Under Roc Curve

Se observa una mejora significativa en la medida-f para fraude cuando utilizamos 3000 *epochs* en vez de 500. Un aumento en la cantidad de *epochs* implica un aumento equivalente de iteraciones del algoritmo, lo que se traduce en un mayor tiempo de entrenamiento. Esto es notorio en la gráfica en la figura 5.1 donde se observa como un entrenamiento realizado con 500 *epochs* no supera los 30 minutos mientras que uno realizado con 3000 *epochs* puede insumir hasta 160 minutos, haciendo evidente la relación directa entre cantidad de *epochs* y tiempo insumido. Se puede observar que la diferencia de tiempos de entrenamiento aumenta con la complejidad de la topología utilizada, reflejándose en la gráfica como un aumento de diferencias de tiempo de izquierda a derecha en cada subconjunto de atributos. De todas formas, consideramos que el tiempo insumido es tolerable, especialmente con el uso de selección de atributos, por lo

que a partir de ahora continuaremos trabajando exclusivamente con 3000 *epochs*.

No se observa diferencia significativa entre una tasa de aprendizaje de 0,3 y momento de 0,2 y una tasa de 0,06 y momento 0,04. Dado que una mayor tasa de aprendizaje aumenta la velocidad de convergencia del algoritmo, sumado un mejor desempeño en nuestras pruebas, es mejor opción en futuras pruebas utilizar una tasa de 0,3 y un momento de 0,2.

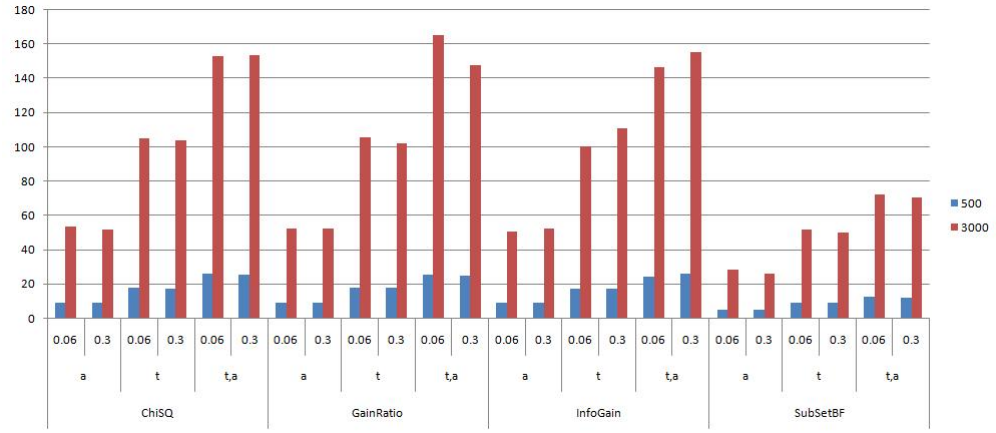

Figura 5.1: Gráfica comparativa tiempo de entrenamiento en minutos entre 500 y 3000 epochs

En el apéndice F se detalla para cada entrenamiento de 3000 *epochs* y tasa de aprendizaje 0,3 la matriz de confusión obtenida. La mayor cantidad de fraudes detectados (FF) es 1230 y corresponde al conjunto de entrenamiento *Subset*.

De los distintos conjuntos de entrenamiento que hemos utilizado, el que obtiene mejor medida-f para la clase positiva es *Gain Ratio* con un valor de 40,7 % en una configuración de 3000 *epochs*, tasa de aprendizaje 0,3 , momento 0,2 y una topología de dos capas ocultas. En la figura 5.2 se puede observar su gráfica de ROC.

Podemos observar que los resultados del perceptrón multicapa son dependientes de su configuración, pudiéndose obtener mejoras adecuando los parámetros al conjunto de entrenamiento tal como lo hemos mostrado empíricamente en este capítulo. Para los distintos conjuntos de entrenamiento se podría buscar obtener mejores resultados con una mayor cantidad de *epochs*, pero esto se traduciría directamente en mayores tiempos de entrenamiento. Una topología más compleja (más nodos y/o más capas) no necesariamente implica mejores resultados, aumentando el tiempo de entrenamiento y posiblemente causando sobreajuste.

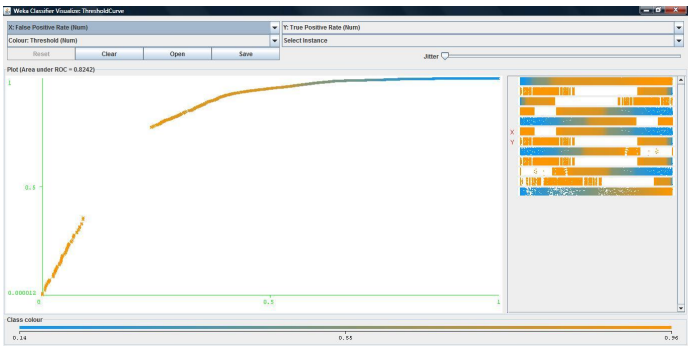

Figura 5.2: Gráfica de ROC para la configuración para el conjunto de entrenamiento Gain Ratio con 3000 *epochs*, tasa de aprendizaje 0,3 , momento 0,2 y topología de 2 capas ocultas.

5.3. Random Forest

La implementación de *random forest* que provee Weka 3.5.8 es limitada en el sentido que no incluye ciertas características del algoritmo como ser: determinar la importancia de variables e interacciones entre variables, calcular proximidades o encontrar datos incorrectamente etiquetados entre otras, descritas en el artículo de Brieman [34].

En la versión que utilizamos, el algoritmo permite la configuración de la cantidad de árboles a construir en el bosque aleatorio y la cantidad de atributos a considerar para la construcción de cada árbol aleatorio. La cantidad de atributos tomados al azar para generar cada árbol fue $\log M + 1$, siendo M el total de atributos de entrada. Cada árbol creado toma los atributos independientemente.

Para la cantidad de árboles del bosque, optamos por realizar tres pruebas por conjunto de entrenamiento, con 25, 50 y 100 árboles. Por limitaciones de capacidad y tiempo de procesamiento no pudimos realizar pruebas con mayor cantidad de árboles en el bosque. La intención es poder detectar la mejora en el rendimiento del algoritmo al agregar árboles en el bosque.

5.3.1. Resultados obtenidos

En el cuadro 5.7 se muestran los resultados obtenidos por conjunto y cantidad de árboles del bosque. En el apéndice F se pueden ver las matrices de confusión generadas. En estos cuadros se puede observar que no se encuentran los resultados correspondientes a todo el subconjunto de atributos ya que por limitaciones de memoria no fue posible ejecutar el algoritmo con todos los atributos como entrada.

Si comparamos los resultados por conjunto de entrenamiento, podemos ver que el aumento en la cantidad de árboles generados en el bosque no aumentó de manera significativa la variación de los resultados para el resto de las medidas. Por

Conjunto	Árb.	AUC	OOB	Prec.	Rec.	Med.-F
Info Gain	25	0,83	0,1136	0,288	0,443	0,349
Info Gain	50	0,835	0,1104	0,295	0,435	0,352
Info Gain	100	0,839	0,109	0,302	0,437	0,357
Gain Ratio	25	0,709	0,1242	0,283	0,328	0,304
Gain Ratio	50	0,713	0,1246	0,273	0,328	0,298
Gain Ratio	100	0,718	**0,1249**	0,281	0,326	0,302
Chi Sq	25	0,832	0,114	0,295	**0,447**	0,355
Chi Sq	50	0,838	0,11	**0,307**	0,441	**0,362**
Chi Sq	100	**0,842**	0,1085	0,305	0,44	0,36
Subset	25	0,828	0,1187	0,273	0,429	0,334
Subset	50	0,835	0,1155	0,286	0,428	0,343
Subset	100	0,839	0,132	0,284	0,428	0,341

Cuadro 5.7: Estadísticas para *random forest*
AUC = Area Under Roc Curve, OOB = Out of Bag Error

restricciones de memoria, no pudimos correr los algoritmos con mayor cantidad de árboles generados para poder determinar si se darían cambios más relevantes en los resultados.

Por otra parte, el error de generalización (*out of bag error*) estuvo entre el 0,1085 y 0,1249, mientras que el área bajo la curva de ROC varió entre 0,709 y 0,842.

En cuanto al resto de medidas observadas, los resultados para la clase positiva no superan los obtenidos con Árboles de Decisión.

La matriz de confusión en el apéndice F muestra que el mayor número de TP es 1294, obtenido con el bosque aleatorio de 25 árboles con el subconjunto *Chi-Sqr*. Si consideramos los TN, el mayor número corresponde a 82 703 obtenido con *Gain Ratio* y 25 árboles; sin embargo, es para este mismo conjunto que obtuvimos casi el menor número de TP. En cuanto a falsas alarmas, que serían aquellas instancias clasificadas como fraude cuando en realidad son legítimas, el mayor número está dado por 3302 para el subconjunto *Subset* con 25 árboles también.

Ahora, si consideramos la medida de la curva de ROC que toma en cuenta tanto los FP como los TP y que da una visión global de el desempeño del clasificador observamos que el mejor clasificador es el obtenido con el subconjunto *Chi-Sqr* y 100 árboles en el bosque aleatorio. Además, las medidas estadísticas de precisión, recuperación y medida-f para la clase positiva de este conjunto son las más altas respecto a los demás subconjuntos.

5.4. Estudio de umbral y cobertura

El sistema desarrollado (detallado en el capítulo 6) es capaz de utilizar uno o varios modelos como los analizados en las secciones anteriores y evaluar un conjunto de instancias sin clasificar, retornando un puntaje de riesgo por transacción

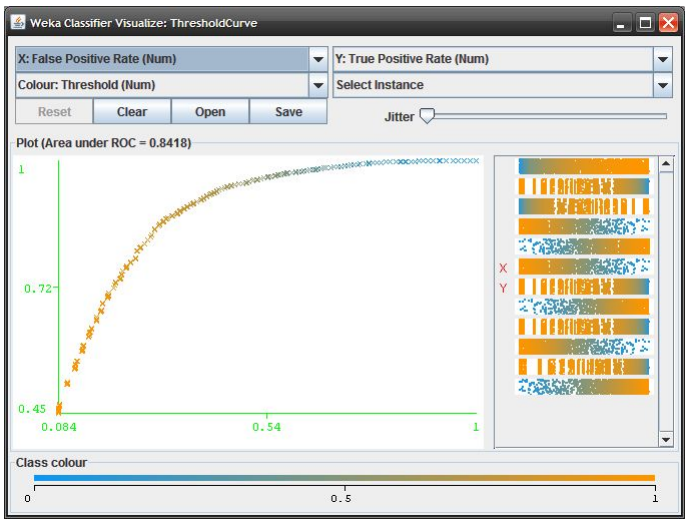

Figura 5.3: *Curva de ROC para Chi Sq con 100 árboles*

según cada modelo utilizado.

Las redes neuronales generan directamente el valor del puntaje en el intervalo [0,1] a partir de una función de mapeo. En cuanto a los árboles de decisión, durante la etapa de entrenamiento se registra en cada hoja la cantidad de instancias que clasifican correctamente e incorrectamente. A partir de estos valores, para cada instancia se obtiene el *score* dividiendo las instancias clasificadas correctamente sobre el total de instancias clasificadas por la hoja alcanzada por la instancia a clasificar. Por último, *random forest* utiliza el *score* obtenido por cada árbol de decisión aleatorio perteneciente al bosque calculando el puntaje individual de igual forma que árboles de decisión, para luego obtener y normalizar el total a partir de la suma de los resultados obtenidos.

Una pregunta natural es qué umbral utilizar para marcar una transacción como posible fraude o no. Como extremo, si tomamos un umbral de 1, todas las transacciones son consideradas fraudulentas (inclusive las legítimas) y por el contrario si el umbral es de 0, todas las transacciones son marcadas como legítimas, no detectándose ninguna fraudulenta.

En las secciones anteriores analizamos modelos de redes neuronales, árboles de decisión y *random forest*. En todos estos modelos, el umbral utilizado para su estudio es de 0,5, exactamente la mitad de la escala del puntaje.

Priorizar precisión sobre cobertura de transacciones positivas o viceversa depende de los intereses de quien ejecute el sistema. Ni siquiera es necesario que haya un único umbral, ya que pueden utilizarse varios representando distintas franjas de alerta para que determinadas transacciones sean analizadas con mayor o menor detalle.

Para estudiar los resultados generados, utilizamos el subconjunto de atributos obtenido por el algoritmo *Chi-cuadrado*, dado que observamos que genera buenos resultados para todos los modelos analizados.

Además de estos, utilizamos dos modelos que no son reales sino que surgen de combinar los puntajes de riesgo de los demás modelos. El puntaje de riesgo del modelo *Avg* es el promedio simple de los puntajes de los tres primeros modelos para cada instancia, mientras que *Max* es generado tomando el máximo puntaje.

5.4.1. Resultados obtenidos

Los modelos utilizados y sus valores para un umbral de 0,5 se pueden ver en el cuadro 5.8. El modelo *Max* al tomar el máximo puntaje prioriza detección de

Modelo	TP	FP	TN	FN	Recup.	Prec.	Med-F
Redes N.	1145	1899	83 206	1750	0,3955	0,3761	0,3856
Rand. F.	1299	3008	82 097	1596	0,4487	0,3016	0,3607
Árboles	1142	**1692**	**83 413**	1753	0,3945	0,4030	0,3987
Max	**1506**	4177	80 928	**1389**	**0,5202**	0,2650	0,3511
Avg	1169	1698	83 407	1726	0,4038	**0,4077**	**0,4058**

Cuadro 5.8: Resultado de modelos a partir de un umbral de 0,5

fraude por sobre precisión, bastando con que un único modelo genere un puntaje que supere el umbral para que la instancia sea marcada como fraude. En cambio el modelo *Avg* es una integración de valores de modelos más cauta, ya que genera un puntaje resultado del promedio sin priorizar valores. Si comparamos ambos modelos en el cuadro 5.8, *Max* cubre un 12 % más de casos positivos, pero con una precisión de casi la mitad que el modelo *Avg*.

En el cuadro 5.9 se pueden observar los valores cuando se disminuye el umbral en mas de un 50 %, en este caso la recuperación aumenta notoriamente. El modelo *Max* supera el 70 % de positivos correctamente clasificados, con una precisión del 12 %, mientras que el modelo *Avg* obtiene mejores valores de precisión. Observamos entonces que el modelo *Avg* es útil para situaciones donde se

Modelo	TP	FP	TN	FN	Recup.	Prec.	Med-F
Redes N.	1716	**7238**	**77 867**	1179	0,5927	**0,1916**	**0,2896**
Rand. F.	1875	9389	75 716	1020	0,6477	0,1665	0,2648
Arboles	1784	9626	75 479	1111	0,6162	0,1563	0,2494
Max	**2133**	14 875	70 230	**762**	**0,7368**	0,1254	0,2143
Avg	1827	7915	77 190	1068	0,6311	0,1875	0,2891

Cuadro 5.9: Resultado de modelos a partir de un umbral de 0,22

busca un equilibrio de precisión y cobertura. En cambio, sacrificando precisión, el modelo *Max* permite detectar casos positivos que no son detectados por todos los modelos, ya que obtiene una recuperación mayor a cualquiera de los demás

modelos que lo generan.

Otra forma de variar la relación entre cobertura de positivos y precisión consiste en variar el umbral. Como ejemplo, un umbral por debajo de 0,5 aumenta la cobertura y por lo tanto detecta más fraudes, pero como contrapartida la certeza de muchas de las transacciones marcadas como positivas es menor, lo que aumenta la cantidad de falsos positivos. Por lo tanto un umbral bajo aumenta la cobertura de positivos pero disminuye la precisión, disminuyendo la certeza que un fraude detectado realmente lo sea.

Esto puede verse claramente la figura 5.4 donde se grafica la recuperación (azul), precisión (rojo) y medida-f (verde) generado sobre distintos umbrales para el modelo *Avg*. Observamos claramente como al disminuir el umbral disminuye la precisión y aumenta la recuperación, mientras que aumentarlo genera el efecto contrario. La intersección de los tres vectores se da justamente en el valor 0,5 del umbral. En la figura 5.5 se grafica lo mismo pero sobre el modelo *Max*, donde se puede observar el mismo comportamiento, con la diferencia que la intersección de vectores se realiza en el umbral 0,725.

Para integrar el resultado de varios modelos se pueden utilizar las funciones *Avg* y *Max*. La decisión de una u otra depende de priorizar precisión o cobertura de transacciones positivas. Una vez elegido el modo de integración de modelos a utilizar, se puede calibrar el umbral ejecutando el *Scorer* sobre un conjunto inicial de datos, graficar el resultado y seleccionar el umbral que genere una certidumbre y cobertura más cercana a la necesidad del analista, tal como fue hecho en esta sección.

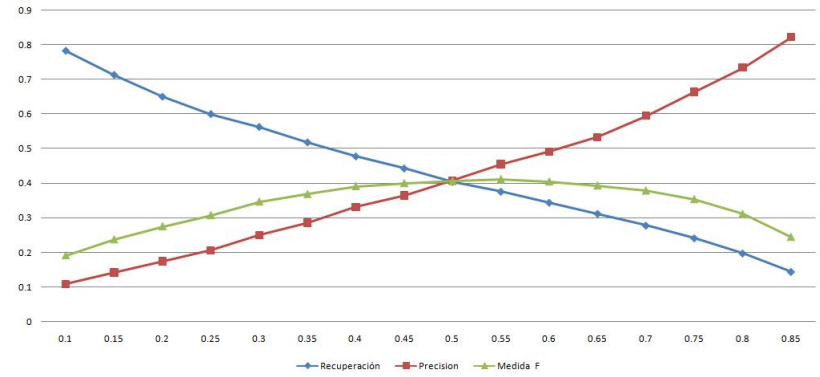

Figura 5.4: Comportamiento del modelo Avg al variar el umbral en el eje x

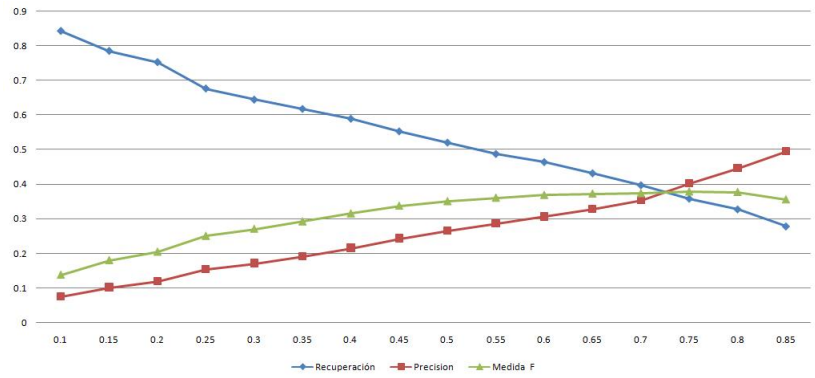

Figura 5.5: Comportamiento del modelo Max al variar el umbral en el eje x

Capítulo 6

Aplicación desarrollada

Para facilitar los procesos requeridos para esta investigación desarrollamos una aplicación que permite, a partir de un conjunto de transacciones clasificadas, entrenar modelos de árboles de decisión, redes neuronales y *random forest*. Esto incluye tomar transacciones de la base de datos, realizar las transformaciones necesarias y aplicar algoritmos de selección de atributos de forma de disminuir el volumen y optimizar los tiempos de entrenamiento. La aplicación desarrollada consta de tres partes: el *Batch Manager*, con el que se actualizan los modelos, el *Scorer* que evalúa transacciones sin clasificar generando un puntaje de riesgo de fraude y la interfaz gráfica que permite visualizar modelos y resultados. Todos los datos analizados en capítulos anteriores fueron generados a partir de esta aplicación.

6.1. Batch Manager

Entrenar un modelo o aplicar un algoritmo de selección de atributos sobre cantidades significativas de datos son tareas que insumen tiempo de ejecución. Normalmente se ejecutan cada cierto tiempo, donde se actualizan modelos en función a nuevos datos de entrenamiento para ser luego incorporados a la aplicación encargada de evaluar nuevas transacciones, *Scorer*. Es por esto que el *Batch Manager* es una aplicación de línea de comandos, de forma que sus tareas puedan ser agendadas para ser ejecutadas con regularidad. Basta con que la tarea agendada actualice los archivos de modelos anteriores en el *Scorer* para que éste los comience a utilizar.

Las funcionalidades presentes en éste módulo van desde convertir instancias en un archivo ARFF hasta la evaluación de instancias, pasando por la aplicación de filtros, selección de atributos, generación y visualización de modelos. Las funcionalidades se presentan en el cuadro 6.1.

Su funcionalidad es expuesta por la biblioteca principal del sistema, para que pueda ser integrado directamente en el código fuente de otras aplicaciones que lo requieran. A modo de ejemplo, la figura 6.1 muestra una secuencia de ejecución completa desde la creación de los archivos ARFF, a la evaluación de un conjunto sin clasificar y generación de reglas. En el apéndice E se detallan los

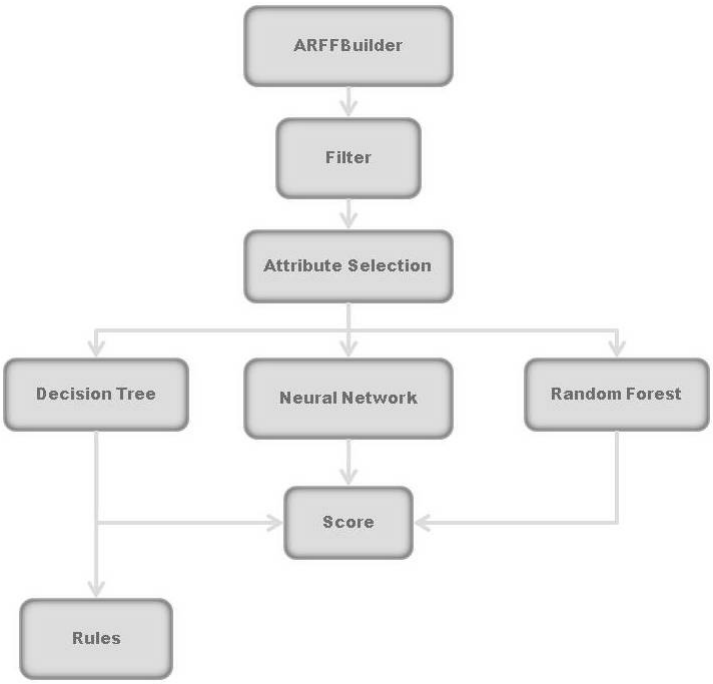

Figura 6.1: Secuencia de ejecución típica

comandos utilizados con sus respectivos parámetros.

Ver apéndice E para más información sobre cómo ejecutar el *Batch Manager*.

6.2. Scorer

Si no se considera el comando de visualización de reglas y gráficas de ROC, la sección anterior explica básicamente todos los pasos necesarios para llegar a construir el *scorer*. El principal objetivo de esta funcionalidad es permitir evaluar instancias sin clasificar y retornar el puntaje (en inglés *score*) que determinará si cada instancia evaluada es o no positiva (fraude). Cuanto más cercano a 1 es el puntaje, mayor es la certeza que la transacción es de la clase fraude.

Para facilitar tanto su uso como integración con aplicativos de terceros, el módulo cuenta con varios modos de ejecución. Permite ser ejecutado por línea de comando desde el *Batch Manager* sobre un conjunto de transacciones, pudiendo ser invocado de forma *"batch"* mediante tareas agendadas o de forma inmediata por transacción a evaluar desde un sistema externo. Mediante su interfaz gráfica puede ser accedido de forma intuitiva por un analista de riesgo, sin preocuparse por aspectos de configuración. Por último, es expuesto mediante la biblioteca principal, pudiendo ser incorporado al código fuente de otra aplicación que desee integrar su funcionalidad.

Conversor a ARFF: Permite acceder a una base de datos y convertir instancias en un archivo ARFF para prueba o evaluación. ARFF es un formato de archivo separado por comas donde la primera línea indica el nombre del dato y su tipo, y las demás líneas corresponden a las filas con sus respectivos valores. El conversor no simplemente genera el archivo, sino que además procesa los datos para ser utilizados en los modelos. Por ejemplo, si un dato es multivaluado, se encarga de enumerar en la primera línea sus valores posibles. La razón de utilizar un archivo es que los algoritmos de entrenamiento realizan un mejor manejo de memoria con éstos que accediendo directamente a la base de datos.

Filtros: Muchas veces, atributos de clase pueden estar representados por un campo numérico en la base de datos aunque en la realidad sean campos nominales. Al mismo tiempo, muchos algoritmos no funcionan con atributos numéricos por lo que deben ser discretizados previamente. Mediante los filtros, una vez obtenidos los archivos del paso anterior, se pueden realizar las modificaciones necesarias que requieran los algoritmos a utilizar.

Selección de atributos: A partir de un archivo ARFF con transacciones retorna los mejores atributos en orden de importancia a partir de la aplicación de algoritmos de *feature ranking* y *subset selection* detallados en el capítulo 4.4.

Generación de modelos: Dado un conjunto de entrenamiento y otro de prueba, entrena modelos de clasificadores de árboles de decisión, redes neurales y *random forests*. Permite configurar los distintos modelos y obtener estadísticas sobre el comportamiento del modelo generado en el conjunto de prueba. También guarda el modelo entrenado en un archivo para poder ser utilizado en etapas posteriores.

Evaluación: Dado un conjunto de instancias sin clasificar y un conjunto de modelos, evalúa cada instancia generando su puntaje de riesgo en todos los modelos utilizados. Esta información puede utilizarse con diferentes umbrales y funciones de agregación para decidir si la instancia es clasificada como fraudulenta o no.

Visualización de modelos: Para el caso de redes neuronales y random forest, dado un conjunto de prueba y un archivo que contenga un modelo entrenado previamente, puede construir la curva de ROC y mostrarla en pantalla. Para el caso de árboles de decisión, a partir del modelo despliega en pantalla el árbol obtenido.

Obtención de reglas: Para un modelo de árbol de decisión, permite listar las mejores reglas. Permite elegir la cantidad de reglas a listar, un criterio y el tipo de reglas a considerar.

Cuadro 6.1: Funcionalidades del módulo *Batch Manager*

Como ya se detalló previamente, los modelos generados permiten ser guardados luego de ser configurados y entrenados. De esta forma, el *scorer* es capaz de levantar un modelo de forma inmediata, sin necesidad de repetir el proceso de entrenamiento. Incluso puede utilizar varios modelos al mismo tiempo y listar el puntaje de fraude según cada modelo utilizado.

La salida de la ejecución es un archivo de extensión **CSV** (en inglés *Coma Separated Value*), un estándar para representar valores en forma de tabla. Dicho formato puede ser abierto por programas como Microsoft Excel u Open Office, permitiendo analizar los datos generados en la herramienta de preferencia. Como resultado, el archivo lista para cada instancia evaluada los valores que la componen y el puntaje de riesgo según cada modelo utilizado durante la evaluación.

Como se describió en la sección 5.4, el analista puede determinar distintos umbrales y utilizar funciones de agregación para obtener mayor precisión o cobertura según sus necesidades.

6.3. Interfaz gráfica

La intención de la interfaz gráfica es que pueda ser utilizada por un analista de riesgo. Por lo tanto es importante que no se preocupe de la creación de los modelos, para lo cual debería tener ciertos conocimientos de experto en la materia, sino que hayan sido originados previamente a través del *Batch Manager* por una persona especializada.

En el menú principal de la aplicación se cuentan con tres opciones *Archivo, Evaluación y Visualización*. Dentro del menú *Archivo* se cuenta con dos opciones: *Configuración y Salir*. Al seleccionar *Configuración*, visualizamos la figura

Figura 6.2: Menú *Archivo*

6.3. El recuadro *Modelos* sirve para configurar las rutas de los modelos a utilizar en cada uno de los algoritmos. El recuadro *Conjunto para construir la curva de ROC*, como su nombre lo indica, es para ingresar el conjunto a ser utilizado en la visualización de las curvas de ROC (menú *Visualización*). Luego de completar las rutas, escribiéndolas o buscando el archivo seleccionando *Cargar*, debe seleccionarse *Aceptar* para que los cambios tengan efecto.

El segundo menú, *Evaluación*, cuenta con solo una opción a través de la cual el usuario ejecutará el *Scorer*. En este caso tomará en cuenta los tres modelos

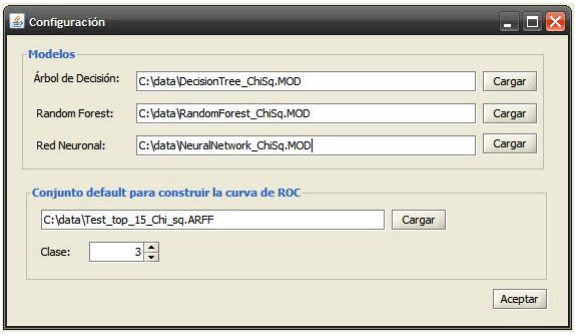

Figura 6.3: Configuración

ingresados. En la figura 6.5, se puede ver que es necesario ingresar el conjunto a evaluar, que debe estar en formato ARFF así como su índice de clase (comienza en 0). También se ingresa una ruta al archivo resultado, sin extensión, que además de contener las evaluaciones contiene los datos de la transacción. El formato de este archivo será CSV. Al seleccionar el comando *Evaluar*, se obtienen los puntajes desplegados en pantalla y al mismo tiempo los resultados CSV anteriores. Al seleccionar Abrir en Excel, el archivo generado se abre en MS Excel para poder ser analizado por el analista en más detalle.

Figura 6.4: Menú *Evaluación*

Pasamos ahora al último menú, el de *Visualización*. Este menú tiene como objetivo mostrar resultados al usuario. Se tienen tres opciones, *Árbol de Decisión, Reglas y Curva de ROC*. Al seleccionar la primera opción, se obtiene la representación del modelo de árbol de decisión previamente ingresado en *Configuración* como se puede ver en la figura 6.7. Esta ventana puede arrastrarse utilizando el *mouse* para cubrir todo el árbol y recorrer las hojas.

Al seleccionar la opción *Reglas*, vemos la figura 6.8. Es necesario ingresar la ruta al resultado del la construcción de un modelo de árbol de decisión y una ruta resultado, sin extensión. Es posible seleccionar el orden en el cual se quieren elegir las reglas según la medida-f, la recuperación o la precisión. También debe especificarse la cantidad de reglas a retornar y por último si las reglas a considerar son solo aquellas que clasifican a una transacción como fraudulentas, no fraudulentas o si se consideran todas las transacciones al ordenar y seleccionar.

La última opción permite ver las gráficas de ROC para un modelo *Random Forest* o de *Red Neuronal*, ingresados previamente en *Configuración*. Para la

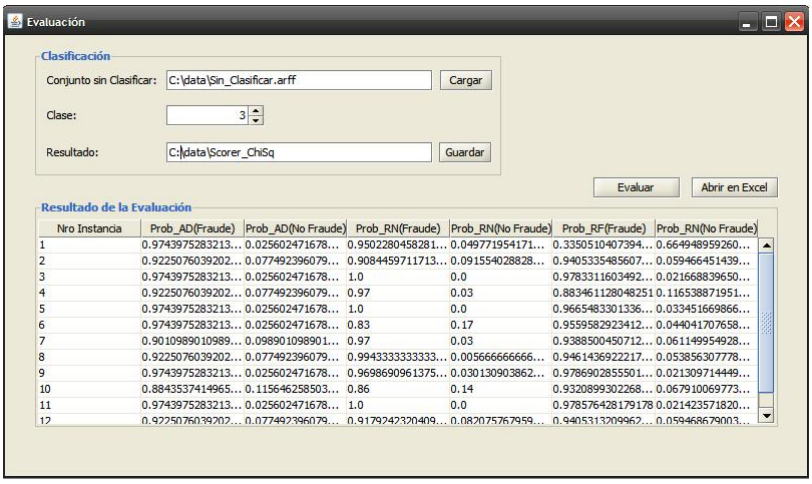

Figura 6.5: Scorer

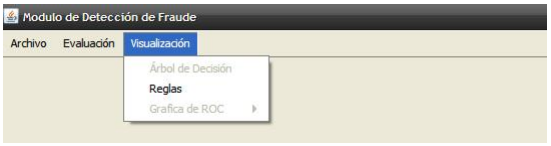

Figura 6.6: Menú *Visualización*

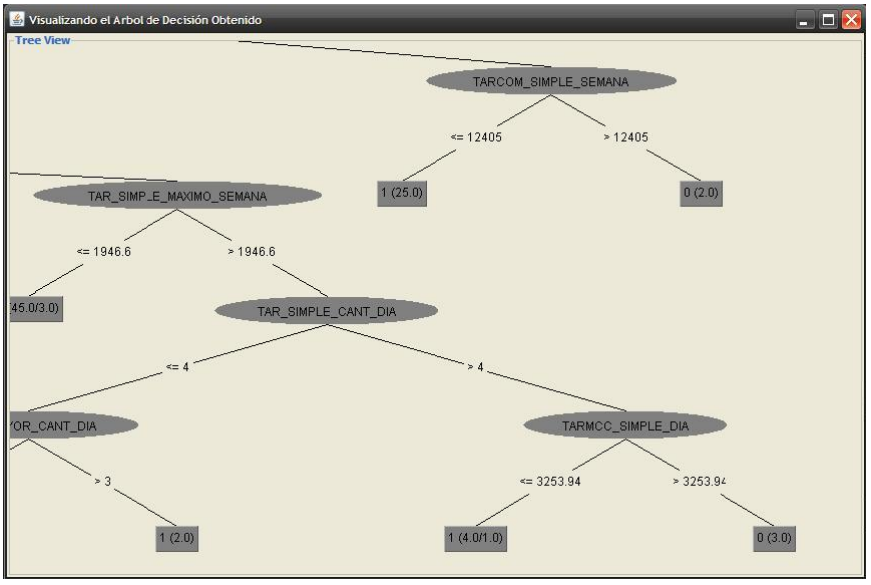

Figura 6.7: Visualización del Árbol de Decisión

construcción de la curva se utiliza el conjunto determinado por ruta ingresada previamente, también en *Configuración*.

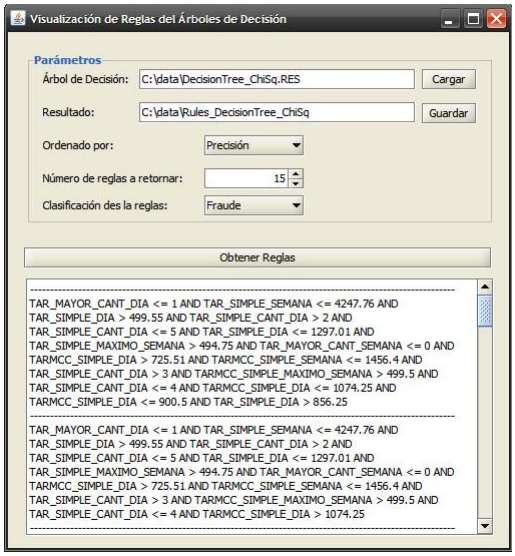

Figura 6.8: Reglas

Capítulo 7

Conclusiones

El objetivo de este proyecto es lograr un sistema capaz de identificar patrones de fraude en un conjunto de transacciones clasificadas como fraudulentas o legítimas de manera de poder utilizar estos patrones para modificar la configuración del *Detection Engine*.

El proyecto se divide en tres tareas:

- Definir un conjunto de atributos calculados con la finalidad de detectar cuáles de ellos resultan relevantes para la determinación de fraude.
- Evaluar y obtener métodos de clasificación para la generación de reglas.
- Evaluar y obtener funciones de evaluación para el cálculo de puntaje de riesgo (*scorer*).

Para realizar estas tareas debemos clasificar las autorizaciones del conjunto de datos de prueba proporcionado por el cliente del proyecto. Luego necesitamos reducir el conjunto de instancias con las cuales trabajar. El resultado de las siguientes tareas está fuertemente atado a la calidad de los datos conseguidos y a su proceso de clasificación.

A continuación presentamos los aportes y conclusiones de cada una de las tareas.

Atributos Calculados

A partir de los datos disponibles en las transacciones definimos nuevos atributos que incluyen información histórica de la tarjeta y del comercio involucrado que pueden llegar a ser de interés para la detección de fraude.

Tomamos las funciones de agregación *count, sum, min, max, avg* (sobre el campo *monto*) agrupando distintas autorizaciones, con igual *día, semana y mes* según un grupo de condiciones. De esta manera agregamos a cada instancia 150 atributos calculados. El objetivo es poder determinar cuáles de ellos son relevantes en la clasificación de fraude. Realizamos pruebas utilizando distintos métodos de selección de atributos obteniendo para cada prueba un subconjunto de los atributos originales.

Observamos que de los más de 150 atributos disponibles únicamente `TAR_MAYOR_DIA`, `TAR_MAYOR_CANT_DIA` se encuentran en todos los subconjuntos obtenidos, siendo de especial interés para la detección de fraude. El atributo `TAR_MAYOR_SEMANA` aparece en cuatro de los cinco subconjuntos calculados, seguido de cerca por los atributos `TAR_SIMPLE_SEMANA`, `TAR_SIMPLE_CANT_DIA` y `TAR_MENOR_AVERAGE_MES` presentes en tres de los subconjuntos. Considerando la unión de todos los subconjuntos obtenidos en las pruebas se obtienen 33 atributos siendo posible descartar un gran número de atributos que fueron calculados originalmente.

Por limitaciones computacionales no pudimos utilizar en los *scorers* el conjunto total de atributos. En los árboles de decisión observamos que los algoritmos se comportan de mejor manera con los subconjuntos calculados que con la totalidad de atributos disponibles.

Generación de reglas

Para esta tarea estudiamos técnicas de aprendizaje automático para deducir reglas a partir de un conjunto de instancias clasificadas.

Como entrada utilizamos los conjuntos de entrenamiento y evaluación tomando distintos subconjuntos de atributos. En primer lugar utilizamos el conjunto de atributos completo y luego los subconjuntos obtenidos en la etapa de selección de atributos. Para cada conjunto ejecutamos algoritmos de aprendizaje de árboles de decisión con distintos parámetros. Los árboles de decisión obtenidos pueden transformarse casi inmediatamente a un conjunto de reglas interpretables por un analista.

Si consideramos la clase positiva se obtuvieron resultados variados entre sí con una precisión entre 17 % y 48 % y una recuperación entre 32 % y 47 %. Esto puede deberse a la calidad del conjunto de entrenamiento, al proceso de marcado de fraude, a la distribución despareja del conjunto de entrenamiento, a que el conjunto de atributos calculados no sean los más relevantes para la detección de fraude o a la complejidad del problema planteado. La prueba en la que obtuvimos la mayor medida-f fue la que utilizó el árbol J48 con el conjunto de entrada *Gain Ratio*.

Puntaje de riesgo

Para esta tarea elegimos inicialmente dos tipos de algoritmos: un paradigma de aprendizaje automático inspirado en el funcionamiento del sistema nervioso animal, denominado red neural y *random forest*, el cual se basa en un conjunto de árboles de decisión. En la etapa de construcción del scorer optamos por incluir el modelo creado para árboles de decisión, cuyo primer objetivo es descubrir reglas, para que permita aportar a la predicción.

Al igual que en los árboles, las redes neuronales y los bosques aleatorios obtenidos presentaron un gran nivel de precisión total y se comportan de mejor respecto a la clase *No Fraude* que a la clase *Fraude*, con una precisión superior

al 97 %.

De igual forma que en los árboles si consideramos la clase positiva los resultados logrados son variados. Obtuvimos con las pruebas de redes neuronales una precisión entre 33 % y 55 % y una recuperación entre 25 % y 43 %. Obtuvimos la mayor medida-f con el conjunto de entrenamiento generado por el algoritmo de selección Gain Ratio, en una red de dos capas ocultas, 3000 *epochs* y tasa de aprendizaje de 0,3. En nuestras pruebas con bosques obtuvimos una precisión entre 27 % y 31 % y una recuperación entre 32 % y 45 %. Obtuvimos la mayor medida-f y área bajo la curva de ROC con el subconjunto *Chi-Sqr* y 100 árboles en el bosque aleatorio.

Es importante observar que alterando el valor de umbral utilizado al interpretar los datos del scorer se puede variar la relación entre cobertura de transacciones positivas y precisión tal como lo hemos estudiado en la sección 5.4. De esta forma se pueden obtener valores de equilibrio que se adecuen a la necesidad del analista, priorizando cobertura sobre precisión y viceversa. Como fue visto, pueden utilizarse resultados de varios modelos, mediante funciones que integren el puntaje de riesgo, como por ejemplo la función Max, la cual obtiene una cobertura de casi el 74 % con una precisión de casi el 13 %, logrando una cobertura mayor que cualquiera de los modelos que la integran por sí solos.

Como punto de comparación podemos observar los resultados obtenidos por el antecedente de este proyecto [11] sobre un conjunto de transacciones reales donde lograron como mejor precisión un 16 % y mejor recuperación 50 %.

Para el aprendizaje de reglas utilizaron árboles de decisión, en particular los programas C4.5 y QUEST. Para obtener conjuntos de entrenamiento realizaron *sampling*, pero a diferencia de nuestro proyecto en donde realizamos un muestreo al azar, realizaron un muestreo a nivel de cuenta, eligiendo todas las transacciones de cuentas que cumplen con cierta condición. Para el entrenamiento utilizaron un conjunto con las cuentas con fraude y conjunto con las cuentas con fraude y transacciones en el extranjero. Notar que el segundo conjunto es un subconjunto del primero. Contaban con transacciones de tres meses de actividad de una misma institución financiera, entrenaron con dos meses de actividad y evaluaron con el mes restante.

Para el aprendizaje de *score* utilizaron un enfoque *Naive Bayesian*. A diferencia de nuestro proyecto, al entrenar el *Naive Bayesian* se utilizaron las mejores reglas aprendidas en la fase anterior, eligiendo aquellas reglas cuya precisión superó el 3 %. De esta forma entrenaron sobre subconjuntos formados por las transacciones clasificadas como fraude por cada regla con precisión mayor a 3 %.

Los resultados que obtuvieron utilizando QUEST para el aprendizaje de reglas fueron inferiores a los que obtuvieron por C4.5 por lo que utilizaremos éstos últimos para comparar. Obtuvieron una precisión del 10 % y una cobertura del 47 % utilizando el primer conjunto y una precisión del 16 % y una cobertura del 29 % utilizando el segundo conjunto. Utilizando la unión de los dos modelos obtuvieron una precisión del 10 % y una cobertura del 50 %.

En el cuadro 7.1 se muestran los mejores resultados obtenidos con los tres paradigmas de aprendizaje automático utilizados en este proyecto y el mejor resultado obtenido por el antecedente [11].

Las mejoras obtenidas en este proyecto pueden deberse a varios factores dentro

Algoritmo	Prec.	Rec.	Med.-F
Árboles	0,48	0,37	0,41
Redes	0,50	0,35	0,41
Bosque	0,31	0,44	0,36
Proy. Anterior	0,16	0,29	0,21

Cuadro 7.1: Estadísticas para mejor medida-f obtenida para la clase positiva

de los cuáles identificamos: la existencia de una etapa de selección de atributos, de forma de poder entrenar los modelos con menos y *mejores* atributos; el método de *sampling* elegido, en donde buscamos balancear la distribución de las clases sin elegir transacciones con características especiales, sesgando lo menos posible los conjuntos de entrenamiento; el utilizar los conjuntos enteros para el entrenamiento y no utilizar únicamente las transacciones identificadas como fraudulentas por los árboles de decisión para la etapa de *score*, intentando nuevamente, no sesgar los conjuntos de entrenamiento y utilizar la mayor cantidad de ejemplos posibles. Por último nuestro proyecto contaba con la información de un año de actividad, a diferencia del antecedente, que contaba únicamente con tres meses.

7.1. Trabajo futuro

En cada una de las tares realizadas durante el desarrollo del proyecto encontramos puntos a mejorar o investigar.

Se podría buscar una mejora en los resultados cambiando o agregando atributos a las instancias. Se deberían definir nuevos atributos calculados a partir del histórico o de la información disponible en cada transacción. Estos nuevos atributos podrían participar de la selección de atributos obteniendo nuevos subconjuntos para ser utilizados en las posteriores etapas.

Otro punto a investigar es cambiar el método de *sampling* que utilizamos. Si bien en nuestra consideración la decisión de utilizar la totalidad de instancias positivas y 200 000 no fraudulentas tomadas al azar fue correcta, no realizamos pruebas para validar esta decisión. Se podría utilizar un muestreo a nivel de cuenta en lugar de transacción o utilizar alguna condición para particionar el conjunto inicial de datos. Se podrían considerar por ejemplo, únicamente cuentas con fraude, transacciones en el extranjero o transacciones por un monto mayor a un umbral.

Algunos trabajos [41, 44] insinúan que con una distribución de 50/50 en los datos de entrenamiento se obtienen mejores resultados que con distribuciones

más desparejas aunque estas últimas sean más aproximadas a la realidad.

Otra tarea a realizar es probar nuevos algoritmos de aprendizaje. Existen varios algoritmos que tienen las características necesarias para poder ser aplicados a la problemática de fraude y que podrían dar mejores resultados. Inclusive los algoritmos que utilizamos en nuestro proyecto pueden parametrizarse de distinta manera buscando mejorar los resultados obtenidos. Por ejemplo se pueden realizar pruebas con un mayor número de árboles en el bosque o con distinta cantidad de capas y/o nodos en redes. Los nuevos modelos podrían ser fácilmente incorporados en el *scorer*, permitiendo mejorar la precisión y cobertura.

Por último, restaría realizar un estudio para determinar cada cuánto sería necesario la actualización de los modelos para mantener el sistema actualizado y lo más efectivo posible, dado que la realidad particular del fraude es muy dinámica. Un artículo presentado por *Fair Isaac* en su página web [14], concluye que para el caso de riesgo crediticio sería necesario al menos actualizar los modelos cada cuatrimestre, y en caso de ser posible cada mes.

Glosario

Acuerdo comercial	Contrato que suscriben el comercio y el adquirente para permitir al establecimiento que acepte tarjetas de determinada marca como pago por los bienes y servicios que vende, y el cual requiere al comercio acatar determinadas reglas que rigen la aceptación y el procesamiento de las transacciones con productos de la marca.
Adquirente	Institución que mantiene las relaciones comerciales con los comercios, recibe todas sus transacciones, cobrándoles cargos por el servicio. El adquirente se encarga de pagarles a los comercios a cambio de las transacciones que los tarjetahabientes hayan efectuado.
Algoritmo de aprendizaje	Algoritmo a través del cual, un programa puede mejorar alguna medida de performance a partir de un conjunto de ejemplos o de la experiencia.
Aprendizaje Automático (AA)	Rama de la inteligencia artificial que estudia algoritmos de aprendizaje y generación automática de conocimiento.
Aprendizaje Supervisado	Es una técnica de aprendizaje automático usualmente utilizada para el ajuste de una función a partir de datos de entrenamiento. Los datos consisten en pares de entrada y su salida deseada. La salida de la función puede predecir valores continuos (regresión), o puede predecir una etiqueta de clase de un objeto de entrada (clasificación).
Autorización	Proceso mediante el cual el adquirente y el emisor aprueban una transacción efectuada con una tarjeta bancaria. Con el uso de los terminales de punto de venta *(POS)* y otros dispositivos electrónicos de procesamiento de transacciones, la autorización es automática. También se puede obtener autorización por vía telefónica llamando a un centro de autorización.

Banda Magnética	Una banda de cinta magnética que se coloca en el reverso de todas las tarjetas bancarias para que el terminal de punto de venta *(POS)* la lea cuando se pasa la tarjeta. La banda está codificada con la información que identifica la cuenta. En una tarjeta válida, el número de cuenta codificado en la banda magnética coincide exactamente con el número grabado en alto relieve en el frente de la tarjeta.
Comercio	Establecimiento comercial afiliado por la entidad adquirente que le permitirá aceptar tarjetas de crédito/débito/prepago como medio de pago de sus clientes.
Cross-validation	Es la práctica estadística de particionar datos de muestra en subconjuntos de forma que su análisis es realizado en un subconjunto dado, mientras los restantes son utilizados posteriormente para confirmar y validar el análisis inicial.
CVV	Valor de verificación de la tarjeta, CVV por sus siglas en inglés *Card Verification Value*. Es un código contenido en las tarjetas de crédito o débito para incrementar la protección contra el fraude. Existen varios tipos de CVV, el CVV1 está codificado en la banda magnética de la tarjeta magnética y se usa para transacciones de tarjeta presente, el CVV2 es utilizado en escenarios de tarjeta no presente y se encuentra impreso en el reverso de la tarjeta.
Disponible	Monto que es posible gastar con determinado medio de pago. El disponible se calcula a partir del límite de crédito, el saldo del último cierre, el total de compras del período en curso, el saldo de reservas, los pagos del período en curso, y saldos a vencer.
Ejemplo de entrenamiento	Usualmente consiste en un par (entrada, salida) donde se indica la salida deseada para una entrada dada, para un concepto a aprender.
Embedded Methods	Métodos Embebidos.

Emisor	Banco, institución financiera u otro, encargado de la emisión del medio de pago y administración de éste, otorgándole al cliente financiación de sus compras. El negocio del emisor se basa en el cobro de cargos por asumir este riesgo y por brindar el servicio. Encargado de estudiar la capacidad de pago del tarjetahabiente, asignarle un límite de crédito, autorizar las transacciones, controlar el disponible de compra, emitir el estado de cuenta para sus clientes y realizarles los pagos correspondientes a los adquirentes, entre otras funciones.
EMV	Estándar de interoperabilidad de tarjetas con chip, para la autentificación de pagos mediante tarjetas de crédito y débito. El nombre EMV es un acrónimo de *Europay MasterCard VISA*, las tres compañías que inicialmente colaboraron en el desarrollo del estándar.
Entropía	El concepto básico de entropía en teoría de la información tiene mucho que ver con la incertidumbre que existe en cualquier experimento o señal aleatoria. La entropía se puede ver como una medida de la cantidad de información que tiene un determinado medio o mensaje.
Epoch	Un paso en el proceso de entrenamiento de una red neuronal artificial.
Feature Ranking	Ordenamiento jerárquico de atributos.
Filter	Filtro.
Marca	Institución que representa y regula a todos los miembros para la aceptación de una tarjeta específica. Puede ser una marca privada o internacional.
MCC (Merchant Category Code)	Tipo de negocio o rubro del producto o servicio.
MOTO (mail or telephone order)	Transacción en donde el comercio no puede ver la tarjeta, por ejemplo, por teléfono, por mail o por Internet. Es conocido también como escenario de tarjeta no presente.
POS	Terminal de punto de venta, POS por sus siglas en inglés *Point Of Sale*, es un sistema informático que gestiona el proceso de venta mediante una interfaz accesible para los vendedores. El mismo sistema permite la creación e impresión del ticket de venta mediante las referencias de productos, realiza cambios en el stock en la base de datos y otras labores del negocio.

Procesadora	Una organización que realiza funciones de autorización y procesamiento de transacciones, mantenimiento de registros y contabilidad de cuentas y otras funciones administrativas y comerciales de rutina a favor de un emisor o adquirente.
Subset Selection	Selección de subconjunto.
Tarjeta No Presente	El término *Tarjeta Ausente* o *Tarjeta No Presente* se utiliza en referencia a los comercios y ambientes que procesan órdenes por correo/teléfono, así como a los comercios que operan en línea a través de la Red Internet.
Tarjeta Presente	Un comercio, mercado o entorno de procesamiento de transacciones de venta dónde la transacción se puede completar solamente si tanto la tarjeta como el tarjetahabiente están presentes en el punto de transacción. Las transacciones en que la tarjeta está presente incluyen las transacciones de compra cara a cara en comercios minoristas y los desembolsos de efectivo.
Tarjetahabiente	Persona que tiene relación comercial con la institución o banco emisor, poseedor de una tarjeta para el uso en comercios afiliados.
Ticket / Voucher	Nota de consumo o de venta firmada por el cliente autorizando el cargo a su tarjeta de crédito. Formulario que presenta el comercio por cada transacción manual realizada.
Transacción	El acto que ocurre entre un tarjetahabiente y un comercio o institución financiera y que trae como resultado la venta de bienes o servicios. También se llama transacción a los mensajes que se intercambian entre emisores y adquirentes.
Wrapper	Envoltorio.

Apéndice A

ISO 8583

Número de cuenta primario (*Primary Account Numbero PAN*): Un número que identifica la cuenta del cliente, o sea, un número de tarjeta de hasta 19 dígitos. A partir del PAN se obtiene el número identificador del emisor/producto.

Código de procesamiento (*Processing Code*): Este campo contiene un código que identifica el tipo de transacción y de cuenta afectadas por la transacción. El campo es numérico de largo 6. A continuación se listan los valores posibles para las primeras 4 posiciones:

```
Posiciones 1 a 2: Tipo de Transacción
   00 Compra de Bienes/Servicios - Débito
   02 Ajuste - Débito
   11 Transacción en efectivo
   17 Scrip - Débito
   19 Tasa - Débito
   20 Retorno (de mercancías)-Crédito
   22 Ajuste - Crédito
   28 Activación y carga - Prepago
   29 Fondos de desembolso - Crédito
   30 Consulta de fondos disponibles
   72 Activación - Prepago

Posiciones 3 a 4: Tipo de Cuenta
   00 No aplica / No especificada
   10 Cuenta de ahorros
   20 Cheques
   40 Cuenta Universal (representada por ID de cliente)
```

Monto de la transacción: Contiene el monto de la transacción en la moneda especificada en el campo código de moneda. Es un campo numérico de largo 12 que no incluye punto o coma decimal y los decimales se asumen basados en el campo código de moneda.

Monto de la venta: Contiene el monto de la transacción convertido a la mo-

neda de compra. Es un campo numérico de largo 12 que no incluye punto o coma decimal; los decimales se asumen basados en el campo código de moneda de compra. Se utiliza para mantener el monto en múltiples monedas.

Monto en moneda de la facturación del tarjetahabiente: Contiene el monto de la transacción convertido a la moneda de facturación del tarjetahabiente. Es un campo numérico de largo 12 que no incluye punto o coma decimal, los decimales se asumen basados en el campo tipo de conversión, moneda de facturación del tarjetahabiente.

Fecha y Hora de la transacción: Contiene la fecha y hora en GMT de la transacción con formato *MMDDhhmmss*.

Tipo de conversión, venta: Contiene el tipo de conversión utilizado para convertir el monto de la transacción al monto de venta.

Tipo de conversión, moneda de la facturación del tarjetahabiente: Contiene el tipo de conversión utilizado para convertir el monto de la transacción al monto en moneda de la facturación del tarjetahabiente.

Hora local de la transacción: Contiene la hora de la transacción expresado en hora local de la ubicación del tarjetahabiente con formato *hhmmss*.

Fecha local de la transacción: Contiene la fecha de la transacción expresada en hora local de la ubicación del tarjetahabiente con formato *MMDD*.

Fecha de vencimiento: Contiene el año y mes luego del cual se vence la tarjeta en formato *YYMM*.

Fecha de venta: Contiene la fecha en la cual el mensaje ingresó en la red con formato *MMDD*.

Fecha de conversión: Contiene la fecha efectiva del tipo de conversión utilizada para convertir el monto de la transacción al monto de venta, con formato *MMDD*.

MCC (*Merchant Category Code*): Contiene un código describiendo el tipo de negocio o rubro del producto o servicio. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [50].

```
3001 American Airlines
3504 Hilton Hotels
5411 Supermercado
5814 Restaurantes de comida rápida.
6010/6011 ATM
```

Código de país del adquirente: Contiene un código que identifica al país del adquirente del comercio o ATM. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [32].

```
032 Argentina
076 Brazil
858 Uruguay
```

Código de país del emisor: Contiene un código que identifica al país del emisor de la tarjeta, usando la misma codificación que para el país adquirente.

Modo de captura de datos del POS (*POS entry mode*): Contiene un código que identifica el método utilizado para capturar el número y fecha de vencimiento de la tarjeta cuando se usa una terminal electrónica y la capacidad de captura de la terminal. A continuación se listan los valores posibles para los dos subcampos.

```
Modo de captura
   00 Desconocido, o no se usó una terminal electrónica
   01 Digitado manualmente
   02 Lectura de la banda magnética de la tarjeta, con la salvedad de que
      es posible que el CVV (código de verificación grabado en la banda)
      no se haya podido leer en forma confiable
   03 Lectura de código de barras
   04 Lectura mediante reconocimiento óptico de caracteres (OCR)
   05 Lectura de circuito integrado, la captura del CVV (en este caso,
      grabado en el chip) es confiable
   90 Lectura de la banda magnética de la tarjeta, el CVV se leyó
      correctamente
   95 Lectura de circuito integrado, el CVV no es confiable


Capacidad de la terminal
   0 Desconocido
   1 No se utilizó POS
   2 Lectura de banda magnética
   3 Lectura de código de barras
   4 Lectura mediante reconocimiento óptico de caracteres (OCR)
   5 Lectura de circuito integrado
   9 Digitación manual, sin capacidad de lectura electrónica
   N No se proporcionó
```

Adicional al número de tarjeta: Campo numérico de largo 3 usado para diferenciar entre tarjetas con el mismo número o tarjetas adicionales a la principal.

Código de condición del POS (*POS condition code*): Contiene un código que identifica las condiciones en las que se realizó la transacción. A continuación se listan los valores posibles.

```
00 Transacción normal
01 Cliente no presente
02 Terminal automática operada por el cliente (p.e. ATM)
03 Comercio o tarjeta sospechosos
```

```
05 Cliente presente, tarjeta no presente
06 Pedido de preautorización.
08 Compra por teléfono o correo; incluye transacciones recurrentes
   (como cobro automático de facturas de servicios)
10 Se verificó la identidad del cliente
51 Verificación de dirección o cuenta, no se está pidiendo autorización
   para una transacción.
59 Solicitud de comercio electrónico por una red pública, como Internet
   71 Tarjeta presente, pero no se pudo leer la banda; los datos fueron
   digitados manualmente
```

Número identificador del adquirente: Campo numérico de hasta 11 caracteres que identifica la institución financiera que actúa como adquirente.

Código de respuesta: Contiene un código que define la respuesta a un pedido de autorización. Los códigos 00 o 10 indican aprobación y el resto de los códigos indican la razón de rechazo. A continuación se listan a modo de ejemplo algunos valores posibles.

```
01 Referirse al emisor
03 Comercio inválido
15 No existe el emisor
```

Número identificador de la terminal de venta: Contiene un código que identifica la terminal.

Número identificador del comercio: Contiene un código alfanumérico que puede identificar el comercio o una terminal.

Nombre y ubicación del comercio: Campo alfanumérico que indica el nombre, ciudad, estado y país del comercio.

Moneda de la transacción: Contiene el código de la moneda del monto de la transacción.

Moneda de la venta: Contiene el código de la moneda del monto de la venta.

Moneda de la facturación del tarjetahabiente: Contiene el código de la moneda de facturación del tarjetahabiente.

Apéndice B

Conjunto de datos

Se listan a continuación los atributos del conjunto original de datos proporcionados por *PayTrue solutions*, junto a su descripción.

AMOUNT: Monto de la autorización.

TRANDATE: Fecha de la autorización.

MERCHANTCODE: Código del comercio.

MERCHANTNAME: Nombre del comercio.

MERCHANTCITY: Ciudad a la que pertenece el comercio.

MERCHANTSTATE: Estado al que pertenece el comercio.

MERCHANTZIP: Código postal del comercio.

TERMINALID: Identificador de la terminal.

MCC: Tipo de comercio o rubro del servicio.

ACQUIRER: Número identificador del adquirente.

CARDID: Número de tarjeta.

MERCHANTNAME, MERCHANTCITY, MERCHANTSTATE, MERCHANTZIP resultan redundantes con el campo MERCHANTCODE; TERMINALID no se considera debido a que contiene casi tantos valores distintos como instancias en el conjunto.

Apéndice C

Etiquetado del Fraude

El objetivo del etiquetado del fraude es marcar aquellas autorizaciones que hayan resultado en una venta fraudulenta. Esta tarea, si bien trivial en teoría, se vuelve extremadamente costosa y delicada al tratar con tantas instancias y al tener que obtener una única autorización como correspondiente con un fraude detectado.

Para el fraude reportado, contamos con tres campos que indican el *número de tarjeta*, el *monto* y la *fecha* en que se realizó la venta. Un inconveniente que debemos superar es que pueden existir diferencias entre el *monto y/o fecha* en el fraude reportado y la autorización que originó la venta. El motivo de estas diferencias puede darse no sólo a errores humanos (dado que el fraude se reporta manualmente) sino que también ocurren debido a la naturaleza del negocio. Un fácil ejemplo de este último caso, es el caso de un hotel, donde la autorización se realiza el primer día de estadía, durante el *check in*, por un *monto* aproximado del costo de la estadía. El último día, en el *check out*, se procesa la venta con lo cual no solo la *fecha* sino el *monto* puede variar (por compras cargadas a la habitación por ejemplo).

El resultado del resto del proyecto está fuertemente atado a la calidad de los datos conseguidos, principalmente la calidad con las que se etiqueten las autorizaciones. Para lograr un buen etiquetado se optó por desarrollar una pequeña herramienta para la configuración de diferencias entre *fechas* y *montos* en las consultas de SQL. De esta manera, primero se etiquetaron las correspondencias *perfectas* en los tres campos, luego con diferencias de hasta un día y cero diferencia en el monto, y así de manera incremental para que el fraude siempre se corresponda con la autorización con menor diferencia posible. Las autorizaciones marcadas no se vuelven a considerar en la iteración siguiente a la que son marcadas.

Tarjeta	***Fecha***	***Monto***	***...***	***...***	***Fraude***
12345	15-08-2008	456.30	...	...	SI
...	...	...	...	...	?
...	...	...	...	...	?
54321	20-08-2008	1326.0	...	...	SI

Cuadro C.1: Tabla de autorizaciones

Tarjeta	***Fecha***	***Monto***
12345	17-08-2008	456.30
...	...	...
...	...	...
54321	20-08-2008	1325.70

Cuadro C.2: Fraude reportado

Apéndice D

Cálculo de atributos

En la siguiente recuadro mostramos los atributos utilizados para calcular las distintas funciones de agregación, así como los filtros en las particiones. Estas particiones se repitieron tomando en cuenta *día, semana y mes* y las funciones de agregación sobre estas particiones fueron *COUNT* (sobre la cantidad de autorizaciones) y *SUM, MIN, MAX y AVG* (sobre el campo AMOUNT).

Particiones
CARDID2
CARDID2, MERCHANTCODE
CARDID2, MCC
MERCHANTCODE
CARDID2, $MCC = 6011 or MCC = 6010$, $MONTO < 10$
CARDID2, $MCC = 6011 or MCC = 6010$, $MONTO > 500$
CARDID2, $MONTO < 10$
CARDID2, $MONTO > 500$
MERCHANTCODE, $MONTO < 10$
MERCHANTCODE, $MONTO > 500$

Cuadro D.1: Particiones utilizadas para las funciones de agregación

Listamos a continuación los atributos agregados:

Igual CARDID2 y DIA:
`TAR_SIMPLE_CANT_DIA` - Cantidad de transacciones diarias para una misma tarjeta
`TAR_SIMPLE_DIA` - Monto acumulado de transacciones diarias para una misma tarjeta
`TAR_SIMPLE_MINIMO_DIA` - Menor monto de transacciones diarias para una misma tarjeta
`TAR_SIMPLE_MAXIMO_DIA` - Mayor monto de transacciones diarias para una misma tarjeta
`TAR_SIMPLE_AVERAGE_DIA` - Promedio del monto de las transacciones diarias para una misma tarjeta

Igual CARDID2 y SEMANA:
TAR_SIMPLE_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta
TAR_SIMPLE_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta
TAR_SIMPLE_MINIMO_SEMANA - Menor monto de transacciones semanales para una misma tarjeta
TAR_SIMPLE_MAXIMO_SEMANA - Mayor monto de transacciones semanales para una misma tarjeta
TAR_SIMPLE_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para una misma tarjeta

Igual CARDID2 y MES:
TAR_SIMPLE_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta
TAR_SIMPLE_MES - Monto acumulado de transacciones mensuales para una misma tarjeta
TAR_SIMPLE_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta
TAR_SIMPLE_MAXIMO_MES - Mayor monto de transacciones mensuales para una misma tarjeta
TAR_SIMPLE_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta

Igual CARID2, MERCHANTCODE y DIA:
TARCOM_SIMPLE_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_DIA - Monto acumulado de transacciones diarias para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MAXIMO_DIA - Mayor monto de transacciones diarias para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta y mismo comercio

Igual CARID2, MERCHANTCODE y SEMANA:
TARCOM_SIMPLE_CANT_SEMANA - Cantidad de transacciones semanal para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_SEMANA - Monto acumulado de transacciones semanal para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MINIMO_SEMANA - Menor monto de transacciones semanal para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MAXIMO_SEMANA - Mayor monto de transacciones semanal para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_AVERAGE_SEMANA - Promedio del monto de las transacciones semanal para una misma tarjeta y mismo comercio

Igual CARID2, MERCHANTCODE y MES:
TARCOM_SIMPLE_CANT_MES - Cantidad de transacciones mensuales para una mis-

ma tarjeta y mismo comercio
TARCOM_SIMPLE_MES - Monto acumulado de transacciones mensuales para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_MAXIMO_MES - Mayor monto de transacciones mensuales para una misma tarjeta y mismo comercio
TARCOM_SIMPLE_AVERAGE_MES- Promedio del monto de las transacciones mensuales para una misma tarjeta y mismo comercio

Igual CARID2, MCC y DIA:
TARMCC_SIMPLE_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_DIA - Monto acumulado de transacciones diarias para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MAXIMO_DIA - Mayor monto de transacciones diarias para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta y mismo tipo de negocio o rubro del servicio

Igual CARID2, MCC y SEMANA:
TARMCC_SIMPLE_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MINIMO_SEMANA - Menor monto de transacciones semanales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MAXIMO_SEMANA - Mayor monto de transacciones semanales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_AVERAGE_SEMANA - Promedio del monto las transacciones semanales para una misma tarjeta y mismo tipo de negocio o rubro del servicio

Igual CARID2, MCC y MES:
TARMCC_SIMPLE_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MES - Monto acumulado de transacciones mensuales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_MAXIMO_MES - Mayor monto de transacciones mensuales para una misma tarjeta y mismo tipo de negocio o rubro del servicio
TARMCC_SIMPLE_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta y mismo tipo de negocio o rubro del servicio

Igual MERCHANTCODE y DIA:
COM_SIMPLE_CANT_DIA - Cantidad de transacciones diarias para un mismo comercio
COM_SIMPLE_DIA - Monto acumulado de transacciones diarias para un mismo

comercio
COM_SIMPLE_MINIMO_DIA - Menor monto de transacciones diarias para un mismo comercio
COM_SIMPLE_MAXIMO_DIA - Mayor monto de transacciones diarias para un mismo comercio
COM_SIMPLE_AVERAGE_DIA - Promedio del monto de las transacciones diarias para un mismo comercio

Igual MERCHANTCODE y SEMANA:
COM_SIMPLE_CANT_SEMANA - Cantidad de transacciones semanales para un mismo comercio
COM_SIMPLE_SEMANA - Monto acumulado de transacciones semanales para un mismo comercio
COM_SIMPLE_MINIMO_SEMANA - Menor monto de transacciones semanales para un mismo comercio
COM_SIMPLE_MAXIMO_SEMANA - Mayor monto de transacciones semanales para un mismo comercio
COM_SIMPLE_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para un mismo comercio

Igual MERCHANTCODE y MES:
COM_SIMPLE_CANT_MES - Cantidad de transacciones mensuales para un mismo comercio
COM_SIMPLE_MES - Monto acumulado de transacciones mensuales para un mismo comercio
COM_SIMPLE_MINIMO_MES - Menor monto de transacciones mensuales para un mismo comercio
COM_SIMPLE_MAXIMO_MES - Mayor monto de transacciones mensuales para un mismo comercio
COM_SIMPLE_AVERAGE_MES - Promedio del monto de las transacciones mensuales para un mismo comercio

Igual CARDID2, DIA, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT < 10$:
TAR_MENOR_CAJ_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_DIA - Monto acumulado de transacciones diarias para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MAXIMO_DIA - Mayor acumulado de transacciones diarias para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta y tipo de rubro cajero menores a 10 USD

Igual CARDID2, SEMANA, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT < 10$:
TAR_MENOR_CAJ_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MINIMO_SEMANA - Menor monto de transacciones semanales para

una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MAXIMO_SEMANA - Mayor acumulado de transacciones semanales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para una misma tarjeta y tipo de rubro cajero menores a 10 USD

Igual CARDID2, MES, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT < 10$:
TAR_MENOR_CAJ_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MES - Monto acumulado de transacciones mensuales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_MAXIMO_MES - Mayor acumulado de transacciones mensuales para una misma tarjeta y tipo de rubro cajero menores a 10 USD
TAR_MENOR_CAJ_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta y tipo de rubro cajero menores a 10 USD

Igual CARDID2, DIA, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT > 500$:
TAR_MAYOR_CAJ_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_DIA - Monto acumulado de transacciones diarias para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MAXIMO_DIA - Mayor acumulado de transacciones diarias para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta y tipo de rubro cajero mayores a 500 USD

Igual CARDID2, SEMANA, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT > 500$:
TAR_MAYOR_CAJ_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MINIMO_SEMANA - Menor monto de transacciones semanales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MAXIMO_SEMANA - Mayor acumulado de transacciones semanales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD

Igual CARDID2, MES, $MCC = 6010$ or $MCC = 6011$ y $AMOUNT > 500$:
TAR_MAYOR_CAJ_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MES - Monto acumulado de transacciones mensuales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD

TAR_MAYOR_CAJ_MAXIMO_MES - Mayor acumulado de transacciones mensuales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD
TAR_MAYOR_CAJ_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta y tipo de rubro cajero mayores a 500 USD

Igual CARDID2, DIA y $AMOUNT < 10$:
TAR_MENOR_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta menores a 10 USD
TAR_MENOR_DIA - Monto acumulado de transacciones diarias para una misma tarjeta menores a 10 USD
TAR_MENOR_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta menores a 10 USD
TAR_MENOR_MAXIMO_DIA - Mayor monto de transacciones diarias para una misma tarjeta menores a 10 USD
TAR_MENOR_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta menores a 10 USD

Igual CARDID2, SEMANA y $AMOUNT < 10$:
TAR_MENOR_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta menores a 10 USD
TAR_MENOR_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta menores a 10 USD
TAR_MENOR_MINIMO_SEMANA - Menor monto de transacciones semanales para una misma tarjeta menores a 10 USD
TAR_MENOR_MAXIMO_SEMANA - Mayor monto de transacciones semanales para una misma tarjeta menores a 10 USD
TAR_MENOR_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para una misma tarjeta menores a 10 USD

Igual CARDID2, MES y $AMOUNT < 10$:
TAR_MENOR_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta menores a 10 USD
TAR_MENOR_MES - Monto acumulado de transacciones mensuales para una misma tarjeta menores a 10 USD
TAR_MENOR_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta menores a 10 USD
TAR_MENOR_MAXIMO_MES - Mayor monto de transacciones mensuales para una misma tarjeta menores a 10 USD
TAR_MENOR_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta menores a 10 USD

Igual CARDID2, DIA y $AMOUNT > 500$:
TAR_MAYOR_CANT_DIA - Cantidad de transacciones diarias para una misma tarjeta mayores a 500 USD
TAR_MAYOR_DIA - Monto acumulado de transacciones diarias para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MINIMO_DIA - Menor monto de transacciones diarias para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MAXIMO_DIA - Mayor monto de transacciones diarias para una misma

tarjeta mayores a 500 USD
TAR_MAYOR_AVERAGE_DIA - Promedio del monto de las transacciones diarias para una misma tarjeta mayores a 500 USD

Igual CARDID2, SEMANA y $AMOUNT > 500$:
TAR_MAYOR_CANT_SEMANA - Cantidad de transacciones semanales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_SEMANA - Monto acumulado de transacciones semanales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MINIMO_SEMANA - Menor monto de transacciones semanales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MAXIMO_SEMANA - Mayor monto de transacciones semanales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para una misma tarjeta mayores a 500 USD

Igual CARDID2, MES y $AMOUNT > 500$:
TAR_MAYOR_CANT_MES - Cantidad de transacciones mensuales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MES - Monto acumulado de transacciones mensuales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MINIMO_MES - Menor monto de transacciones mensuales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_MAXIMO_MES - Mayor monto de transacciones mensuales para una misma tarjeta mayores a 500 USD
TAR_MAYOR_AVERAGE_MES - Promedio del monto de las transacciones mensuales para una misma tarjeta mayores a 500 USD

Igual MERCHANTCODE, DIA y $AMOUNT < 10$:
COM_MENOR_CANT_DIA - Cantidad de transacciones diarias para un mismo comercio menores a 10 USD
COM_MENOR_DIA - Monto acumulado de transacciones diarias para un mismo comercio menores a 10 USD
COM_MENOR_MINIMO_DIA - Menor monto de transacciones diarias para un mismo comercio menores a 10 USD
COM_MENOR_MAXIMO_DIA - Mayor monto de transacciones diarias para un mismo comercio menores a 10 USD
COM_MENOR_AVERAGE_DIA - Promedio del monto de las transacciones diarias para un mismo comercio menores a 10 USD

Igual MERCHANTCODE, SEMANA y $AMOUNT < 10$:
COM_MENOR_CANT_SEMANA - Cantidad de transacciones semanales para un mismo comercio menores a 10 USD
COM_MENOR_SEMANA - Monto acumulado de transacciones semanales para un mismo comercio menores a 10 USD
COM_MENOR_MINIMO_SEMANA - Menor monto de transacciones semanales para un mismo comercio menores a 10 USD
COM_MENOR_MAXIMO_SEMANA - Mayor monto de transacciones semanales para un mismo comercio menores a 10 USD
COM_MENOR_AVERAGE_SEMANA - Promedio del monto de las transacciones semana-

les para un mismo comercio menores a 10 USD

Igual MERCHANTCODE, MES y $AMOUNT < 10$:
COM_MENOR_CANT_MES - Cantidad de transacciones mensuales para un mismo comercio menores a 10 USD
COM_MENOR_MES - Monto acumulado de transacciones mensuales para un mismo comercio menores a 10 USD
COM_MENOR_MINIMO_MES - Menor monto de transacciones mensuales para un mismo comercio menores a 10 USD
COM_MENOR_MAXIMO_MES - Mayor monto de transacciones mensuales para un mismo comercio menores a 10 USD
COM_MENOR_AVERAGE_MES - Promedio del monto de las transacciones mensuales para un mismo comercio menores a 10 USD

Igual MERCHANTCODE, DIA y $AMOUNT > 500$:
COM_MAYOR_CANT_DIA - Cantidad de transacciones diarias para un mismo comercio mayores a 500 USD
COM_MAYOR_DIA - Monto acumulado de transacciones diarias para un mismo comercio mayores a 500 USD
COM_MAYOR_MINIMO_DIA - Menor monto de transacciones diarias para un mismo comercio mayores a 500 USD
COM_MAYOR_MAXIMO_DIA - Mayor monto de transacciones diarias para un mismo comercio mayores a 500 USD
COM_MAYOR_AVERAGE_DIA - Promedio del monto de las transacciones diarias para un mismo comercio mayores a 500 USD

Igual MERCHANTCODE, SEMANA y $AMOUNT > 500$:
COM_MAYOR_CANT_SEMANA - Cantidad de transacciones semanales para un mismo comercio mayores a 500 USD
COM_MAYOR_SEMANA - Monto acumulado de transacciones semanales para un mismo comercio mayores a 500 USD
COM_MAYOR_MINIMO_SEMANA - Menor monto de transacciones semanales para un mismo comercio mayores a 500 USD
COM_MAYOR_MAXIMO_SEMANA - Mayor monto de transacciones semanales para un mismo comercio mayores a 500 USD
COM_MAYOR_AVERAGE_SEMANA - Promedio del monto de las transacciones semanales para un mismo comercio mayores a 500 USD

Igual MERCHANTCODE, MES y $AMOUNT > 500$:
COM_MAYOR_CANT_MES - Cantidad de transacciones mensuales para un mismo comercio mayores a 500 USD
COM_MAYOR_MES - Monto acumulado de transacciones mensuales para un mismo comercio mayores a 500 USD
COM_MAYOR_MINIMO_MES - Menor monto de transacciones mensuales para un mismo comercio mayores a 500 USD
COM_MAYOR_MAXIMO_MES - Mayor monto de transacciones mensuales para un mismo comercio mayores a 500 USD
COM_MAYOR_AVERAGE_MES - Promedio del monto de las transacciones mensuales para un mismo comercio mayores a 500 USD

Apéndice E

Manual del *Batch Manager*

El *Batch Manager* ofrece una interfaz para ejecutar las funcionalidades descritas anteriormente. Los comandos para ejecutar son:

```
execute (-Ranking | -Subset | -DecisionTree | -RandomForest |
-NeuralNetwork | -ArffBuilder | -ROC)
eval (-DecisionTree | -RandomForest | -NeuralNetwork)
filter
score
rule
```

Para solicitar ayuda sobre cómo ejecutar los distintos comandos y sus parámetros alcanza con ingresar `-h` y se obtiene:

```
-------------------------------------------------------------------------
Para obtener ayuda utilizar:
-h eval (-DecisionTree | -RandomForest | -NeuralNetwork)
-h execute (-Ranking | -Subset | -DecisionTree | -RandomForest |
-NeuralNetwork | -ArffBuilder | -ROC)
-h score
-h filter
-h rule
-------------------------------------------------------------------------
```

El mismo mensaje aparece cuando los parámetros ingresados para cada caso son incorrectos. Se describen a continuación los parámetros necesarios para comando, información que puede obtenerse ejecutando el `-h (comando)`.

Ejecución ARFFBuilder
Esta herramienta permite obtener los archivos ARFF a utilizar en la selección de atributos y generación de los diversos modelos. Los parámetros de entrada consisten en los detalles respectivos a la base de datos donde se encuentran los datos y luego a las rutas donde guardar los archivos resultados que serán todos en formato ARFF. Como se ve en los parámetros, se indican los porcentajes a tomar para el conjunto de entrenamiento, el conjunto de test y el conjunto a utilizar en la selección de atributos.

Invocación: `execute -ArffBuilder -TrainPath -TestPath -AttSelPath`

```
-----------------------------------------------------------------------
-ArffBuilder params:
-T tabla: Tabla origen de los datos
-U url: Url de la base de datos (default: jdbc:oracle:thin:@localhost:1521:WEKA)
-D driver: Driver de la base de datos (default: oracle.jdbc.driver.OracleDriver)
-S user: Usuario de la base de datos(default: weka)
-P password: Password del usuario (default: weka)
-N porcentaje conjunto de train: (default: 48)
-M porcentaje conjunto de test: (default: 40)
(El resto del conjunto se utiliza en la selección de atributos)
-Q query: Columnas a tomar en la consulta (de la tabla -T)

-TrainPath trainPath: La ruta donde guardar el conjunto de entrenamiento
-TestPath testPath: La ruta donde guardar el conjunto de test
-AttSelPath attSelPath: La ruta donde guardar el conjunto para selección
de atributos
-----------------------------------------------------------------------
```

Ejecución Filter
Dado que es posible que algunos de los atributos en la base de datos tengan una representación distinta a la que esperan los algoritmos, es necesario poder contar con filtro que reciba como parámetros de entrada el archivo ARFF a filtrar, el tipo de filtro a utilizar y los atributos sobre los cuales actuar. Los atributos se cuentan de $1aN$ en orden de aparición. La salida consiste en el archivo filtrado, también en formato ARFF.

Invocación: `filter (-N|-D|-F) -Input -InputIndex -Result`

```
-----------------------------------------------------------------------
(-N|-D|-F) atributos: -N para pasar atributos numéricos a nominales,-D
para discretizarlos segun el método MDL de Fayyad e Irani y -F para
filtrar por los atributos indicados
-Input input: La ruta del archivo a filtrar
-InputIndex inputIndex: El índice del atributo que indica la clase
del conjunto de entrada
-Output output: La ruta del archivo filtrado
-----------------------------------------------------------------------
```

Ejecución *Ranking*
Permite obtener un nuevo conjunto de datos, seleccionando solo los N atributos que tengan mayor clasificación en el *ranking*. La entrada consiste los parámetros del algoritmo que indican el tipo de *ranking* a realizar, identificado por un número así como la cantidad de atributos del *ranking* a retornar. Se debe indicar la ruta donde guardar los resultados, ya que serán utilizados luego para filtrar el conjunto de entrenamiento y el de *test*. Es necesario indicar la clase del conjunto de entrada ingresando el índice del atributo de clase, cuyo rango varía

de $0aN$, en orden de aparición en el archivo (esto se mantiene para los comando posteriores).

Invocación: `execute -Ranking -Input -InputIndex -Result`

```
------------------------------------------------------------------------
-Ranking params: los parámetros posibles se muestran a continuación
-R RankingType (default 1): 1 - InfoGain; 2 - GainRatio; 3 - ChiSquared
-N Cantidad de atributos a retornar (default 15)

-Input input: La ruta al conjunto de entrada
-InputIndex inputIndex: El índice del atributo que indica la clase
del conjunto de entrada
-Result resPath: La ruta donde guardar los resultados del ranking (se
concatena '.RES' internamente)
------------------------------------------------------------------------
```

Ejecución *Subset Selection*
Este algoritmo no recibe parámetros y ejecuta la selección de atributos utilizando una búsqueda del conjunto *Best First*, en ambas direcciones (*forward* y *backward*) y evaluando el subconjunto obtenido tomando en cuenta la habilidad predictiva del atributo individual así como la correlación entre ellos. Se obtiene, como en el caso anterior los resultados que serán utilizados luego para filtrar el conjunto de entrenamiento y el de *test*. A diferencia que en caso anterior, se obtienen dos archivos, para que el usuario pueda filtrar tomando la unión de resultados. Debe indicarse el conjunto de entrada y el atributo de clase.

Invocación: `execute -Subset -Input -InputIndex -Result`

```
------------------------------------------------------------------------
(Subset no recibe parámetros)
-Input input: La ruta al conjunto de entrada
-InputIndex inputIndex: El índice del atributo que indica la clase
del conjunto de entrada
-Result resPath: La ruta donde guardar los resultados del ranking (se
concatena '.RES' internamente)
------------------------------------------------------------------------
```

Ejecución modelo *Decision Tree*
Con la creación de este modelo, se podrán obtener tanto reglas para clasificar las transacciones como un modelo posible de ser ejecutado con el *scorer* como detallaremos posteriormente. Como entrada recibe la referencia al conjunto de entrenamiento y el de *test* para construir el modelo y los respectivos índices de clase, así como los parámetros del algoritmo que indican el tipo de árbol a utilizar. El resultado se guardará en la ruta indicada seguido por .RES y el modelo en la ruta indicada seguido de .MOD.

Invocación: `execute -DecisionTree -TrainPath -TrainIndex -TestPath -TestIndex`

```
-Result

------------------------------------------------------------------------
-DecisionTree params: los parámetros posibles se muestran a continuación
-T :TreeType (default 1): 1 - TreeJ48 Unpruned; 2 - TreeJ48; 3 - TreeJ48
Split; 4 - RandomTree

-TrainPath trainPath: La ruta al conjunto de entrenamiento
-TrainIndex testIndex: El índice del atributo que indica la clase del
conjunto de entrenamiento
-TestPath testPath: La ruta al conjunto de test
-TestIndex testIndex: El índice del atributo que indica la clase del
conjunto de test
-Result resPath: La ruta donde guardar los resultados del modelo y
la que se utiliza para guardar el modelo (internamente se concatena
la extensión '.RES' y '.MOD' respectivamente)
------------------------------------------------------------------------
```

Ejecución modelo *Neural Network*
El modelo generado en este caso se utilizará para clasificar instancias o formar parte del *scorer*. Como entrada recibe la referencia al conjunto de entrenamiento y el de *test* para construir el modelo y los respectivos índices de clase. El algoritmo recibe varios parámetros el usuario debe ingresar también.

Invocación: `execute -NeuralNetwork -TrainPath -TrainIndex -TestPath -TestIndex -Result`

```
------------------------------------------------------------------------
-NeuralNetworks params: los parámetros posibles se muestran a continuación
-L num: Setea la tasa de aprendizaje (learning rate - default 0,3)
-M num: Setea el momento (momentum - default 0,2)
-N num: Setea el número de epochs de entrenamiento. (default 500)
-H str: Setea el número de nodos a ser utilizados en cada capa. Cada
número representa su propia capa y el número de nodos de esa capa.
Cada número debe ser separado por comas. Existen los siguientes wildcards
'a' = (número de atributos + número de clases) / 2, 'i' = número de
atributos, 'o' = número de clases, and 't' = número de atributos +
número de clases.

-TrainPath trainPath: La ruta al conjunto de entrenamiento
-TrainIndex testIndex: El índice del atributo que indica la clase del
conjunto de entrenamiento
-TestPath testPath: La ruta al conjunto de test
-TestIndex testIndex: El índice del atributo que indica la clase del
conjunto de test
-Result resPath: La ruta donde guardar los resultados del modelo y
la que se utiliza para guardar el modelo (internamente se concatena
la extensión '.RES' y '.MOD' respectivamente)
------------------------------------------------------------------------
```

Ejecución modelo *Random Forest*
El modelo generado servirá como en el caso anterior, para evaluar instancias sin clasificar y formar parte del *scorer* a construir. Este algoritmo recibe un parámetro para indicar la cantidad de árboles a generar en el bosque aleatorio. Al igual que en los casos anteriores, recibe como entrada la referencia al conjunto de entrenamiento y el de *test* para construir el modelo y los respectivos índices de clase.

Invocación: `execute -RandomForest -TrainPath -TrainIndex -TestPath -TestIndex -Result`

```
------------------------------------------------------------------------
-RandomForest params: los parámetros posibles se muestran a continuación
Número de arboles a construir en el bosque, por ej: 50

-TrainPath trainPath: La ruta al conjunto de entrenamiento
-TrainIndex testIndex: El índice del atributo que indica la clase del
conjunto de entrenamiento
-TestPath testPath: La ruta al conjunto de test
-TestIndex testIndex: El índice del atributo que indica la clase del
conjunto de test
-Result resPath: La ruta donde guardar los resultados del modelo y
la que se utiliza para guardar el modelo (internamente se concatena
la extensión '.RES' y '.MOD' respectivamente)
------------------------------------------------------------------------
```

Ejecución de la evaluación de una instancia sin clasificar
El objetivo de nuestro estudio es poder clasificar una instancia y determinar si es fraudulenta o no. Para esto es necesario ingresar la ruta donde se encuentra el modelo a utilizar (creado con cualquier de los tres comandos anteriores) así como la ruta al archivo ARFF que contiene una sola instancia. Esta limitación viene dada porque la intención es utilizar el scorer para el fin de la evaluación de varias instancias, pero de todas maneras se consideró conveniente implementar esta funcionalidad. Para el caso de querer evaluar un conjunto de instancias puede utilizarse el scorer como se verá más adelante utilizando solamente un modelo (en vez de una combinación de modelos). Es necesario ingresar en qué atributo se encuentra la clase aunque éste campo esté claramente vacío a no tener la clasificación. En el archivo ARFF esto se indica con el símbolo '?'.

Invocación: `eval (-RandomForest|-DecisionTree|-NeuralNetwork) -Input -InputIndex -Result`

```
------------------------------------------------------------------------
(-RandomForest|-DecisionTree|-NeuralNetwork): Corresponde al tipo de
algoritmo a invocar
-Input input: La ruta del archivo con UNA instancia sin clasificar;
si son mas instancias solo clasifica la primera"); -Result result:
La ruta donde se guarda el modelo a utilizar (sin '.MOD')
```

```
----------------------------------------------------------------------
```

Ejecución para observar la curva de ROC de un modelo
Para ejecutar esta funcionalidad se ingresa el tipo de modelo, la ruta donde está guardado modelo y un conjunto indicando su atributo de clase a ser utilizado en la construcción de la curva, por lo general, el propio conjunto de test.

Invocación: `execute -ROC (RandomForest|NeuralNetwork) -TestPath -TestIndex -Input`

```
----------------------------------------------------------------------
(-RandomForest|-NeuralNetwork): Corresponde al tipo de modelo
-TestPath testPath: La ruta al conjunto de test para construir la curva
de ROC
-TestIndex testIndex: El índice del atributo que indica la clase del
conjunto de test
-Result result: La ruta donde se guarda el modelo a utilizar (sin '.MOD')
----------------------------------------------------------------------
```

Ejecución para obtener las N *mejores* reglas
Para la visualización de las reglas obtenidas mediante la ejecución de alguno de los tipos de árboles de decisión, es necesario indicar la ruta al modelo, la ruta donde guardar el resultado en formato CSV (en inglés *Coma Separated Value*), la cantidad de reglas a retornar, el criterio de selección y el tipo de clasificación a considerar. Si se quiere obtener todas las reglas se debe utilizar 0 en la cantidad de reglas. Se puede optar por tres criterios de selección: medida-f, precisión y cobertura de la regla. Se pueden considerar reglas que clasifiquen como fraude, no fraude o todas las reglas.

Invocación: `rule -Input -Result -OrderBy -CntRules -Clasif`

```
----------------------------------------------------------------------
-Input input: La ruta del archivo de resultado de un modelo de árbol
a ser parseado para obtener las reglas.
-Result result: La ruta y nombre donde se guardará el resultado de
la ejecución sin extension, el archivo generado es de extensión CSV
-OrderBy order by: Criterio de ordenamiento,(0|1|2). 0 - f measure;1
- precision; 2 - recall
-CntRules cantidad de reglas: Cantidad de reglas a ser desplegadas,
por ejemplo, las 10 primeras.
-Clasif clasificación: Tipo de reglas a considerar,(0|1|2). 0 - no
fraude; 1 - fraude; 2 - todas.
----------------------------------------------------------------------
```

Ejecución del *Scorer*
El scorer permite obtener el puntaje para un conjunto de transacciones utilizando cada uno de los modelos ingresados como parámetros. Los modelos deben

obtenerse como se explica en los pasos previos. Como en casos anteriores, se necesita indicar la clase del archivo de entrada aunque ésta no contenga información. El resultado es un archivo en formato CSV que contiene para cada transacción el puntaje respectivo a cada clase para cada uno de los algoritmos utilizados.
Invocación: `score -Models -Input -InputIndex -Result`

```
-----------------------------------------------------------------------
-Models: Modelos a utilizar en la evaluación, separados por punto y
coma (sin '.MOD'). Ejemplo: path1;path2;path3
-Input input: La ruta del archivo con una o VARIAS instancias sin clasificar
-InputIndex: Índice del atributo que indica la clase fraude en el archivo
indicado en Input. Dicho índice es obligatorio y el atributo es ignorado
durante el cálculo del puntaje de riesgo.
-Result result: La ruta y nombre donde se guardará el resultado de
la ejecución sin extensión, el archivo generado es de extensión CSV
-----------------------------------------------------------------------
```

E.1. Ejemplo de secuencia de ejecución

A continuación se detalla como se ejecutaría una secuencia de ejecución completa desde línea de comandos utilizando el `BatchManager.jar`.

Suponiendo que los datos se encuentran en la tabla "transacciones" en la ubicación default de la base de datos, se opta por el default en los porcentaje de *train* y *test* y cada transacción real cuenta con cuatro atributos más la clase:

```
java -Xmx1024M -jar BatchManager execute -ArffBuilder ‘‘-T transacciones
-Q atr1, atr2, atr3, atr4, clase’’ -TrainPath ‘‘C:\train.ARFF’’ -TestPath
‘‘C:\test.ARFF’’ -AttSelPath ‘‘C:\attSel.ARFF’’
```

Suponiendo que el atributo de clase debe ser convertido a nominal por estar representado como numérico en la base:

```
java -Xmx1024M -jar BatchManager filter -N 5 -Input ‘‘C:\train.ARFF’’
-InputIndex 4 -Output ‘‘C:\train_N.ARFF’’
java -Xmx1024M -jar BatchManager filter -N 5 -Input ‘‘C:\test.ARFF’’
-InputIndex 4 -Output ‘‘C:\test_N.ARFF’’
java -Xmx1024M -jar BatchManager filter -N 5 -Input ‘‘C:\attSel.ARFF’’
-InputIndex 4 -Output ‘‘C:\attSel_N.ARFF’’
```

Suponiendo que los atributos atr1 y atr3 son numéricos y deben ser discretizados previo a la aplicación de algoritmos:

```
java -Xmx1024M -jar BatchManager filter -D ‘‘1,3’’ -Input ‘‘C:\train_N.ARFF’’
-InputIndex 4 -Output ‘‘C:\train_D.ARFF’’
java -Xmx1024M -jar BatchManager filter -D ‘‘1,3’’ -Input ‘‘C:\test_N.ARFF’’
-InputIndex 4 -Output ‘‘C:\test_D.ARFF’’
java -Xmx1024M -jar BatchManager filter -D ‘‘1,3’’ -Input ‘‘C:\attSel_N.ARFF’’
```

```
-InputIndex 4 -Output ‘‘C:\attSel_D.ARFF’’
```

Suponiendo que se quiere utilizar la selección de atributos mediante la aplicación de *ranking* con *Chi-Squared*:

```
java -Xmx1024M -jar BatchManager execute -Ranking 3 -Input ‘‘C:\attSel_D.ARFF’’
-InputIndex 4 -Result ‘‘C:\attSel.RES’’
```

Se filtra el conjunto de entrenamiento por los atributos obtenidos previamente. Supongamos que se obtuvieron el 1, 3, 4 y 5 (la clase debe incluirse):

```
java -Xmx1024M -jar BatchManager filter -F ‘‘1,3,4,5’’ -Input ‘‘C:\test_D.ARFF’’
-InputIndex 4 -Output ‘‘C:\test_ChiSq.ARFF’’
java -Xmx1024M -jar BatchManager filter -F ‘‘1,3,4,5’’ -Input ‘‘C:\train_D.ARFF’’
-InputIndex 4 -Output ‘‘C:\train_ChiSq.ARFF’’
```

Ejecución de los tres modelos, suponiendo que se quiere ejecutar el tipo J48 Unpruned para árboles de decisión, un bosque aleatorio con 100 árboles para *random forest* y una red neuronal artificial de una capa oculta. Notar que luego del filtro, el índice del atributo de clase cambia:

```
java -Xmx1024M -jar BatchManager execute -DecisionTree ‘‘-T 1’’ -TrainPath
‘‘C:\train_ChiSq.ARFF’’ -TrainIndex 3 -TestPath
‘‘C:\test_ChiSq.ARFF’’ -TestIndex 3 -Result ‘‘C:\DecisionTree_ChiSq’’
java -Xmx1024M -jar BatchManager execute -RandomForest 100 -TrainPath
‘‘C:\train_ChiSq.ARFF’’ -TrainIndex 3 -TestPath‘‘C:\test_ChiSq.ARF’’
-TestIndex 3 -Result ‘‘C:\RndForest_ChiSq’
java -Xmx1024M -jar BatchManager execute -NeuralNetwork ‘‘-N 3000 -H
a -L 0.3 -M 0.2’’ -TrainPath ‘‘C:\train_ChiSq.ARFF’’ -TrainIndex 3 -TestPath
‘‘C:\test_ChiSq.ARF’’ -TestIndex 3 -Result ‘‘C:\NeuralNetwork_ChiSq’’
```

Se obtienen las reglas, suponiendo que queramos 15 reglas que clasifiquen instancias como fraude por medida-f:

```
java -Xmx1024M -jar BatchManager rule -Input ‘‘C:\DecisionTree_ChiSq’’
-Result ‘‘C:\rulesTree_ChiSq.txt’’ -OrderBy 0 -CntRules 15 -Clasif 1
```

Clasificación de instancias del archivo `unlabeled.ARFF` mediante el scorer:

```
java -Xmx1024M -jar BatchManager score -Models ‘‘C:\DecisionTree_ChiSq;
C:\RndForest_ChiSq; C:\NeuralNetwork_ChiSq’’ -Input ‘‘C:\unlabeled.ARFF’’
-InputIndex 3 -Result ‘‘C:\score_ChiSq’’
```

Apéndice F

Detalle Resultados Obtenidos

		J48		J48 Split		J48 Unpruned		Random	
		NF	F	NF	F	NF	F	NF	F
Total	NF	82.967	1.761	82.166	**1.535**	**83.983**	1.751	80.059	**1.535**
	F	2.138	1.134	2.939	**1.360**	4.736	1.144	5.046	**1.360**
Gain Ratio	NF	83.935	1.836	83.413	1.753	83.511	1.863	82.578	1.968
	F	1.170	1.059	1.692	1.142	**1.122**	1.032	2.527	927
Info Gain	NF	83.434	1.754	83.434	1.754	82.979	1.759	80.175	1.654
	F	1.671	1.141	1.671	1.141	1.594	1.136	4.930	1.241
Chi Sqr	NF	83.413	1.753	83.935	1.836	80.649	1.718	79.922	1.591
	F	1.692	1.142	1.170	1.059	2.126	1.177	5.183	1.304
Subset	NF	83.299	1.744	83.299	1.744	83.154	1.737	79.464	1.663
	F	1.806	1.151	1.806	1.151	1.951	1.158	5.641	1.232

Cuadro F.1: Matrices de Confusión para Áboles de Decisión

		a neuronas		t neuronas		t,a neuronas	
		NF	F	NF	F	NF	F
Gain Ratio	NF	84.047	1.889	**84.138**	1.932	84.062	1.890
	F	1.058	1.006	**967**	963	1.043	1.005
Info Gain	NF	83.207	1.819	83.329	1.824	83.101	1.795
	F	1.898	1.076	1.776	1.071	2.004	1.100
Chi Sqr	NF	83.206	1.750	83.172	1.735	83.068	1.740
	F	1.899	1.145	1.933	1.160	2.037	1.155
Subset	NF	82.870	1.695	82.758	**1.665**	83.252	1.765
	F	2.235	1.200	2.347	**1.230**	1.853	1.130

Cuadro F.2: Matrices de confusión para Perceptrón Multicapa con 3000 *epochs* y tasa de aprendizaje 0,3
a =(atributos + clases)/2, t = (atributos + clases)

		25		50		100	
		NF	F	NF	F	NF	F
~~Info Gain~~	NF	81.935	1.613	82.101	1.637	82.179	1.629
	F	3.170	1.282	3.004	1.258	2.926	1.266
~~Gain Ratio~~	NF	**82.703**	1.945	82.578	1.945	82.684	1.951
	F	**2.402**	950	2.527	950	2.421	944
~~Chi Sq~~	NF	82.008	**1.601**	82.222	1.619	82.297	1.622
	F	3.097	**1.294**	2.883	1.276	2.898	1.276
~~Subset~~	NF	81.803	1.653	82.011	1.655	81.977	1.656
	F	3.302	1.242	3.094	1.240	3.128	1.239

Cuadro F.3: Matrices de Confusión para *Random Forest*

					Fraude			No Fraude		
Atributos	Neurs.	Epochs	LR	AUC	Prec.	Rec.	Med.-f	Prec.	Rec.	Med.-f
ChiSQ	a	3000	0,3	0,851	0,376	0,396	0,386	0,979	0,978	0,979
ChiSQ	a	500	0,3	0,841	0,339	0,391	0,363	0,979	0,974	0,977
ChiSQ	a	3000	0,06	0,846	0,38	0,4	0,39	**0,98**	0,978	0,979
ChiSQ	a	500	0,06	0,84	0,359	0,371	0,365	0,979	0,978	0,978
ChiSQ	t	3000	0,3	0,851	0,375	0,401	0,387	**0,98**	0,977	0,978
ChiSQ	t	500	0,3	0,841	0,343	0,389	0,364	0,979	0,975	0,977
ChiSQ	t	3000	0,06	0,846	0,379	0,4	0,389	**0,98**	0,978	0,979
ChiSQ	t	500	0,06	0,84	0,355	0,375	0,365	0,979	0,977	0,978
ChiSQ	t,a	3000	0,3	**0,852**	0,362	0,399	0,379	0,979	0,976	0,978
ChiSQ	t,a	500	0,3	0,841	0,341	0,388	0,363	0,979	0,975	0,977
ChiSQ	t,a	3000	0,06	0,847	0,369	0,414	0,39	**0,98**	0,976	0,978
ChiSQ	t,a	500	0,06	0,841	0,361	0,37	0,365	0,979	0,978	0,978
GainRatio	a	3000	0,3	0,822	0,487	0,347	0,406	0,978	0,988	0,983
GainRatio	a	500	0,3	0,821	0,543	0,267	0,356	0,975	0,992	**0,984**
GainRatio	a	3000	0,06	0,818	0,507	0,317	0,391	0,977	0,99	0,983
GainRatio	a	500	0,06	0,823	0,545	0,257	0,35	0,975	**0,993**	**0,984**
GainRatio	t	3000	0,3	0,815	0,499	0,333	0,399	0,978	0,989	0,983
GainRatio	t	500	0,3	0,819	0,534	0,268	0,357	0,976	0,992	**0,984**
GainRatio	t	3000	0,06	0,824	0,502	0,32	0,391	0,977	0,989	0,983
GainRatio	t	500	0,06	0,823	0,54	0,262	0,353	0,975	0,992	**0,984**
GainRatio	t,a	3000	0,3	0,824	0,491	0,347	**0,407**	0,978	0,988	0,983
GainRatio	t,a	500	0,3	0,82	**0,546**	0,257	0,349	0,975	**0,993**	**0,984**
GainRatio	t,a	3000	0,06	0,824	0,49	0,344	0,404	0,978	0,988	0,983
GainRatio	t,a	500	0,06	0,822	0,535	0,265	0,354	0,975	0,992	**0,984**
InfoGain	a	3000	0,3	0,837	0,362	0,372	0,367	0,979	0,978	0,978
InfoGain	a	500	0,3	0,825	0,436	0,298	0,354	0,976	0,987	0,982
InfoGain	a	3000	0,06	0,837	0,367	0,372	0,369	0,979	0,978	0,978
InfoGain	a	500	0,06	0,821	0,441	0,296	0,354	0,976	0,987	0,982
InfoGain	t	3000	0,3	0,837	0,376	0,37	0,373	0,979	0,979	0,979
InfoGain	t	500	0,3	0,825	0,438	0,297	0,354	0,976	0,987	0,982
InfoGain	t	3000	0,06	0,837	0,37	0,37	0,37	0,979	0,979	0,979
InfoGain	t	500	0,06	0,821	0,441	0,296	0,354	0,976	0,987	0,982
InfoGain	t,a	3000	0,3	0,836	0,354	0,38	0,367	0,979	0,976	0,978
InfoGain	t,a	500	0,3	0,823	0,439	0,297	0,355	0,976	0,987	0,982
InfoGain	t,a	3000	0,06	0,837	0,362	0,371	0,367	0,979	0,978	0,978
InfoGain	t,a	500	0,06	0,823	0,434	0,303	0,357	0,977	0,987	0,982
SubSetBF	a	3000	0,3	0,849	0,349	0,415	0,379	**0,98**	0,974	0,977
SubSetBF	a	500	0,3	0,849	0,388	0,365	0,376	0,978	0,98	0,979
SubSetBF	a	3000	0,06	0,849	0,354	0,404	0,378	**0,98**	0,975	0,977
SubSetBF	a	500	0,06	0,848	0,385	0,377	0,381	0,979	0,98	0,979
SubSetBF	t	3000	0,3	0,848	0,344	**0,425**	0,38	0,98	0,972	0,976
SubSetBF	t	500	0,3	0,848	0,377	0,388	0,382	0,979	0,978	0,979
SubSetBF	t	3000	0,06	0,849	0,351	0,411	0,378	**0,98**	0,974	0,977
SubSetBF	t	500	0,06	0,848	0,386	0,377	0,381	0,979	0,98	0,979
SubSetBF	t,a	3000	0,3	0,848	0,379	0,39	0,384	0,979	0,978	0,979
SubSetBF	t,a	500	0,3	0,848	0,385	0,363	0,374	0,978	0,98	0,979
SubSetBF	t,a	3000	0,06	**0,852**	0,366	0,399	0,382	0,979	0,977	0,978
SubSetBF	t,a	500	0,06	0,85	0,405	0,36	0,381	0,978	0,982	0,98

Cuadro F.4: Resultados completos de perceptrón multicapa
a =(atributos + clases)/2, t = (atributos + clases)
LR = Learning Rate, AUC = Area Under Roc Curve

Apéndice G

WEKA

WEKA soporta distintos formatos de origen de datos, como ser formato CSV, C4.5 ó un formato de archivo denominado *arff (Attribute-Relation File Format)*. También puede recibir los datos directamente desde una base de datos utilizando el protocolo JDBC [10].

Es posible aplicar una gran diversidad de filtros sobre los datos para realizar todo tipo de transformaciones, como ser discretizar, normalizar, agregar, modificar o eliminar atributos así como combinar varios filtros. Los filtros pueden ser aplicados a nivel de atributos o de instancias.

WEKA permite clasificar por varios métodos los datos ya cargados. Entre los clasificadores disponibles se encuentran árboles de decisión, aprendizaje de reglas, *Naive Bayes*, tablas de decisión y perceptrón multicapa. Existen cuatro modos de prueba:

- *Use training set* donde se entrena el método con todos los datos disponibles y luego se aplica sobre los mismos datos.
- *Supplied test set* donde se entrena el método con todos los datos iniciales y luego se aplica sobre otros datos .
- *Cross-validation* donde se realiza una validación cruzada estratificada de un número de particiones dado.
- *Percentage split* donde se define un porcentaje con el que se construye el clasificador y se aplica a la parte restante.

De manera similar a la clasificación es posible realizar *Clustering* y búsqueda de asociaciones.

También es posible realizar selección atributos. Se debe seleccionar el método de evaluación de atributos. Este método evalúa cada uno de los casos y asigna a cada atributo un peso específico. Es necesario además, elegir el método de búsqueda que será el encargado de generar el espacio de pruebas.

Existen tres formas distintas de utilizar WEKA, desde la línea de comandos, desde una de las cuatro interfaces de usuario o desde un programa Java.

Al invocar los algoritmos incluidos en WEKA desde la línea de comandos de MS-DOS los resultados se muestran únicamente en modo texto.

El modo o interfaz *Simple CLI* permite introducir comandos para ejecutar las operaciones soportadas en WEKA en forma directa.

El modo *Explorer* es el modo más utilizado y permite realizar operaciones sobre un sólo conjunto de datos. Al abrir un conjunto de datos se muestra en la ventana todos los atributos, un resumen con estadísticas y su representación gráfica. Tiene un modo de visualización que muestra gráficamente la distribución de todos los atributos mostrando gráficas en dos dimensiones, en las que va representando en los ejes todos los posibles pares de combinaciones de los atributos. Este modo nos permite ver correlaciones y asociaciones entre los atributos de una forma gráfica.

El modo *Experimenter* es un modo útil para aplicar uno o varios métodos de clasificación sobre un gran conjunto de datos y obtener índices estadísticos pudiendo guardar los resultados. Hay tres tipos de validación: validación-cruzada estratificada, entrenamiento con un porcentaje de la población tomando ese porcentaje de forma aleatoria y entrenamiento con un porcentaje de la población tomando el porcentaje de forma ordenada.

El modo *Knowledge flow* muestra de una forma explícita el funcionamiento interno del programa. Su funcionamiento es gráfico y se basa en situar en el panel de trabajo, elementos base de manera que creemos un circuito que defina nuestro experimento. Se pueden agregar al circuito fuentes de los datos (en este caso no es posible trabajar con los datos directamente desde una base de datos), se pueden utilizar clasificadores y es posible ver el resultado de manera gráfica o texto.

WEKA es un programa en continuo desarrollo por lo que las interfaces han evolucionado de manera independiente, por lo que hay ciertas funcionalidades que no se encuentran disponibles en todos las interfaces.

Bibliografía

[1] *APACS*. Fraud: the facts. `http://www.apacs.org.uk/resources_publications/documents/FraudtheFacts2007.pdf`, 2007. (último acceso en febrero de 2009).

[2] A. C Davison y D. V. Hinkley. *Bootstrap Methods and Their Application (Cambridge Series in Statistical and Probabilistic Mathematics , No 1)*. Cambridge University Press, Octubre 1997.

[3] Adriana Jorba, Marcelo Lozano, José Moraes y Emiliano Sacchi. Sistema de identificación de fraude. 2007.

[4] Anna Bosch, Andrew Zisserman y Xavier Munoz. Image classification using random forests and ferns. In *IEEE 11th International Conference on Computer Vision, ICCV 2007.*, pages 1–8, 2007.

[5] Ben Krose y Patrick van der Smagt. *An Introduction to Neural Networks*. The University of Amsterdam, eighth edition, 1996.

[6] Bernard Widrow y Marcian E. Hoff. Adaptive switching circuits. pages 123–134, 1988. Referenciado desde el libro Ben Krose, Patrick van der Smagt - An Introduction to Neural Networks.

[7] C.E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[8] Christopher D. Manning, Prabhakar Raghavan y Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Julio 2008.

[9] Colin Fyfe. Artificial neural networks. Material del Departamento de Computos y Sistemas de Información, The University of Paisley, 1996.

[10] Diego García Morate. Manual de weka. `http://www.metaemotion.com/diego.garcia.morate/download/weka.pdf`. (último acceso en abril del 2009).

[11] Diego Rivero, Álvaro Rodríguez y Dieter Spangenberg. Documentación del proyecto de grado. Junio 2006.

[12] EMVCo LLC. `http://www.emvco.com`. (último acceso en febrero del 2009).

[13] Fair Isaac. http://www.fairisaac.com. (último acceso en febrero del 2009).

[14] Fair Isaac. Credit scoring. http://www.fairisaac.com/NR/exeres/166A27B7-4AFE-44A2-B5F5-2F3D530A9F67,frameless.htm. (último acceso en abril del 2009).

[15] Fan Yang, Hua-zhen Wang, Hong Mi y Cheng-de Lin y Wei-wen Cai. Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC Bioinformatics*, 10(Suppl 1):S22, 2009.

[16] Fayyad e Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Uncertainty in AI*, pages 1022–1027, 1993.

[17] Frederick Livingston. Implementing breimans random forest algorithm into weka. http://s112088960.onlinehome.us/ml2005/Conference%20Paper/ConferencePaper-Fred.pdf, 11 2005. (último acceso en marzo del 2009).

[18] Friedrich Lange y Gerhard Hirzinger. Application of multilayer perceptrons to decouple the dynamical beaviour of robot links. In *In Int. Conf. on Artificial Neural Networks ICANN*, pages 9–13, 1995.

[19] George Forman, Isabelle Guyon y André Elisseeff. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[20] Huiqing Liu, Jinyan Li y Limsoon Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.

[21] Hung Son Nguyen y Sinh Hoa Nguyen. Approximated measures in construction of decision trees from large databases. pages 595–604, Amsterdam, The Netherlands, The Netherlands, 2003. IOS Press.

[22] Ian H. Witten, Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., second edition, 2005.

[23] Instituto de Computación. Presentación de propuesta de proyecto de grado. detección de patrones de comportamiento regulares y fraudulentos sobre un conjunto de transacciones financieras. Abril 2008.

[24] Isabelle Guyon, A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[25] ISO 8583. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31628. (último acceso en febrero del 2009).

[26] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[27] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[28] John K. Williams y Jennifer Abernethy. Using random forest and fuzzy logic for automated storm type identification. *Sixth Conference on Artificial Intelligence Applications to Environmental Science*, 2008.

[29] John Makhoul, Francis Kubala, Richard Schwartz y Ralph Weischedel. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252, 1999.

[30] Jon Simmons. Analysis of machine learning meta learners: Meritocracies vs. democracies. `http://www.docstoc.com/docs/2365668/Analysis-of-Machine-Learning-Meta-Learners-Meritocracies-vs`, 2005.

[31] Keechul Jung, Kwang In Kim, Takeshi Kurata, Masakastu Kourogi y Junghyun Han. Text scanner with text detection technology on image sequences. In *In Proc. 16th International Conference on Pattern Recognition (ICPR*, pages 473–476, 2002.

[32] Lantel Group. Códigos iso países. `http://www.segurosaduana.com/utilcodigos.htm`. (último acceso en febrero del 2009).

[33] Leo Breiman. Out of bag estimation. University of California, Berkley. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3712`.

[34] Leo Breiman y E. Schapire. Random forests. In *Machine Learning*, pages 5–32, 2001.

[35] Mark A. Hall. Correlation-based feature selection for machine learning. Technical report, University of Waikato, 1998.

[36] Michael I. Jordan, Christopher M. Bishop. Neural networks. In *CRC Handbook of Computer Science*. MIT Press, 1996.

[37] Pat Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall symposium on relevance*, pages 140–144. AAAI Press, 1994.

[38] Paytrue Solutions. `http://www.paytrue.com`. (último acceso en febrero del 2009).

[39] Peter Burns y Anne Stanley. Fraud management in the credit card industry. Payment Cards Center Discussion Paper 02-05, Federal Reserve Bank of Philadelphia, 2002. `http://ideas.repec.org/p/fip/fedpdp/02-05.html`.

[40] Philip K. Chan, Wei Fan, Andreas Prodromidis y Salvatore J. Stolfo. Distributed data mining in credit card fraud detection. *Intelligent Systems and Their Applications, IEEE*, 14:67–74, 1999.

[41] Philip K. Chan y Salvatore J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168. AAAI Press, 1998.

[42] Richard J. Bolton, David J. Hand y David J. H. Statistical fraud detection: A review. In *Statistical Science*, volume 17, pages 235–255, 2002.

[43] Ron Kohavi y George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[44] Salvatore J. Stolfo, David W. Fan, Wenke Lee, Andreas L. Prodromidis y Philip K. Chan. Credit card fraud detection using meta-learning: Issues and initial results. In *Proceedings of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, 1997.

[45] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel y Bernard Manderick. Credit card fraud detection using bayesian and neural networks. In *Interactive image-guided neurosurgery. American Association Neurological Surgeons*, pages 261–270, 1993.

[46] Simon Haykin. *Neural Networks: A comprehensive Foundation*. Pearson Education, Inc. Pearson Prentice Hall, second edition, 1999.

[47] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[48] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[49] Toni Giorgino. An introduction to text classification. Technical report, Bio-Medical Informatics, Diciembre 2004. (Disertación final para SAFI).

[50] Visa. Merchant category classification (mcc) codes directory. `http://www.usda.gov/procurement/card/cardx/mcc.pdf`. (último acceso en febrero del 2009).

[51] W. Duch, T. Wieczorek, J. Biesiada y M. Blachnik. Comparison of feature ranking methods based on information entropy. volume 2, pages 1415–1419 vol.2. IEEE International Joint Conference on Neural Networks, Julio 2004.

[52] W. Duch, T. Winiarski, J. Biesiada y A. Kachel. Feature ranking, selection and discretization. In *Proceedings of Int. Conf. on Artificial Neural Networks (ICANN)*, pages 251–254, Istanbul, 2003. Bogazici University Press.

[53] Waikato Enviroment for Knowledge Analysis. `http://www.cs.waikato.ac.nz/ml/weka`. (último acceso en abril del 2009).

[54] Wesley Kenneth Wilhelm. Manager, Strategic Planning. Fair Isaac Company. The fraud management lifecycle theory: A holistic approach to fraud management. *Journal of Economic Crime Management*, 2, Abril 2004. `http://www.utica.edu/academic/institutes/ecii/jecm/`.

[55] Yiming Yang y Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997.

[56] Yves Moreau, Ellen Lerouge, Herman Verrelst, Christof Stormann y Peter Burge. A hybrid system for fraud detection in mobile communication. In *European Symposium on Artificial Neural Networks*, pages 447–454, 1999.

Facultad de Ingeniería

Ingeniería en Computación

Tutores : Msc. Diego Garat, Msc. Guillermo Moncecchi

Estado del Arte

Detección de patrones de comportamiento regulares y fraudulentos sobre un conjunto de transacciones financieras

Lucía Adinolfi, Martín Ferreira, Lucía Ramos

Montevideo, 15 de Julio de 2008

Índice general

Capítulo 1

Introducción

Una problemática recurrente en el negocio de los medios de pago es el fraude. Ya sea en los pagos realizados por Internet, mediante tarjeta de banda magnética, chip, etc., las organizaciones emisoras de medios de pago deben afrontar pérdidas financieras importantes debidas al fraude (transacciones realizadas en forma ilegal, que el emisor debe pagar pero no puede cobrar pues son repudiadas por el responsable del medio de pago) [4].

A medida que los emisores de medios de pago establecen medidas de control y detección de fraude, las formas de realizar fraude evolucionan también de forma de sobrepasar estos controles, generando así nuevos patrones de transacciones fraudulentas.

En general, el trabajo de monitoreo de transacciones fraudulentas es realizado conjuntamente por una herramienta de software que genera alertas al detectar transacciones que se ajustan a ciertos perfiles de fraude, y un analista de riesgo que decide cuáles de esas alertas merecen ser investigadas y cuáles no.

PayTrue Solutions es una empresa de software dedicada al diseño de soluciones integrales para el negocio de medios de pago que cuenta actualmente con un conjunto de soluciones específicamente diseñadas para el monitoreo de transacciones financieras y no financieras cuyo objetivo es detectar patrones predefinidos considerados como fraudulentos. La aparición de uno de estos patrones dentro del sistema deriva en la generación de una alerta, la cual es posteriormente analizada por un equipo de personas especializadas que determinarán si esta alerta es realmente fraude o no.

Lógicamente, es deseable que la efectividad (cuánto fraude del total real está siendo detectado por el sistema) sea lo más alta posible y el *Falso/Positivo* (cantidad de alertas que finalmente no resultan en fraude real) sea lo más bajo posible. Hoy en día, la efectividad y el *Falso/Positivo* de las alertas generadas dependen del talento y conocimiento de mercado que los analistas de negocio tengan en cuanto a la problemática de fraude que atraviesan y de un conjunto de reportes predefinidos que ponen de relieve tendencias generales de fraude.

Este proyecto es continuación de **Análisis y Detección de Patrones de Fraude en Medios de Pago** [8] realizado durante el año 2005, también impulsado por PayTrue Solutions. El proyecto realizó un estudio completo de la problemática del fraude en medios de pago, técnicas aplicables y formas de evaluación. En los resultados obtenidos se identificaron tres grandes carencias que se desean atacar mediante un nuevo proyecto de grado:

- Problemas para procesar los volúmenes de datos necesarios.
- Imposibilidad de sugerir variables complejas (variables calculadas aplicando una función de agregación o variables que representan las tendencias en las transacciones cercanas en el tiempo).
- Escasa mejora obtenida al aplicar un clasificador naive bayesian para la predicción del puntaje de riesgo. Este puntaje $\in [0, 1]$ representa el resultado del análsis de las propiedades de la transacción y se utiliza para filtrar las alertas con bajo puntaje de riesgo.

El objetivo de este proyecto es enfocarse en las técnicas aplicables a la resolución del problema de medios de pago, buscando alternativas que permitan superar las carencias mencionadas, y presentar un estudio comparativo de los resultados obtenidos al aplicar estas alternativas.

Capítulo 2

El Negocio de los Medios de Pago y Problemática del Fraude

Un ciclo de negocio se inicia cuando el tarjetahabiente realiza una transacción para obtener el bien o servicio deseado, presentando en el comercio su medio de pago. A continuación el comercio solicita una autorización para la transacción que debe ser aprobada tanto por el adquirente como por el emisor. Luego se efectúa la confirmación de la transacción. Para el intercambio de transacciones electrónicas de tarjetas de crédito se utiliza el éstandar **ISO 8583**.

El fraude es una problemática recurrente en el negocio de los medios de pago que tiene un costo asociado que es asumido por alguno de los actores que involucrados.

En este capítulo se describe el negocio de los medios de pago y la problemática del fraude. En la sección 2.1 se describe la mecánica del negocio incluyendo los actores principales y su interacción. La seccion 2.2 contiene un resumen del **ISO 8583**. La sección 2.3 describe las características y modalidades del fraude en medios de pago.

2.1. Mecánica del Negocio

En el negocio de los medios de pago existen diferentes actores que interactúan cumpliendo con los roles fundamentales para su funcionamiento.

En un ciclo de negocio habitual, un sujeto obtiene un bien o servicio mediante el pago de un valor. Para completar esta transacción intervienen medios físicos e instituciones que regulan y procesan el flujo normal de la transacción. El proceso involucra a un tipo de tarjeta (crédito, débito o prepago) que va a sustituir el pago en efectivo del bien o servicio a adquirir y aparecen instituciones financieras (emisores), como bancos, que son las encargadas de otorgar la tarjeta con su consiguiente límite de crédito y mecanismos de financiación al solicitante (tarjetahabiente o *cardholder*); aparecen otras instituciones (adquirentes) que establecen diferentes acuerdos con variados comercios para que el tarjetahabiente pueda realizar sus transacciones en los comercios; aparecen figu-

ras que interactuarán entre los emisores y los adquirentes, brindándoles servicios de procesamiento (procesadoras) y en el caso de transacciones internacionales aparecen las marcas internacionales como Visa, MasterCard o American Express para completar el ciclo. Mediante la definición de estándares y protocolos, la marca internacional permite la interacción de tarjetahabientes y comercios de distintos países.

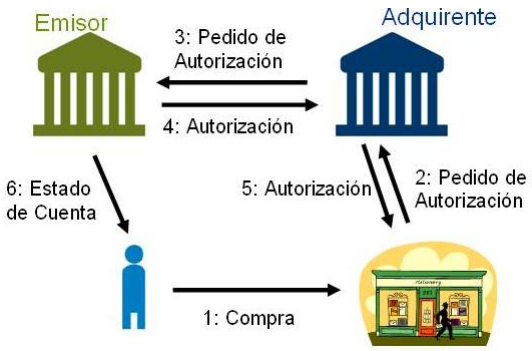

Figura 2.1: Ciclo de Negocio

Un ciclo de negocio depende del soporte físico, de la infraestructura disponible en el comercio y de factores operativos. El ciclo se inicia cuando el tarjetahabiente realiza una transacción para obtener el bien o servicio deseado, presentando en el comercio (*card acceptor*), su medio de pago en lugar de dinero en efectivo o cheques. A continuación el comercio solicita una autorización para la transacción, en forma automática o telefónica. La autorización no tiene efecto financiero y debe ser aprobada tanto por el adquirente como por el emisor.

Para pedir una autorización se requieren varios datos del tarjetahabiente, que incluyen, mínimamente, el número y fecha de vencimiento de su tarjeta. El emisor autoriza la transacción si los datos del tarjetahabiente son correctos y la tarjeta tiene dinero disponible según el monto de la transacción. El adquirente autoriza la transacción si los datos del comercio son correctos, y el comercio está habilitado para realizar la transacción solicitada. Luego se efectúa la confirmación de la transacción, que sí tiene un efecto financiero, idealmente refrendada mediante la firma de una nota (ticket o voucher) por parte del tarjetahabiente.

Es posible que algunas autorizaciones nunca sean confirmadas y no lleguen a tener efecto financiero, lo cual a efectos prácticos equivale a nunca haber hecho la transacción. También es posible que existan confirmaciones sin autorización previa.

A fin de mes el tarjetahabiente recibe un estado de cuenta que le informa cuánto y cuándo pagar por sus transacciones del mes. Usualmente el comercio cobra sus transacciones antes que el tarjetahabiente las pague al emisor, inclusive en las compras en cuotas. El ciclo de pagos, mostrado en la **Figura 2.2**, se puede resumir de la siguiente manera: primero, el adquirente le paga al comercio; luego, el emisor le paga al adquirente; por último, el tarjetahabiente le paga al emisor. Los números reflejan el orden en que se realizan los pagos.

A continuación se describen más en detalle los actores principales:

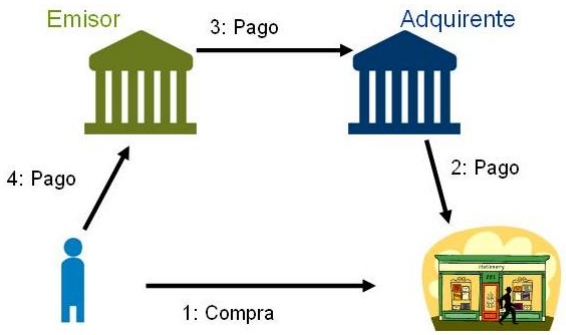

Figura 2.2: Ciclo de Pagos

Tarjetahabiente
Es la persona que mantiene una relación comercial con la institución o banco emisor, poseedor de una tarjeta para el uso en comercios afiliados.

Comercio
Establecimiento comercial afiliado por la entidad adquirente que le permitirá aceptar tarjetas de crédito, débito o prepago como medio de pago de sus clientes.

Adquirente
Un adquirente (*acquirer*), es la institución que mantiene las relaciones comerciales con los comercios, recibe todas sus transacciones, cobrándoles cargos por el servicio, entre los cuales se incluye la comisión por transacción (que varía de país en país y según el acuerdo realizado con el comercio), así como el alquiler de las terminales en los puntos de venta (*Point Of Sale o POS*) en algunos casos.

El adquirente es el encargado de reenviar la transacción a la entidad que corresponda ya sea la entidad emisora de la tarjeta (en caso de que tengan un acuerdo) o a la marca internacional para que lo reenvíe a la entidad emisora correspondiente, quien es la encargada de la autorización de la transacción (ya que es la que posee los datos del tarjetahabiente). El adquirente se encarga de pagar a los comercios a cambio de las transacciones que los tarjetahabientes hayan efectuado. El adquirente está muchas veces capacitado para autorizar cuando el monto de la transacción está por debajo del denominado límite de piso (*floor limit*).

Emisor
El emisor (*issuer*), es un banco, institución financiera u otro, encargado de la emisión del medio de pago y su administración, otorgándole al cliente financiación de sus compras mediante el uso del medio de pago.

El emisor se responsabiliza de las transacciones realizadas por el tarjetahabiente ante los adquirentes. El negocio del emisor se basa en el cobro de cargos por asumir este riesgo y por brindar el servicio. Los cargos más comunes son: costo anual por uso de la tarjeta, costos periódicos por impresión y envío del estado de cuenta, cobro de intereses por financiación cuando el tarjetahabien-

te no paga el total del saldo y cobro de intereses por financiación de compras realizadas en cuotas. Es el encargado de estudiar la capacidad de pago del tarjetahabiente, asignarle un límite de crédito, autorizar las transacciones, controlar el disponible de compra, emitir el estado de cuenta para sus clientes y realizar los pagos correspondientes a los adquirentes, entre otras funciones.

Una misma organización puede actuar como emisor y adquirente simultáneamente. Por ejemplo, en Uruguay opera OCA Card, que es a la vez adquirente y emisor, en oposición con VISANET Uruguay que es únicamente adquirente de Visa para comercios de Uruguay. A su vez, la mayoría de los bancos importantes son emisores de Visa y MasterCard.

Es posible que en un mismo mercado convivan múltiples emisores y adquirentes. También es posible que un mismo comercio esté afiliado a varios adquirentes. Cuando un comercio acepta un cierto medio de pago, significa que alguno de los adquirentes a los que está afiliado el comercio tiene un acuerdo comercial con el emisor de dicha tarjeta o con la marca internacional de dicha tarjeta.

Algunos de los factores que afectan el negocio de los medios de pago y las relaciones entre los actores que intervienen son el tipo de contrato entre el emisor y el tarjetahabiente (producto) y el soporte físico del medio de pago (tarjeta). El producto determina el tipo de transacción que el tarjetahabiente puede realizar e incluye el tipo de tarjeta.

La tarjeta puede ser de crédito, débito y/o prepago. En el caso de las tarjetas de crédito los tarjetahabientes tienen límites con respecto a la cantidad que pueden cargar. Cada mes el tarjetahabiente puede pagar su saldo por completo o pagar una parte. El emisor establece el pago mínimo y determina los cargos de financiamiento para el saldo pendiente. Las tarjetas de crédito también pueden usarse en los cajeros automáticos o en un banco para servirse de un adelanto de efectivo aunque, a diferencia de las tarjetas de débito, se cobra un interés al tarjetahabiente.

La tarjeta de débito es usada para extraer dinero de un cajero automático y también para pagar las compras realizadas en comercios que tengan un terminal lector de tarjetas bancarias. Se diferencia de la tarjeta de crédito en que el dinero que se usa nunca se toma a crédito sino que se debita del que se disponga en la cuenta bancaria asociada a la tarjeta. Algunos bancos realizan acuerdos con sus clientes para permitirles extraer dinero en descubierto, generando un préstamo con sus respectivos intereses. Su cuota anual es más barata que la de crédito o incluso gratis.

Una tarjeta de prepago es aquella en la que se anticipa el importe del consumo que se realizará con la tarjeta. Se efectúa una carga de dinero en la tarjeta y pueden realizarse operaciones hasta consumir el importe cargado. Además, el producto determina la cantidad de cuotas y el interés cobrado y límite de crédito entre otros.

La forma física que toma el medio de pago puede ser una tarjeta plástica de banda magnética, una tarjeta plástica con chip (conocidas como tarjetas EMV [7], un teléfono celular, u otros. Los distintos soportes físicos incluyen distintos mecanismos para salvaguardar la información que contienen, por lo que algunos son más riesgosos que otros respecto al fraude.

2.2. ISO 8583

El **ISO 8583** *(Estándar para Transacciones Financieras con Mensajes originados en una tarjeta - Especificaciones de los mensajes de intercambio)* es el estándar de la *International Organization for Standardization* para sistemas que intercambian transacciones electrónicas realizadas por poseedores de tarjetas de crédito [15].

Una transacción basada en una tarjeta usualmente sale desde un dispositivo de compra, tal como un POS o un cajero automático (ATM), a través de una red (o redes) hacia un sistema del emisor de la tarjeta para obtener una autorización en función de la cuenta del titular de la tarjeta. La transacción contiene información que se obtiene de la tarjeta (ej. número de cuenta), la terminal (ej. nro. de comercio) y la transacción (ej. importe) en conjunto con otra información que puede generarse o agregarse dinámicamente por los sistemas intervinientes. El sistema emisor de la tarjeta puede autorizar o rechazar la transacción y genera un mensaje de respuesta que debe ser devuelto a la terminal en un tiempo breve.

ISO 8583 define un formato de mensaje y un flujo de comunicación para que diferentes sistemas puedan intercambiar estas transacciones. Todas las redes de tarjetas basan sus transacciones en el estándar ISO 8583.

Aunque el ISO 8583 define un estándar común, no se usa normalmente en forma directa y cada red lo adapta para su propio uso con campos adecuados a sus necesidades particulares.

Un mensaje ISO 8583 consta de las siguientes partes:

- **Indicador de Tipo de Mensaje (*Message Type Indicator o MTI)*:** es un campo numérico de 4 dígitos que clasifica la función de alto nivel del mensaje. Un MTI incluye la versión ISO 8583, la clase (*Message Class*), la función (*Message Function*) y el origen del mensaje (*Message Origin*).
- **Uno o más *bitmaps*:** indicando qué elementos están presentes en el mensaje.
- **Elementos de Datos (*Data elements*):** los campos del mensaje.

Los **Elementos de Datos** son los campos individuales que llevan la información sustancial acerca de la transacción. Hay 128 campos definidos en el estándar ISO8583:1987, y 192 en ediciones posteriores. Mientras que cada elemento tiene un significado y formato específico, el estándar también incluye algunos campos de propósito general y algunos campos especiales para sistemas o países, los cuales varían notoriamente en su forma y uso de una implementación a otra.

A continuación se describen los campos relevantes, la totalidad de los campos se definen en A.

Número de cuenta primario (*Primary Account Numbero PAN*): Un número que identifica la cuenta del cliente, o sea, un número de tarjeta de hasta 19 dígitos. A partir del PAN se obtiene el número identificador del emisor/producto.

Monto de la transacción: Contiene el monto de la transacción en la moneda especificada en el campo código de moneda. Es un campo numérico de largo 12 que no incluye punto o coma decimal y los decimales se asumen basados en el campo código de moneda.

Fecha y Hora de la transacción: Contiene la fecha y hora en GMT de la transacción con formato *MMDDhhmmss*.

MCC (*Merchant Category Code*): Contiene un código describiendo el tipo de negocio o rubro del producto o servicio. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [12].

```
3001 American Airlines
3504 Hilton Hotels
5411 Supermercado
5814 Restaurantes de comida rápida.
6010/6011 ATM
```

Código de país del adquirente: Contiene un código que identifica al país del adquirente del comercio o ATM. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [6].

```
032 Argentina
076 Brazil
858 Uruguay
```

Código de país del emisor: Contiene un código que identifica al país del emisor de la tarjeta, usando la misma codificación que para el país adquirente.

Modo de captura de datos del POS (*POS entry mode*): Contiene un código que identifica el método utilizado para capturar el número y fecha de vencimiento de la tarjeta cuando se usa una terminal electrónica y la capacidad de captura de la terminal. A continuación se listan los valores posibles para los dos subcampos.

```
Modo de captura
   00 Desconocido, o no se usó una terminal electrónica
   01 Digitado manualmente
   02 Lectura de la banda magnética de la tarjeta, con la salvedad de que
      es posible que el CVV (código de verificación grabado en la banda)
      no se haya podido leer en forma confiable
   03 Lectura de código de barras
   04 Lectura mediante reconocimiento óptico de caracteres (OCR)
   05 Lectura de circuito integrado, la captura del CVV (en este caso,
      grabado en el chip) es confiable
   90 Lectura de la banda magnética de la tarjeta, el CVV se leyó
      correctamente
   95 Lectura de circuito integrado, el CVV no es confiable
```

```
Capacidad de la terminal
   0 Desconocido
   1 No se utilizó POS
   2 Lectura de banda magnética
   3 Lectura de código de barras
   4 Lectura mediante reconocimiento óptico de caracteres (OCR)
   5 Lectura de circuito integrado
   9 Digitación manual, sin capacidad de lectura electrónica
   N No se proporcionó
```

Código de condición del POS (*POS condition code*): Contiene un código que identifica las condiciones en las que se realizó la transacción. A continuación se listan los valores posibles.

```
00 Transacción normal
01 Cliente no presente
02 Terminal automática operada por el cliente (p.e. ATM)
03 Comercio o tarjeta sospechosos
05 Cliente presente, tarjeta no presente
06 Pedido de preautorización.
08 Compra por teléfono o correo; incluye transacciones recurrentes
   (como cobro automático de facturas de servicios)
10 Se verificó la identidad del cliente
51 Verificación de dirección o cuenta, no se está pidiendo autorización
   para una transacción.
59 Solicitud de comercio electrónico por una red pública, como Internet
   71 Tarjeta presente, pero no se pudo leer la banda; los datos fueron
   digitados manualmente
```

Número identificador del adquirente: Campo numérico de hasta 11 caracteres que identifica la institución financiera que actúa como adquirente.

Código de respuesta: Contiene un código que define la respuesta a un pedido de autorización. Los códigos 00 o 10 indican aprobación y el resto de los códigos indican la razón de rechazo. A continuación se listan a modo de ejemplo algunos valores posibles.

```
01 Referirse al emisor
03 Comercio inválido
15 No existe el emisor
```

Número identificador de la terminal de venta: Contiene un código que identifica la terminal.

Número identificador del comercio: Contiene un código alfanumérico que puede identificar el comercio o una terminal.

Nombre y ubicación del comercio: Campo alfanumérico que indica el nombre, ciudad, estado y país del comercio.

Moneda de la transacción: Contiene el código de la moneda del monto de la transacción.

2.3. Fraude en Medios de Pago

2.3.1. Características

El fraude se define en [3] como actividad no autorizada de una cuenta por una persona para quién la cuenta no fue creada. En la práctica son transacciones que son repudiadas por el tarjetahabiente y cuyo costo debe ser asumido por alguno de los actores que intervienen en el negocio de los medios de pago. Existe una intención de parte de todos los actores involucrados de tomar acciones e incorporar prácticas de administración de riesgo para prevenir y/o disminuir el progreso del fraude.

El fraude puede ser cometido por el tarjetahabiente, un comercio, un empleado de un emisor, un empleado de un adquirente y/o un tercero. Para que se produzca un fraude tiene que haber existido previamente compromiso de la información de la tarjeta, que puede haber sido cometido por el mismo o distinto defraudador.

El fraude en tarjetas de crédito presenta las siguientes características [13, 14, 8]:

- El volumen de transacciones legales y fraudulentas varía dependiendo del contexto y en el tiempo, pero se procesan millones de transacciones por día.
- El fraude es significativamente menor a las ventas legítimas (en el entorno del 0,1 %).
- Se pueden observar múltiples patrones de fraude simultáneamente. Cada patrón puede ser ocasional, regular, estacional, o seguir varios tipos de comportamiento en el tiempo.
- El comportamiento legítimo varía dependiendo del contexto y en el tiempo.
- Luego de que se descubre el modus operandi de los defraudadores profesionales, ellos reaccionan modificando su comportamiento hasta que logran evitar los controles, causando una evolución constante en los patrones de fraude.
- En general una institución se entera de que una transacción fue fraudulenta cuando un tarjetahabiente repudia la transacción, por lo que el reconocimiento de una transacción como fraude demora normalmente entre uno y tres meses.
- No todo el fraude es detectado y no todas las transacciones denunciadas como fraude lo son realmente. Hay transacciones de fraude que son

pagadas por los tarjetahabientes al recibir el estado de cuenta sin darse cuenta.

- Existe una visión parcial de los datos. Sólo aquellas entidades que son a la vez emisor y adquirente conocen todos los datos de las transacciones realizadas. En general, los adquirentes sólo conocen aquellas transacciones realizadas en sus comercios, pero no todas las transacciones de los tarjetahabientes que compraron allí. Simétricamente, los emisores sólo conocen aquellas transacciones realizadas por sus tarjetahabientes, pero no todas las transacciones realizadas en los comercios donde ellos compran. Las marcas internacionales sólo conocen las transacciones que pasan por su red, pero no las transacciones locales. A su vez, los datos conocidos de cada transacción son los intercambiados con el protocolo ISO 8583. Los datos adicionales a los de este protocolo son conocidos únicamente por el emisor o adquirente al que pertenece el tarjetahabiente o comercio respectivamente. Algunos de estos datos pueden ser de interés para la detección de fraude, como por ejemplo, el monto disponible en la cuenta o el límite de crédito o fecha de fundación del comercio.

2.3.2. Modalidades de Fraude

Para poder realizar una transacción es necesario conocer datos de una cuenta válida dentro del sistema, como mínimo el número de tarjeta y fecha de vencimiento. Estos datos se pueden obtener de diferentes maneras.

El tipo de fraude que se puede realizar depende de la información que tenga el defraudador, por ejemplo:

- Si conoce el disponible.
- Si tiene una tarjeta física con la cual ir a un comercio.
- Si sólo tiene el número de tarjeta y fecha de vencimiento (con lo cual a veces basta para hacer una compra por Internet). Este caso, conocido como escenario de tarjeta no presente, es mucho más riesgoso desde el punto de vista del fraude.

Métodos de obtención de información de una cuenta (compromiso de la información).

Existen diversas modalidades que emplean los defraudadores para obtener información de cuentas. A continuación detallamos las principales.

Skimming
Se trata de la copia de la información de la banda magnética de una tarjeta válida para su posterior uso en transacciones fraudulentas. Todos los datos de la pista de la banda magnética de la tarjeta válida se capturan y recodifican en una tarjeta codificada. El término *skimming* también se utiliza para hacer referencia a cualquier situación en la cual se duplican datos de cuentas guardados en un sistema o transmitidos por medios electrónicos, los cuales se recodifican posteriormente en tarjetas falsificadas o se utilizan de otras formas para efectuar transacciones fraudulentas.

Phishing
Se comete mediante el uso de ingeniería social, caracterizado por intentar adquirir información confidencial de forma fraudulenta (como puede ser una contraseña o información detallada sobre tarjetas de crédito u otra información bancaria). El estafador, conocido como phisher, realiza una aparente comunicación oficial utilizando comúnmente un correo electrónico, o algún sistema de mensajería instantánea y convence al tarjetahabiente de que le proporcione la información haciéndose pasar por un actor válido (por ejemplo un emisor o un adquirente).

Intercepción de correo
Cuando el correo que contiene datos de una tarjeta válida es interceptado por un defraudador.

Robo o pérdida de tarjetas
El defraudador obtiene acceso a la tarjeta y debe utilizarla para cometer el fraude antes de que el tarjetahabiente lo detecte y pueda cancelarla. Es el modo de obtención de información más común.

Datos comprometidos por diversas fallas de seguridad
Incluye intrusiones en los sistemas informáticos de la entidad financiera, pérdida de paquetes con tarjetas a entregar en la empresa de transporte, acceso en forma indebida a archivos con datos de tarjetas, etc.

Utilización de la información para cometer fraude.

Existen diversas modalidades que emplean los defraudadores para utilizar la información para cometer fraudes. A continuación detallamos las principales.

Solicitudes de tarjeta con datos falsos
Consiste en llenar una solicitud de tarjeta de crédito con datos falsos u obtenidos previamente. Una vez obtenida la tarjeta, el defraudador la utiliza como si fuera el propietario legítimo.

Cambio de dirección
Solicitud de cambio de dirección haciéndose pasar por un tarjetahabiente, para luego solicitar una reimpresión de la tarjeta o emisión de una tarjeta adicional, que será enviada a la nueva dirección.

Autofraude
Se comete fraude con la propia cuenta del defraudador.

Clonación de tarjetas
Se crea una tarjeta con un número de tarjeta válido. Se puede codificar la información en una tarjeta de banda magnética en blanco o cambiar una robada.

Falsificación manual de tarjetas
Ocurre cuando se crea un número de tarjeta válido usando un software que los genere. Se puede codificar la información en una tarjeta de banda magnética en blanco o cambiar una robada.

Prueba de cuenta
El defraudador pide autorización por montos bajos para probar que una tarjeta robada, clonada o falsa está activa y poder realizar luego compras hasta agotar el disponible.

Fraude con tarjeta no presente (MOTO mail order / telephone order)
Se realizan transacciones con los datos sin tener físicamente la tarjeta.

Comercio Fraudulento
Un comercio aparentemente legítimo que abre una cuenta válida con un Adquirente y después de un breve período de actividad de venta normal deposita un gran número de transacciones fraudulentas por altos montos. Una vez recibido el pago de las transacciones, el comercio vacía su cuenta de depósito bancaria y desaparece. Los comercios fraudulentos con frecuencia presentan solicitudes de afiliación fraudulentas a varios adquirentes al mismo tiempo.

Patrones de riesgo conocidos:

- Compras en comercios de *MCC* de fácil reventa, como ser joyerías, productos electrónicos, artículos de lujo, etc.
- Retiros en efectivo excesivos o inusuales.
- Transacciones de una misma cuenta en cortos períodos de tiempo.
- Pagos adelantados en cheques de terceros o contra otras cuentas, previos al vencimiento.
- Transacción originadas en países considerados como riesgosos.
- Transacciones en el extranjero.
- Transacciones originadas en diversos países en un mismo día.
- Transacciones de alto valor, o de mayor valor al promedio de ventas del comercio o del *MCC*.
- Parámetros de alto riesgo en la transacción como ser escenario de tarjeta no presente, modo de captura de los datos inconsistente con la capacidad de la terminal, indicador de entrega de dinero en efectivo u otros similares.
- Múltiples transacciones de la misma cuenta en el mismo comercio o en comercios del mismo *MCC* (sobre todo en *MCC* que no suelen tener repetición, como restaurantes, hoteles, aerolíneas, etc.).
- Múltiples intentos de obtener autorización en un mismo comercio y con tarjetas distintas pero del mismo emisor o producto.
- Comercios con más devoluciones que compras.
- Varias transacciones rechazadas por la misma razón (número de cuenta no existente, fecha de vencimiento errónea, carencia de disponible) en un mismo comercio.

- Varias solicitudes de autorización con tarjetas de un mismo emisor o producto, caso conocido como *ensayo de BIN* (*Bank Identification Number*).
- Varias transacciones con una misma tarjeta rechazadas por no coincidir el código de verificación.
- Pagos en una cuenta fuera de las fechas normales, o antes de la emisión de la factura correspondiente.
- Múltiples transacciones de una misma tarjeta por montos similares.

2.3.3. Control de Fraude

Aunque se han desarrollado distintos métodos de detección y prevención de fraude, el control de fraude se realiza de formas muy variadas. Algunas entidades financieras (en general muy pequeñas) no dedican ningún esfuerzo a monitorear o prevenir el fraude. Otras han hecho importantes inversiones. A continuación se listan algunas de las formas de detección y control de fraude [1]:

- **Sistema de verificación de direcciones (*Address Verification System o AVS*):** consiste en verificar que la dirección a la que se envía los bienes adquiridos coincida o se encuentre dentro de un rango aceptable respecto a la dirección donde se está enviando el estado de cuenta de la tarjeta.
- **Métodos de verificación de tarjetas (*Card Verification Methods o CVM*):** se trata de la verificación de un número de 3 ó 4 dígitos impreso en la tarjeta, que no forma parte de la información que contiene la banda magnética.
- **Listas positivas y negativas:** información almacenada en bases de datos donde se califican ciertas características como positivas y/o negativas para detectar transacciones de alto riesgo, un ejemplo de esto es la clasificación de los países según el volumen de fraudes cometidos, pero también se aplica a comercios, rangos de IP, etc.
- **Autenticación del cliente:** se basa en el concepto de los cajeros automáticos y el PIN (*Personal Identification Number*), el cual no está impreso ni codificado en la banda magnética de la tarjeta.
- **Biométricas (*Biometrics*):** esta es una técnica que se apoya en una característica única del titular de la tarjeta, como por ejemplo la huella dactilar, de manera de tener la certeza de que quien está en poder de la tarjeta es efectivamente su propietario.
- **Monitoreo de comportamiento transaccional (*Transaction behavior monitoring*):** consite en realizar un seguimiento de las transacciones para detectar comportamientos inusuales y/o riesgosos para predecir transacciones fraudulentas. Existen variadas implementaciones, por ejemplo, sistemas de reglas simples, sistemas de votación, redes neuronales o revisiones manuales.

- **Sistema de reglas simples:** es un sistema de reglas del estilo *If ... Then* para filtrar las transacciones entrantes. Estas reglas son diseñadas por un experto en el dominio. La eficacia del sistema depende del conocimiento del experto y de la buena definición de las reglas.
- **Sistema de votación (*Scoring*):** herramientas basadas en modelos estadísticos diseñados para reconocer transacciones fraudulentas de acuerdo con un número de indicadores derivados de las características de la transacción.
- **Redes Neuronales (*Neural Networks*):** las redes se basan en el reconocimiento de patrones de fraude. Se apoyan en el conocimiento de transacciones históricas válidas y fraudulentas para su entrenamiento (aprendizaje), dando como resultado un sistema que identifica el grado de riesgo o fraude de una transacción en base a los patrones aprendidos.
- **Revisión manual:** Consiste en la revisión manual de las transacciones sospechosas. Este método es extremadamente caro, lento y poco efectivo.

Capítulo 3

Risk Center

El *Risk Center* es un conjunto de soluciones desarrolladas por PayTrue Solutions orientado a detectar, prevenir, analizar y contener el fraude en cualquier sistema de medios de pago. Para ello cuenta con un sistema integral e integrador que se integra e interactúa con otros sistemas existentes [**?**].

Fraud Intelligence
Herramienta para analistas de riesgo, que permite visualizar información histórica de fraude tanto de comercios como de tarjetahabientes, en un solo lugar. Comparte información entre todos los componentes para mejorar la investigación y detección de fraude.

Detection Engine
Solución de monitoreo y generación de alertas para transacciones online, emisor y adquirente.

Advanced Console
Consola de análisis de riesgo, que soporta alertas de múltiples fuentes, emisor y adquirente.

RMW
Solución diseñada para bancos de Latinoamérica y el Caribe, desarrollada para poder utilizar la solución de Visa Internacional Issuer Fraud Detection (IFD). Permite procesar las alertas según los requerimientos de la marca, ofreciendo funcionalidades y beneficios que simplifican las tareas de los usuarios y proveen un mayor control sobre las transacciones realizadas.

CPP Finder
Herramienta de detección e investigación sobre comercios donde se comprometió la información de una tarjeta.

Suspicious Activity
Monitorea las actividades de fraude sospechosas o excesivas de comercios.

Statistical Manager
Es un módulo de reportes estadísticos para el análisis del comportamiento de la

actividad fraudulenta. Permite asistir a los analistas de riesgo en la comprensión de las situaciones de fraudes a partir de una herramienta de reportes que permite navegar sobre los datos.

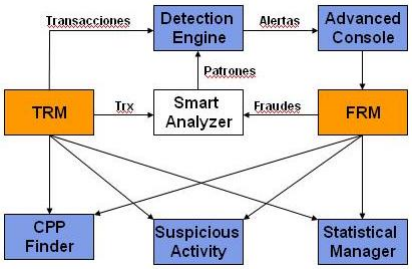

Figura 3.1: Risk Center

Se menciona el objetivo y los componentes del *Risk Center* ya que el producto a desarrollar en nuestro proyecto tiene como propósito formar parte de del mismo, como veremos a continuación.

3.1. Detection Engine

Al Detection Engine llegan todas las autorizaciones en forma inmediata o casi inmediata (online). Las transacciones recibidas son evaluadas y la herramienta decide si se genera o no una alerta asociada a la transacción. La decisión se toma a partir de un conjunto de reglas, un scoring y un umbral. El proceso de decisión es el siguiente:

1. **Evaluación de reglas:** En el Detection Engine se deben configurar un conjunto de reglas de la forma *If* **condiciones** *Then* **fraude**. Si una transacción cumple con al menos una regla pasa a la siguiente etapa y si no cumple con ninguna de las reglas definidas no se generará una alerta para esta transacción.

2. **Cálculo de puntaje de riesgo:** Esta fase, también llamada *Scoring*, consiste en calcular un puntaje de riesgo para aquellas transacciones que pasaron la primera etapa. Para realizarlo en el Detection Engine también se debe configurar un scorer.

3. **Umbral:** Si el puntaje de riesgo supera el umbral definido, se genera una alerta para esa transacción.

Este proceso requiere una configuración (definición de reglas, forma de cálculo del puntaje de riesgo, umbrales) que debe realizarse en forma manual.

3.1.1. Evaluación de Reglas

Esta fase consiste en evaluar un conjunto de reglas sobre cada transacción. Para mostrar una regla nos limitaremos a dar sus condiciones, ya que todas

siguen la forma: *If* **condiciones** *Then* **fraude**.

Por ejemplo, una regla podría ser:

$MCC = 6010 \wedge MONTO \geq 10000$

Si una transacción es clasificada como fraudulenta por alguna regla se realizan las siguientes fases. Si no es clasificada por ninguna regla no se genera una alerta para ella. Las reglas expresan condiciones sobre las propiedades de cada transacción, la historia y otros factores. Separaremos las propiedades en tres niveles según su complejidad.

Nivel 1

En un primer nivel, las propiedades disponibles se corresponden con los datos intercambiados en el protocolo ISO 8583. Además, si la entidad financiera es un emisor o un adquirente, es posible disponer de datos adicionales como nombre del tarjetahabiente, dirección del comercio y otros. Para conocer las propiedades de nivel 1 basta con conocer los datos asociados a una transacción. Las propiedades en los niveles mayores a 1 son calculadas y requieren conocer datos asociados a varias transacciones. Para que las propiedades de los niveles mayores a 1 estén disponibles deben ser indicadas al Detection Engine, que debe calcularlas a medida que le llegan transacciones.

Nivel 2

Las propiedades de nivel 2 son aquellas calculadas aplicando una función de agregación a una propiedad del conjunto de transacciones anteriores que cumplen alguna condición de filtrado. Las condiciones de filtrado más comunes son: misma tarjeta, mismo producto, mismo comercio, o mismo comercio y misma tarjeta, siempre respecto a la transacción para la cual se está calculando la propiedad. De esta forma se puede contar con nuevas propiedades tales como suma de los montos de las transacciones realizadas por la misma tarjeta en el último mes o cantidad de transacciones rechazadas en el mismo día para la misma tarjeta y el mismo comercio. A su vez es posible componer varias funciones de agregación, por ejemplo, para calcular el máximo del gasto mensual de una tarjeta, o máximo de la suma de los montos de las transacciones por mes de la misma tarjeta durante el último año.

Nivel 3

Las propiedades de nivel 3 representan las tendencias en las transacciones cercanas en el tiempo. Para su cálculo se consideran los mismos tipos de filtrado que para las transacciones de nivel 2, pero no se aplican funciones de agregación.

Algunos ejemplos:

- Si las últimas 3 transacciones de la misma tarjeta fueron por montos crecientes
- Si las últimas 3 transacciones de la misma tarjeta se realizaron en el mismo comercio
- Si se realizaron varias transacciones con la misma tarjeta y el mismo comercio con pocos segundos de diferencia entre cada una (velocity check)
- Si en las últimas 20 transacciones de algún comercio aparecen 5 números de tarjeta consecutivos

Para poder configurar este tipo de reglas el Detection Engine permite acceder a cualquier propiedad de alguna de las últimas N transacciones que cumplen alguna de las condiciones de filtrado disponibles para el cálculo de atributos agregados (misma tarjeta, mismo producto, mismo comercio, o mismo comercio y misma tarjeta). Así es posible expresar algunas reglas de nivel 3, pero no todas. En particular, los dos primeros ejemplos de la lista anterior se pueden expresar de esta manera (el primer ejemplo se muestra en el siguiente párrafo), pero los dos últimos ejemplos no. Para saber si las últimas 3 transacciones de la misma tarjeta fueron por montos crecientes, se podría escribir la regla:
$MONTO > TX_1.MONTO \wedge TX_1.MONTO > TX_2.MONTO$
donde TX_N es la N-ésima transacción hacia atrás en el tiempo.

3.1.2. Cálculo del Puntaje de Riesgo y Umbral

Sólo aquellas transacciones clasificadas como fraudulentas por alguna regla se procesan en esta fase. En general sucede que las reglas clasifican muchas transacciones legítimas como fraudulentas. Para afinar más el resultado, el Detection Engine genera alertas sólo para aquellas transacciones cuyo puntaje de riesgo (*risk score*) supere un cierto umbral. Este puntaje $\in [0, 1]$ se calcula a partir de las propiedades de la transacción. La forma concreta de hacer el cálculo puede variar. Se puede calcular como una combinación lineal de las propiedades de la transacción, definir por extensión en forma de matriz N-dimensional (con N igual a la cantidad de propiedades de la transacción), o mediante cualquier forma que se desee, por ejemplo mediante una red neural.

El Detection Engine cuenta con un mecanismo de extensiones para sustituir el componente que realiza este cálculo. El umbral se utiliza para ajustar la proporción de falsas alertas generadas por el Detection Engine. Al incrementar el umbral, se generan menos falsas alertas pero se acepta más fraude, mientras que al decrementarlo, se captura más fraude pero al costo de generar más alertas falsas.

Es importante aclarar que el puntaje de riesgo se puede calcular de forma distinta para cada regla, y que es posible que una transacción sea clasificada como fraudulenta por varias reglas, en cuyo caso se le podrían calcular varios puntajes de riesgo distintos. A partir de los distintos puntajes de riesgo y umbrales, el Detection Engine genera una alerta si al menos una de las predicciones indica que la transacción es fraudulenta.

3.2. Smart Analyzer

El Smart Analyzer es un producto que PayTrue Solutions desea desarrollar para asistir a mantener la efectividad de la configuración del Detection Engine. Será utilizado de forma periódica (por ejemplo semanal o mensualmente) para generar nuevas configuraciones para el Detection Engine. El Smart Analyzer no debe operar en forma *online*, por esto se admite que su ejecución insuma varias horas de ser necesario. Para poder utilizarlo se deberá contar con una base de datos con varios meses de transacciones clasificadas como fraudulentas o legítimas. Usualmente esta base de datos es una copia de la base de datos operativa de la entidad financiera.

Nuestro proyecto tiene como fin desarrollar el núcleo de este producto. Por lo tanto, se estudiarán técnicas de aprendizaje automático para determinar el o los algoritmos más adecuados para deducir reglas y calcular un puntaje de riesgo (*scorer*) de acuerdo al dominio particular del problema.

El Smart Analyzer debe sugerir configuraciones del Detection Engine que el analista de riesgo considerará; aquellas que le resulten interesantes las incluirá en la configuración del Detection Engine. Dentro de la configuración a sugerir se considerará:

- Configuración de reglas mediante técnicas de aprendizaje de reglas y selección de atributos.
- Configuración de scorers mediante técnicas de aprendizaje de scoring y selección de atributos.

Capítulo 4

Técnicas de Aprendizaje de Reglas

El campo de Aprendizaje Automático [9] se centra en responder a la interrogante de cómo construir programas que aprendan automáticamente con la experiencia. La Minería de Datos, por otra parte, se enfoca hacia la extracción de información no trivial que reside implícitamente en los datos [17]. Ambas disciplinas se encuentran muy relacionadas y están basadas en la rama de la informática conocida como *Inteligencia Artificial*.

Los avances tecnológicos permiten almacenar grandes cantidades de datos y en la denominada *era de la información* surge la necesidad de transformar los datos en información y la información en conocimiento. Nuestra meta es obtener conocimiento sobre patrones fraudulentos a partir de un historial de transacciones y permitir que este conocimiento evolucione. En este capítulo, introduciremos los principales algoritmos de aprendizaje de reglas.

4.1. Árboles de Decisión

Dentro de los enfoques posibles para el aprendizaje de reglas, encontramos a los *Árboles de Decisión*. El problema de aprendizaje a partir de un conjunto de instancias independientes nos conduce naturalmente a este tipo de representación.

Como se puede ver en [5], en términos generales los nodos en un árbol involucran una prueba de un atributo en particular: usualmente su comparación con una constante. Sin embargo, algunos árboles comparan más de un atributo utilizando una función de uno o más atributos. Las hojas del árbol suponen una clasificación para todo el conjunto de instancias que alcanzan dicha hoja, o un conjunto de clasificaciones, o una distribución de probabilidad sobre todas las posibles clasificaciones. Los árboles de decisión pueden ser representados también como un conjunto de reglas *if-then-else* para facilitar la interpretación por parte de una persona. Éstos métodos se encuentran entre los más populares de inferencia inductiva y han sido aplicados exitosamente a un amplio rango de tareas de aprendizaje, como ser diagnósticos médicos y asesoramiento para riesgo crediticio en préstamos bancarios.

Los problemas a los cuales podemos aplicar árboles de decisión como método de aprendizaje son aquellos en los cuales las intancias se representan como pares atributo-valor, la función objetivo que se desea aprender tiene valores discretos de salida y el conjunto de entrenamiento puede presentar errores así como valores ausentes dado que el algoritmo soporta ruido en los datos.

Los árboles de decisión clasifican instancias desde la raíz del árbol bajando hacia las hojas en donde se encuentran las clasificaciones. Como se explicó anteriormente, en cada nodo se realiza una prueba sobre un atributo y cada rama que desciende desde ese nodo representa un valor posible para el atributo en el caso de que su valor sea discreto. Para atributos numéricos, la prueba podría consistir en determinar si el atributo es mayor o menor que una constante determinada dando como resultado dos ramas posibles. En el caso de atributos no presentes en la instancia, podemos tratar la ausencia como otro posible valor y tener otra rama para ese caso o simplemente basarnos en la rama más popular y tomar ese camino.

En general, un árbol de decisión representa una disyunción de conjunciones de restricciones sobre los atributos. Cada camino desde la raíz hacia las hojas corresponde con una conjunción de pruebas sobre atributos y el árbol en sí a una disyunción de estas conjunciones.

La construcción de un árbol de decisión puede ser expresada de forma recursiva. Se selecciona el atributo a colocar en la raíz y se dividen las ramas según los posibles valores que éste pueda tomar. Así, se divide el conjunto de ejemplos en subconjuntos, uno por cada valor posible del atributo. El proceso se repite recursivamente para cada rama, utilizando las instancias que las alcanzan. En el momento en que todas las instancias en un nodo tienen la misma clasificación, se finaliza el proceso. Lo que resta por definir es cómo determinar el atributo por el cual particionar dado un conjunto de ejemplos con diferentes clases.

Si contáramos con una medida de la *pureza* del nodo, podríamos elegir entonces el atributo que produjera los hijos más puros. La medida de pureza que consideramos se denomina *información* y se mide en unidades llamadas *bits*. Asociada al nodo de un árbol, la pureza representa la cantidad de información que sería necesaria para clasificar una instancia. La cantidad esperada de información son fracciones de bits, generalmente menores a 1. Se calcula la ganancia de información para cada atributo, eligiendo aquel que gane mayor información para el nodo y luego se vuelve a calcular para los atributos restantes en cada rama. El proceso termina cuando todas las hojas son puras, es decir, cuando contienen instancias con igual clasificación. Describiremos a continuación cómo se calcula la información mencionada anteriormente. Es necesario que la solución cuente con ciertas propiedades:

1. Si la cantidad de clasificaciones es cero, entonces la información es cero.

2. Si la cantidad de clasificaciones para cada clase es la misma, la información es máxima.

3. Las decisiones pueden ser realizadas en una sola etapa o en varias etapas, y la información involucrada es la misma en ambos casos.

Una función que satisface estas propiedades es conocida como *entropía*.

$$entropía(p_1, .., p_n) = -p_1 \log p_1 - ... - p_n \log p_n$$

Los argumentos $p_1, .., p_n$ de la entropía se expresan en fracciones que suman 1.

$$info[u, v, w] = entropía\left(\frac{u}{u+v+w}, \frac{v}{u+v+w}, \frac{w}{u+v+w}\right)$$

La razón de los signos negativos es que los logaritmos de las fracciones $p_1, .., p_n$ son negativos y así logramos tener una entropía positiva. Los logaritmos se expresan generalmente en base 2 y la unidad de entropía en bits.

La decisión en múltiples etapas (cuando existen más de 2 clases posibles) puede escribirse como:

$$entropía(p, q, r) = -entropía(p, q + r) + (q + r) * entropía\left(\frac{q}{q+r}, \frac{r}{q+r}\right)$$

donde p+q+r=1.

Los árboles de decisión fueron desarrollados hace ya varios años y el método que utiliza la ganancia de información en los nodos es conocido como ID3. El uso de la tasa de ganancia fue una mejora que se le hizo posteriormente al ID3. Es una solución práctica y robusta pero sacrifica parte de la elegancia y motivación teórica del criterio de ganancia de información. Se realizaron mejoras posteriores, en el sistema C4.5 y el posterior C5.0, que incluyen el manejo de atributos numéricos, valores ausentes y ruido en los datos así como generación de reglas a partir del árbol. [9]

4.2. Algoritmos de Cobertura

Un enfoque alernativo para obtener reglas es tomar cada clase, buscar una forma de cubrir todas las instancias que pertenecen a esa clase y excluir aquellas instancias que no lo hagan. Este mecanismo se llama *cobertura*, porque se trata de identificar una regla que cubra a las instancias. Por naturaleza, el enfoque de *cobertura* lleva a crear un conjunto de reglas y no un árbol de decisión. Los algoritmos de cobertura operan agregando pruebas a la regla que se encuentra bajo construcción, intentando crear una regla con precisión 100 %. En contraste, los algoritmos *divide y vencerás* como los árboles de decisión agregan pruebas intentando maximizar la separación entre clases. [5]

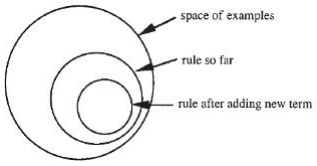

Figura 4.1: Espacio de Instancias

Consideramos un espacio de instancias separado en tres círculos como en la figura anterior, donde en el exterior encontramos el espacio de ejemplos, en el medio las reglas hasta el momento y en el centro las reglas luego de agregar un nuevo término. El nuevo término restringe la cobertura de la regla: la idea es incluir tantas instancias como sea posible de la clase deseada y excluir tantas

```
Aprendizaje por Cobertura(D, Atributo-objetivo, v)
  Hacer Reglas aprendidas igual a vacío
  Hacer E igual a D
  Mientras E contenga ejemplos cuyo valor de Atributo-objetivo es v, hacer:
    Crear una regla R sin condiciones y conclusión Atributo-objetivo=v
    Mientras que haya en E ejemplos cubiertos por R incorrectamente
    o no queden atributos que usar, hacer:
      Elegir la MEJOR condición A=w para añadir a R, donde A es
      un atributo que no aparece en R y w es un valor de los
      posibles que puede tomar A
      Actualizar R añadiendo la condición A=w a R
    Incluir R en Reglas-aprendidas
    Actualizar E quitando los ejemplos cubiertos por R
  Devolver Reglas-Aprendidas
```

Cuadro 4.1: Algoritmo PRISM

instancias de las otras clases como sea posible. Supongamos que la nueva regla cubre un total de t instancias, de las cuales p son positivas y t-p están en otras clases. Luego debemos elegir el nuevo término para maximizar la tasa p/t. Este algoritmo se denomina PRISM y continúa agregando cláusulas a la regla hasta que la precisión es perfecta.

El algoritmo 4.1 se compone de dos bucles anidados. El bucle externo busca la obtención de reglas para el valor del atributo objetivo pasado v. El bucle interno construye la conjunción de pares (atributo, valor) que contengan ejemplos con dicho valor de objetivo y así crear la regla. Como en una pasada del bucle interno pueden quedar ejemplos sin cubrir, deben crearse nuevas reglas para el par atributo-objetivo = valor. La ausencia de una condición se suele representar con el símbolo *?*.

Si consideramos las reglas producidas para determinada clase parecería que debieran ser interpretadas en orden, como una lista de decisión, probando por turnos hasta que una regla sea adecuada y luego utilizar ésa. Esto ocurre porque las instancias que cubre la nueva regla son removidas del conjunto de instancias cuando se completa la regla. Por lo tanto, las reglas siguientes son creadas para instancias que no cubre la regla anterior. Aunque parezca que debemos verificar las reglas en orden, no es necesario. O bien encontramos una regla que cubra a la instancia, donde la clase correspondiente se predice, o no se encuentra la regla y la clase no se predice. El hecho de que las reglas puedan ser ejecutadas en cualquier orden da modularidad pero tiene como desventaja que no está claro qué se debe hacer cuando hay reglas conflictivas que aplican para la misma instancia. En estos casos se fuerza una clasificación tomando la categoría con más ejemplos de entrenamiento.

Las reglas aprendidas por un algoritmo de cobertura se ajustan perfectamente al conjunto de entrenamiento, por lo que existe un gran peligro de *sobreajuste* a los datos.

4.3. Programación Lógica Inductiva

La Programación Lógica Inductiva, o ILP (*Inductive Logic Programming*), permite combinar resultados experimentales y métodos inductivos de aprendizaje automático con la capacidad de representación y formalismo de la lógica de primer orden de forma de inducir conceptos representados por programas lógicos.

Esta rama del aprendizaje automático induce reglas en forma de *cláusulas de Horn* (como máximo tienen un literal positivo, de la forma H $\leftarrow B_1, .., B_n$) las cuales pueden ser interpretadas como programas lógicos en PROLOG.

Además de obtener una representación más compacta y de tener la capacidad de relacionar propiedades de más de un objeto, una de las ventajas de ILP es que permite incluir conocimiento al dominio dentro del proceso de aprendizaje (*background knowledge*). Dependiendo de la complejidad de nuestro dominio y del conocimiento que tengamos sobre él, es posible explotar esta ventaja en mayor o menor medida.

En ILP, como en cualquier aprendizaje inductivo, buscamos aprender una hipótesis que cubra los ejemplos positivos y no cubra aquellos negativos. El proceso de inducción puede verse como un proceso de búsqueda de una hipótesis dentro del espacio de hipótesis $H = H_1, ..., H_n$. El espacio de hipótesis incluye toda hipótesis posible que el algoritmo pueda producir según su diseño y por lo tanto puede ser demasiado extenso. Por este motivo, generalmente se diseñan estrategias de búsqueda que consideran un número limitado de alternativas. [10]

Las entradas de un sistema ILP son:

- Un conjunto de ejemplos positivos E^+
- Un conjunto de ejemplos negativos E^-
- Un programa lógico consistente, T, tal que $T \nvdash e^+$ para al menos un $e^+ \in E^+$ que provee información sobre el dominio.

El objetivo es inducir un predicado H que define una relación en el dominio que se utiliza para clasificar ejemplos nunca vistos. Este predicado puede ser visto como una definición de un concepto, una aproximación de una función booleana cuyo valor es verdadero para todos los objetos pertenecientes a este concepto.

Más formalmente, el objetivo anteriormente descripto consiste en encontrar un programa lógico H tal que H y T sea completo y consistente: $T \cup H \vdash E^+ \wedge T \cup H \nvdash E^-$

Para realizar una búsqueda de hipótesis de forma eficiente es necesario estructurar el espacio de hipótesis. Para esto se utiliza un modelo de generalización, denominado *subsumption*. Una cláusula C subsume (o es una generalización de) una cláusula D si existe una sustitución θ tal que $C\theta \subseteq D$. La notación es $C \preceq D$.

La búsqueda puede hacerse:

- De hipótesis específica a más general, buscando cláusulas que subsuman a la hipótesis actual.
- De hipótesis general a más específica, buscando cláusulas subsumidas por la hipótesis actual.
- En ambos sentidos.

4.4. Algoritmos Evolutivos

Engloban métodos de optimización y búsqueda de soluciones basados principalmente en la evolución biológica. Se combina un conjunto de entidades (candidatos a solución) obteniendo nuevas entidades que se comparan entre sí (compiten), prevaleciendo los más aptos a lo largo del tiempo. De esta forma se va evolucionando a mejores soluciones. Se utilizan principalmente en espacios de búsqueda no lineales y extensos, donde otros métodos no pueden encontrar soluciones en un tiempo razonable. A continuación detallaremos algunos de los principales conceptos de Algoritmos Evolutivos.

Individuos o Cromosomas: Son las entidades que representan las soluciones al problema.
Población: Conjunto de Individuos.
Operadores genéticos: Son utilizados para modificar los individuos, hay de varios tipos:

- **Cruzamiento:** Mezcla de información de dos o más individuos.
- **Mutación:** Intercambio aleatorio entre los individuos.
- **Selección:** Elección de los individuos que conformarán la siguiente generación (sobrevivientes). En esta etapa se seleccionan los individuos cuyas soluciones son más adecuadas al problema, de forma de ir mejorando gradualmente (evolucionando).

Los algoritmos evolutivos engloban tres paradigmas principales, que fueron originados independientemente y con distintas motivaciones. Actualmente los algoritmos tienden a combinar características de estos tres, y a incluir mecanismos de otros campos de estudio, incorporando de estos, por ejemplo, algoritmos de búsqueda y estructuras de datos. A continuación detallaremos los paradigmas englobados.

Programación Evolutiva: Es una evolución de algoritmos genéticos donde lo que cambia es la representación de los individuos. Los individuos son ternas cuyos valores representan estados de un autómata finito.

Las ternas están conformadas por el valor del estado actual, un símbolo del alfabeto utilizado y el valor del nuevo estado. Los valores se utilizan como en un autómata finito, teniendo en cuenta el valor del estado actual y moviéndose a un nuevo estado en función del símbolo actual.

Estrategias Evolutivas: Métodos computacionales que trabajan con una población de individuos pertenecientes al dominio de los números reales, donde se busca evolucionar (utilizando mutación y cruzamiento) hasta lograr un óptimo en una función objetivo.

Algoritmos Genéticos: Método de búsqueda dirigida basada en probabilidad. Cumpliendo la condición de elitismo (o sea que guarde siempre el mejor elemento de la población) se puede demostrar que el algoritmo converge al óptimo. Esto implica que partiendo de una misma población inicial, cuanto mayor sea el número de iteraciones más cerca del óptimo se podrá estar.

Cada solución se codifica en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados *genes*, y cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se los

conoce como *genotipos*. En cada evolución (o iteración) se obtiene una nueva generación de cromosomas que son evaluados utilizando una medida de aptitud. Las siguientes generaciones (descendencia) se forman utilizando los operadores genéticos (cruzamiento y mutación). [9, 16]

Una gran ventaja de estos métodos está en funcionar sobre espacios de búsqueda no lineales y extensos, donde es difícil medir y modelar el impacto de cada parte (por ejemplo atributo) en la calidad de la solución global. También es interesante la capacidad que tienen estos algoritmos de poder ser paralelizados, permitiendo distribuir el costo computacional entre varias estaciones, además de permitir ser ejecutados ininterrumpidamente, de forma de ir obteniendo una mejor solución a lo largo del tiempo. Es de destacar además el potencial de ser utilizados para mejorar la salida de otro método distinto (utilizando la salida del anterior como población inicial de éste).

También existen desventajas importantes. Por ejemplo si la función a optimizar tiene muchos máximos y/o mínimos locales, se requieren muchas más iteraciones del algoritmo para asegurar un máximo/mínimo global. Además, si la función a optimizar contiene muchos óptimos (o valores muy cercanos al mismo), solamente se puede asegurar encontrar uno de ellos (no necesariamente el óptimo).

En situaciones de aprendizaje de reglas se agrega la dificultad de representar las reglas (por ejemplo en una tira de dígitos binarios). Esta dificultad es clara en atributos que utilizan datos de tipo fecha.

Además, podemos observar que estos algoritmos normalmente convergen a un óptimo y no a varios, por lo que convergen a una regla y no a varias; y aunque en la tira se lograra representar operadores más complejos (And y Or), manejar reglas complejas implica el uso de tiras mucho más largas lo que aumenta la cantidad de iteraciones y disminuye el tiempo de convergencia del algoritmo hasta un punto impráctico.

Capítulo 5

Técnicas de Aprendizaje de *Scoring*

Uno de los artefactos que debemos producir en este proyecto, es un *Scorer* para calcular un puntaje de riesgo en cuanto a fraude dada una transacción. Con el puntaje obtenido y un umbal se determinarán acciones a tomar según corresponda en cada caso. A continuación, describiremos algunos algoritmos de aprendizaje de *scoring*.

5.1. *Random Forests*

Random Forests es un clasificador basado en un conjunto de clasificadores de árboles, tal que cada árbol depende los valores de un vector θ_k con valores tomados al azar y con la misma distribución para todos los árboles del bosque (*forest*). Cada árbol es creado a partir del vector y del conjunto de entrenamiento dando como resultado un clasificador $h(x, \theta_k)$, siendo x un vector de entrada. Cada uno de los árboles generan un voto para la clase más popular respecto a la entrada x. El bosque elige la clasificación que tiene más votos tomando en cuenta a todos los árboles. [2]

Cada árbol se construye de la siguiente manera:

1. Si el número de clases en el conjunto de entrenamiento es N, se toma una muestra con reposición de N al azar del conjunto de datos original. Esta muestra será el conjunto de entrenamiento para el árbol en construcción.

2. Si hay M variables de entrada, se toma un número $m \ll M$ tal que en cada nodo, se eligen m variables al azar de las M y la mejor partición dada por estas m se usa para particionar el nodo. El valor de m queda constante durante la construcción del bosque.

3. Cada árbol se construye hasta la máxima profundidad posible.

No se realiza poda. El error total del bosque depende básicamente de los siguientes factores:

- La correlación (grado de interrelación) entre cualquier par de árboles del bosque. Al incrementarse la correlación, aumenta la tasa de error.

- La *fuerza* de cada árbol individual en el bosque se mide por medio de una función particular.

Un árbol con menor tasa de error es un clasificador más fuerte. Al aumentar la fuerza del clasificador individual, decrementa la tasa de error del bosque.

Luego de construir cada árbol, todo el conjunto de datos se procesa a través de él y para cada par de casos, se calcula la proximidad (*proximity*) entre ellos. Si dos casos ocupan el mismo nodo terminal, su proximidad se incrementa en 1. Al final de la corrida, se normalizan las distancias dividiendo por el número total de árboles. El concepto de proximidad se utiliza para reemplazar datos faltantes, localizar desviaciones y producir vistas de los datos.

Brieman utiliza árboles de decisión sin poda como base de clasificadores e introduce azar adicional a los árboles. En cada nodo interior de cada árbol, un subconjunto de r atributos se elige al azar y se evalúa con la heurística del índice de Gini. El atributo con mayor índice de Gini se elige para particionar en ese nodo.

Dentro de las características más importantes a destacar en *Random Forests* encontramos que:

- Corre eficientemente en grandes bases de datos.
- Funciona correctamente con miles de variables de entrada sin tener que utilizar eliminación de variables.
- Estima la importancia de las variables en la clasificación.
- Genera un estimado interno no sesgado de la generalización del error a medida que se contruye el bosque.
- Posee un método efectivo para estimar los datos faltantes y mantiene la precisión cuando faltan grandes proporciones de datos.
- Cuenta con métodos para balancear el error en conjuntos de datos desbalanceados en cuanto a las clases.
- Los bosques generados pueden guardarse para ser utilizados en el futuro con otros datos.
- Provee información sobre la relación entre las variables y la clasificación.
- Computa la distancia entre pares de casos que puede ser usada tanto en *clustering*, como para localizar desviaciones o dar una vista interesante de los datos.
- Ofrece un método experimental para detectar interacción entre variables.

Es un método efectivo computacionalmente y ofrece un buen rendimiento de predicción. Basándonos en la *Ley de los Grandes Números*, podemos demostrar que este clasificador no sufre de sobreajuste a medida que más árboles son agregados y es menos sensible a ruido en los datos que *Boosting*.

Se han realizado estudios con la intención de mejorar la *fuerza* de los árboles individuales o disminuir la correlación entre árboles en el bosque. Utilizando distintas formas de evaluación de atributos en lugar de solo una pueden obtenerse mejores resultados. Por otra parte, reemplazar el voto ordinario por un

voto ponderado de acuerdo a un margen alcanzado en la mayoría de instancias similares genera una mejora estadísticamente significativa en varios conjuntos de datos. Para conjuntos de datos extensos, se puede ganar precisión combinando *Random Features* con *Boosting*.

5.2. Red Neuronal Artificial

Una red neuronal artificial (a partir de ahora las llamaremos RNA) es un paradigma de aprendizaje automático inspirado en el funcionamiento del sistema nervioso animal, mediante la interconexión de neuronas (nodos) en una red que colabora produciendo un estímulo de salida.

Generalmente una RNA es un sistema adaptativo que cambia su estructura basado en la información que fluye por la red durante la etapa de aprendizaje.

Cada neurona es un elemento simple de procesamiento, las cuales en red pueden lograr un comportamiento global complejo, determinado por las conexiones y los parámetros recibidos. Es interesante observar que el procesamiento es realizado colectivamente y en paralelo mediante las neuronas (unidades de procesamiento).

Cada Neurona recibe una serie de entrada a través de interconexiones y emite una salida que viene dada por tres funciones;

1. Una función de propagación, que generalmente consiste en la sumatoria de cada entrada multiplicada por el peso de su interconexión. Si el peso es positivo, la conexión se llama exitatoria, si es negativo, se llama inhibitoria.

2. Una función de activación que modifica de alguna forma la salida de la función de propagación (esta función puede no existir).

3. A la función de activación se le aplica una función de transferencia, que acota la salida de la neurona y generalmente viene dada por la interpretación que se le quiera dar a la salida. Dentro de las más utilizadas están la función sigmoide (que permite obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]).

Las RNA pueden ser utilizadas para modelar relaciones complejas entre entradas y salidas de información o para encontrar patrones en datos. Las redes no tienen porque ser necesariamente adaptativas, pero en su uso práctico existen algoritmos designados para alterar los pesos de las conexiones de la red alterando el flujo de información. [9, 19]

Algunas de las principales propiedades de las RNA son las siguientes;

Auto Organización: Crea su propia representación de la información en su interior.

Tolerancia a fallos: Dado que almacena la información de forma redundante, si ocurre un error parcial, la RNA puede seguir respondiendo aceptablemente.

Flexibilidad: Puede manejar cambios no importantes en la información de entrada, como señales con ruido.

Paralelismo: Permite una implementación de procesamiento en paralelo, distribuyendo el costo computacional.

5.2.1. Clasificaciones

Existen muchos modelos e implementaciones de redes neuronales que pueden ser clasificados de diferentes formas.

Clasificación en función al patrón de conexiones

Propagación hacia adelante: Todas las señales van desde la capa de entrada hacia la salida sin existir ciclos (acíclicas), ni interconexiones entre neuronas de la misma capa (niveles entre interconexiones de neuronas). Las redes pueden contar con una capa (Monocapa) o con varias capas (Multicapa).

Redes recurrentes: Presentan al menos un ciclo cerrado de activación neuronal (por lo tanto hay neuronas que son invocadas más de una vez en un mismo procesamiento).

Clasificación por tipo de aprendizaje

Aprendizaje supervisado: Necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce.

Aprendizaje no supervisado (auto organizado): No necesitan de un conjunto de datos previo.

Híbridos: Enfoque mixto que utiliza una función de mejora que busca facilitar la convergencia hacia la solución.

Clasificación por tipo de información:

Analógicas: Procesan datos de entrada con valores continuos y, habitualmente, acotados.

Discretas: Procesan datos de entrada de naturaleza discreta (habitualmente valores booleanos).

5.2.2. Características de aplicación:

RNA aplica de mejor forma en problemas que cumplen con las siguientes características.

- Aquellos problemas donde las instancias están representados por varios pares valor - atributo.
- La función objetivo puede ser discreta, real o un vector con varios atributos reales y/o discretos.
- Los ejemplos de entrenamiento pueden contener errores.
- Son aceptables altos tiempos de entrenamiento. Generalmente requieren tiempos mayores que algoritmos de árboles de decisión. Los tiempos varían según factores como en número de pesos en la red, el número de ejemplos a considerar en la etapa de entrenamiento y la complejidad intrínseca del algoritmo.
- Puede ser un requerimiento que la función objetivo aprendida pueda ser evaluada rápidamente.

- No es importante que la función objetivo aprendida sea comprensible por humanos.

5.3. Naive Bayes

Un clasificador *Naive Bayes* utiliza una aproximación probabilística para asignar la clase más probable para cada instancia particular. El término *Naive* surge del hecho de que el clasificador asume que el efecto de un atributo en una clase particular es independiente de los demás atributos. Esto es una restricción muy fuerte ya que la independencia de atributos no se cumple en la mayoría de los casos, sin embargo, en la práctica se ha obtenido buenos resultados asumiendo independencia entre los atributos.

El método implica una etapa de aprendizaje en donde se estiman las probabilidades basándose en las frecuencias existentes en el conjunto de entrenamiento. Luego se clasifica un nuevo ejemplo con el valor más probable dado el valor de sus atributos.

$$V = argmax_{v_j \in V}(P(v_j \,|a_1, ...a_n\,))$$

Usando Bayes:

$$V = argmax_{v_j \in V}\left(\frac{P(a_1, ...a_n \,|v_j\,)}{P(a_1, ...a_n)}\right)$$

$$V = argmax_{v_j \in V}\left(P(a_1, ...a_n \,|v_j\,)P(v_j)\right)$$

$P(v_j)$ se puede estimar con la frecuencia en las clases, pero para $P(a_1, ...a_n \,|v_j\,)$ el clasificador asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase. O sea:

$$P(a_1, ...a_n \,|v_j\,) = \prod_i P(a_i \,|v_j\,)$$

Por lo que

$$V = argmax_{v_j \in V}(P(v_j) \prod_i P(a_i \,|v_j)\,)$$

Los valores $P(a_i \,|v_j)$ se estiman con la frecuencia de los datos de entrenamiento.

El clasificador se puede utilizar en problemas en donde cada instancia X se describe como una conjunción de atributo - valor y la función objetivo puede tomar valores de un conjunto finito. La probabilidad calculada de pertenecer a la clase objetivo para cada instancia se puede tomar como una medida de scoring. [9]

Capítulo 6

Técnicas de Selección de Atributos

La Selección de Atributos (*Feature Selection*) es una técnica utilizada comunmente en aprendizaje automático para seleccionar un subconjunto de atributos relevantes con el propósito de obtener un modelo de datos más robusto. Su objetivo es mejorar el rendimiento del modelo de aprendizaje, removiendo aquellos atributos o variables irrelevantes o redundantes. Las principales ventajas ofrecidas son:

- Disminuir la dimensionalidad de los datos.
- Mejorar la capacidad de generalización (al eliminar atributos específicos irrelevantes).
- Aumentar la velocidad del proceso de aprendizaje.
- Simplificar el modelo, haciéndolo más sencillo de interpretar.

Esta técnica tiene el potencial de señalar los principales atributos y cómo están relacionados unos con otros, ayudando a obtener una mayor comprensión acerca de los datos y sus interdependencias.

Desde un punto de vista teórico, puede observarse que alcanzar una selección óptima requiere de una búsqueda exhaustiva de todos los posibles subconjuntos de atributos para una cardinalidad dada, lo cual resulta impráctico para grandes cantidades de atributos. Es por esto que en la práctica, los diferentes algoritmos por lo general buscan un conjunto que sea satisfactorio, en lugar del óptimo.

Los algoritmos de selección de atributos caen típicamente en dos categorías: clasificación jeráquica de atributos (*Feature Ranking*) y selección del subconjunto (*Subset Selection*). La primera categoriza los atributos utilizando una métrica determinada y elimina todos aquellos que no llegan a determinado umbral, mientras que la segunda busca el conjunto de atributos que más se aproxime a un óptimo.

Existen algoritmos que ya tienen incorporados en su funcionamiento formas de selección de atributos, como *Random Forests* y Árboles de Decisión. [5, 18]

6.1. Clasificación Jerárquica de Atributos

Un enfoque es el de construir un modelo lineal y clasificar jerárquicamente los atributos basados en el peso de sus coeficientes. Una mejora de lo anterior es construir un modelo, sacar el atributo de mayor jerarquía, y repetir el proceso (generando cada vez un nuevo modelo) hasta que todos los atributos fueron clasificados. De esta forma se llega a un método de eliminación de atributos recursivo, lo cual tiene un mejor resultado para determinados conjuntos de datos que una clasificación jerárquica basada en un único modelo. Se debe observar que estos métodos retornan simplemente una jerarquía, por lo que hay que utilizar otro método que seleccione el número de atributos a utilizar.

Otra forma de lograr esto es combinando distintos métodos de aprendizaje automático. Una forma es aplicar un algoritmo de árbol de decisión y utilizar únicamente los atributos que son utilizados en el árbol como entrada de otro algoritmo. Es claro que si el segundo algoritmo es nuevamente un árbol de decisión puede que no tenga un gran efecto, pero si se utiliza como entrada de un algoritmo distinto, puede tener un efecto beneficioso. Por ejemplo el algoritmo del vecino más cercano es muy susceptible a atributos irrelevantes, por lo que su rendimiento generalmente mejora notoriamente si se utilizan únicamente atributos relevantes. De esta forma la combinación de ambos métodos puede comportarse mejor que cualquiera de los dos en forma individual.

En cada etapa de un método recursivo o en la combinación de distintos métodos debe asegurarse la utilización de una misma escala para la jerarquía, dado que de otra forma no hay punto de comparación.

Por último, podemos utilizar algoritmos basados en instancias. De esta manera, se utilizan instancias del conjunto de entrenamiento tomadas al azar y se estudian las clases a las cuales pertenecen, marcando las similitudes y diferencias con las clases más cercanas. Si ocurre que una instancia aplica casi completamente en una determinada clase, el atributo diferencial aparenta ser irrelevante y su peso disminuye. En cambio si tenemos una instancia que no cae en una clase determinada por muy poco margen el atributo diferencial se considera relevante (dado que por el la instancia no pertenece a la clase) y su peso aumenta. Repitiendo este proceso una cantidad significativa de veces, se seleccionan atributos con un peso positivo.

La ventaja de este enfoque radica en que el número de atributos a utilizar viene dado naturalmente, ya que se eligen los pesos positivos. Una desventaja menor es que dependiendo del orden en que se vayan obteniendo los ejemplos, puede llegar a cambiar el resultado. Una mayor, es que dos atributos que sean exactamente iguales (redundantes) serán tratados igual, es decir, ambos serán rechazados o aceptados. Para evitar esto, se puede además medir la intercorrelación de los atributos utilizando el grado de simetría (*symmetric uncertainty*), buscando eliminar atributos redundantes.

6.2. Selección del Subconjunto

Estos métodos consisten en su mayoría en buscar un subconjunto en el espacio de atributos que pueda representar a las clases que lo componen de la mejor forma posible. Es claro que a medida que crece el número de atributos, crece exponencialmente el número de subconjuntos, haciendo que una búsqueda

exhaustiva sea algo impráctico.

Normalmente dentro del espacio se busca en una cierta dirección, ya sea de arriba hacia abajo o de abajo hacia arriba. De esta forma, se considera siempre un subconjunto de atributos (empezando con el conjunto vacío en el primer enfoque o con el espacio completo en el segundo) y en cada etapa se modifica agregando (primer caso) o eliminando (segundo caso) un único atributo. Cada subconjunto generado en cada etapa es evaluado mediante un criterio (por ejemplo *cross-validation*) produciendo una medida numérica del rendimiento esperado del subconjunto. Si ningún atributo mejora el subconjunto en una etapa, la búsqueda finaliza y se selecciona el subconjunto de mayor medida. Ambos métodos permiten encontrar un óptimo local, pero no necesariamente uno global. Para mantener la cardinalidad del subconjunto lo más pequeña posible se puede modificar la evaluación de continuidad para que un nuevo atributo sea agregado solamente si incrementa la medida y además cumple un determinado mínimo. Los métodos pueden ser combinados para lograr una búsqueda bidireccional.

La búsqueda *Best-First* termina con un subconjunto cuando el rendimiento comienza a decaer, pero además maneja una lista de todos los conjuntos revisados previamente, ordenados por rendimiento, de forma que puedan explorarse subconjuntos previos. *Beam Search* es similar al anterior, pero limita la lista de subconjuntos a un número determinado, de forma que maneja los N candidatos más promisorios. También pueden utilizarse algoritmos genéticos que hagan *evolucionar* los subconjuntos usando perturbaciones al azar.

6.3. Rendimiento y Optimización

El proceso de selección de atributos es intensivo desde el punto de vista computacional. Con X atributos, el orden puede ir de X^2 (para selección hacia atrás o hacia adelante) hasta 2^X (para algoritmos exhaustivos, multiplicado por 10 si se utiliza *10-fold cross-validation*). Por este motivo optimizar estos algoritmos es de gran importancia.

Una forma de acelerar el proceso es terminar de evaluar un subconjunto de atributos tan pronto se pueda deducir que no va a superar a otro. A raíz de esta necesidad, surgen metodologías como *Race Search*, en las cuales los atributos simples compiten en un subconjunto y se eliminan aquellos que no cumplen determinado rendimiento. El mecanismo de medida generalmente se realiza utilizando probabilidades.

Schemata Search es un método diseñado específicamente para hacer competir atributos, determinando en cada iteración si un atributo debe ser o no incluido, incluyendo otros atributos de forma aleatoria, y utilizando al ganador de la competencia anterior para la próxima iteración. Estos algoritmos pueden ser combinados con una clasificación jerárquica previa de los atributos, de forma de solo hacer competir a los de mayor jerarquía individual.

De todas formas, el buen comportamiento de estas optimizaciones depende fuertemente de los datos y de su complejidad, siendo necesario caer en prueba y error para encontrar aquella que se adapte mejor sin afectar significativamente la calidad del modelo.

Aunque *Naive Bayes* no es un buen algoritmo para manejar atributos con dependencias o atributos redundantes, en la práctica se ha observado que, uti-

lizado como medida de decisión en algoritmos de selección hacia adelante (*Forward Selection*), ambos en conjunto se pueden comportar igual o mejor que clasificadores de árboles sin perder la capacidad de funcionar bien en conjuntos de datos en los que Naive Bayes funciona bien. Esta técnica es llamada *Selective Naive Bayes*.

Capítulo 7

Conclusiones

El Aprendizaje Automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica. Sin embargo uno de los mayores problemas asociados a la detección de fraude es la falta de literatura que contenga resultados experimentales y la obtención de datos reales. Esto se debe a que la detección de fraude está asociada con información financiera sensible y confidencial.

Debido a las características del fraude, un sistema de detección de fraude debe tener las siguientes características:

- Habilidad para manejar grandes volúmenes de datos ya que diariamente se realizan millones de transacciones.
- Capacidad para manejar distribuciones asimétricas ya que las transacciones fraudulentas representan aproximadamente un 0,1 % de las transacciones totales.
- Capacidad para detectar variados patrones.
- Capacidad de adaptarse a nuevos patrones de fraude ya que luego de descubierto un patrón los defraudadores tratan de encontrar nuevas formas de cometer fraude.
- Habilidad para manejar ruido en los datos.

Existen varios algoritmos que tienen algunas o todas de estas características que han sido aplicados a la problemática de fraude. Entre ellos se encuentran redes neuronales, redes bayesianas, algoritmos genéticos y árboles de decisión. Usualmente es necesario combinar varios de estos algoritmos para obtener buenos resultados, su elección y la manera de combinarlos depende fuertemente de la aplicación específica y de los datos disponibles.

Bibliografía

[1] José Moraes Adriana Jorba, Marcelo Lozano and Emiliano Sacchi. Fraud identifciation system. entregado como requisito para la obtención del título de ingeniero en sistemas. 2007.

[2] Leo Breiman. Random forests. 4 2007. último acceso 14 de Julio de 2008.

[3] Peter Burns and Anne Stanley. Fraud management in the credit card industry, 4 2002.

[4] Carrera de Ingeniería en Computación. Instituto de Computación Facultad de Ingeniería. Presentación de propuesta de proyecto de grado. detección de patrones de comportamiento regulares y fraudulentos sobre un conjunto de transacciones financieras. 4 2008.

[5] Ian H. Witten Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishersh, second edition edition, 2005.

[6] Lantel Group. Códigos iso países. último acceso 14 de Julio de 2008.

[7] EMVCo LLC. último acceso 14 de Julio de 2008.

[8] Diego Rivero Álvaro Rodríguez and Dieter Spangenberg. Principal. documentación del proyecto de grado.

[9] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science / Engineering / Math, 1997.

[10] Eduardo F. Morales. Programación lógica inductiva (ilp). 2006. último acceso 14 de Julio de 2008.

[11] PayTrue Solutions. The payment company. último acceso 14 de Julio de 2008.

[12] Visa. Merchant category classification (mcc) codes directory. último acceso 14 de Julio de 2008.

[13] Strategic Planning. Fair Isaac Company Wesley Kenneth Wilhelm. Manager. The fraud management lifecycle theory: A holistic approach to fraud management. 4 2004.

[14] Wikipedia. Credit card fraud. último acceso 14 de Julio de 2008.

[15] Wikipedia. Iso 8583. último acceso 14 de Julio de 2008.

[16] Wikipedia. Algoritmo evolutivo. 7 2008.

[17] Wikipedia. Data mining. 8 2008.

[18] Wikipedia. Feature selection. 7 2008.

[19] Wikipedia. Red neuronal artificial. 7 2008.

Apéndice A

ISO 8583

Número de cuenta primario (*Primary Account Numbero PAN*): Un número que identifica la cuenta del cliente, o sea, un número de tarjeta de hasta 19 dígitos. A partir del PAN se obtiene el número identificador del emisor/producto.
Código de procesamiento (*Processing Code*): Este campo contiene un código que identifica el tipo de transacción y de cuenta afectadas por la transacción. El campo es numérico de largo 6. A continuación se listan los valores posibles para las primeras 4 posiciones:

```
Posiciones 1 a 2: Tipo de Transacción
   00 Compra de Bienes/Servicios - Débito
   02 Ajuste - Débito
   11 Transacción en efectivo
   17 Scrip - Débito
   19 Tasa - Débito
   20 Retorno (de mercancías)-Crédito
   22 Ajuste - Crédito
   28 Activación y carga - Prepago
   29 Fondos de desembolso - Crédito
   30 Consulta de fondos disponibles
   72 Activación - Prepago

Posiciones 3 a 4: Tipo de Cuenta
   00 No aplica / No especificada
   10 Cuenta de ahorros
   20 Cheques
   40 Cuenta Universal (representada por ID de cliente)
```

Monto de la transacción: Contiene el monto de la transacción en la moneda especificada en el campo código de moneda. Es un campo numérico de largo 12 que no incluye punto o coma decimal y los decimales se asumen basados en el campo código de moneda.

Monto de la venta: Contiene el monto de la transacción convertido a la moneda de compra. Es un campo numérico de largo 12 que no incluye punto o

coma decimal; los decimales se asumen basados en el campo código de moneda de compra. Se utiliza para mantener el monto en múltiples monedas.

Monto en moneda de la facturación del tarjetahabiente: Contiene el monto de la transacción convertido a la moneda de facturación del tarjetahabiente. Es un campo numérico de largo 12 que no incluye punto o coma decimal, los decimales se asumen basados en el campo tipo de conversión, moneda de facturación del tarjetahabiente.

Fecha y Hora de la transacción: Contiene la fecha y hora en GMT de la transacción con formato *MMDDhhmmss*.

Tipo de conversión, venta: Contiene el tipo de conversión utilizado para convertir el monto de la transacción al monto de venta.

Tipo de conversión, moneda de la facturación del tarjetahabiente: Contiene el tipo de conversión utilizado para convertir el monto de la transacción al monto en moneda de la facturación del tarjetahabiente.

Hora local de la transacción: Contiene la hora de la transacción expresado en hora local de la ubicación del tarjetahabiente con formato *hhmmss*.

Fecha local de la transacción: Contiene la fecha de la transacción expresada en hora local de la ubicación del tarjetahabiente con formato *MMDD*.

Fecha de vencimiento: Contiene el año y mes luego del cuál se vence la tarjeta en formato *YYMM*.

Fecha de venta: Contiene la fecha en la cuál el mensaje ingresó en la red con formato *MMDD*.

Fecha de conversión: Contiene la fecha efectiva del tipo de conversión utilizada para convertir el monto de la transacción al monto de venta, con formato *MMDD*.

MCC (*Merchant Category Code*): Contiene un código describiendo el tipo de negocio o rubro del producto o servicio. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [12].

```
3001 American Airlines
3504 Hilton Hotels
5411 Supermercado
5814 Restaurantes de comida rápida.
6010/6011 ATM
```

Código de país del adquirente: Contiene un código que identifica al país del adquirente del comercio o ATM. A continuación se listan a modo de ejemplo algunos valores posibles, para ver la lista completa ver [6].

```
032 Argentina
076 Brazil
```

```
858 Uruguay
```

Código de país del emisor: Contiene un código que identifica al país del emisor de la tarjeta, usando la misma codificación que para el país adquirente.

Modo de captura de datos del POS (*POS entry mode*): Contiene un código que identifica el método utilizado para capturar el número y fecha de vencimiento de la tarjeta cuando se usa una terminal electrónica y la capacidad de captura de la terminal. A continuación se listan los valores posibles para los dos subcampos.

```
Modo de captura
   00 Desconocido, o no se usó una terminal electrónica
   01 Digitado manualmente
   02 Lectura de la banda magnética de la tarjeta, con la salvedad de que
      es posible que el CVV (código de verificación grabado en la banda)
      no se haya podido leer en forma confiable
   03 Lectura de código de barras
   04 Lectura mediante reconocimiento óptico de caracteres (OCR)
   05 Lectura de circuito integrado, la captura del CVV (en este caso,
      grabado en el chip) es confiable
   90 Lectura de la banda magnética de la tarjeta, el CVV se leyó
      correctamente
   95 Lectura de circuito integrado, el CVV no es confiable


Capacidad de la terminal
   0 Desconocido
   1 No se utilizó POS
   2 Lectura de banda magnética
   3 Lectura de código de barras
   4 Lectura mediante reconocimiento óptico de caracteres (OCR)
   5 Lectura de circuito integrado
   9 Digitación manual, sin capacidad de lectura electrónica
   N No se proporcionó
```

Adicional al número de tarjeta: Campo numérico de largo 3 usado para diferenciar entre tarjetas con el mismo número o tarjetas adicionales a la principal.

Código de condición del POS (*POS condition code*): Contiene un código que identifica las condiciones en las que se realizó la transacción. A continuación se listan los valores posibles.

```
00 Transacción normal
01 Cliente no presente
02 Terminal automática operada por el cliente (p.e. ATM)
03 Comercio o tarjeta sospechosos
05 Cliente presente, tarjeta no presente
06 Pedido de preautorización.
08 Compra por teléfono o correo; incluye transacciones recurrentes
   (como cobro automático de facturas de servicios)
```

```
10 Se verificó la identidad del cliente
51 Verificación de dirección o cuenta, no se está pidiendo autorización
   para una transacción.
59 Solicitud de comercio electrónico por una red pública, como Internet
   71 Tarjeta presente, pero no se pudo leer la banda; los datos fueron
   digitados manualmente
```

Número identificador del adquirente: Campo numérico de hasta 11 caracteres que identifica la institución financiera que actúa como adquirente.

Código de respuesta: Contiene un código que define la respuesta a un pedido de autorización. Los códigos 00 o 10 indican aprobación y el resto de los códigos indican la razón de rechazo. A continuación se listan a modo de ejemplo algunos valores posibles.

```
01 Referirse al emisor
03 Comercio inválido
15 No existe el emisor
```

Número identificador de la terminal de venta: Contiene un código que identifica la terminal.

Número identificador del comercio: Contiene un código alfanumérico que puede identificar el comercio o una terminal.

Nombre y ubicación del comercio: Campo alfanumérico que indica el nombre, ciudad, estado y país del comercio.

Moneda de la transacción: Contiene el código de la moneda del monto de la transacción.

Moneda de la venta: Contiene el código de la moneda del monto de la venta.

Moneda de la facturación del tarjetahabiente: Contiene el código de la moneda de facturación del tarjetahabiente.