Instituto de Computación
Facultad de Ingeniería
Universidad de la República

PayTrue Solutions

Análisis y Detección de Patrones de Fraude en Medios de Pago

Informe del Proyecto de Grado

Álvaro Rodríguez
Diego Rivero
Dieter Spangenberg

Supervisores:
Andrés Vignaga
Daniel Perovich

15 de septiembre de 2006

Álvaro Rodríguez, Diego Rivero, Dieter Spangenberg

Creado utilizando LaTeX 2_ε y Microsoft Visio 2003

Revision : 73

Análisis y Detección de Patrones de Fraude en Medios de Pago

Álvaro Rodríguez	alvaro.rodriguez@gmail.com
Diego Rivero	drivero@fing.edu.uy
Dieter Spangenberg	dspangenberg@paytrue.com

Resumen

El negocio de los medios de pago, que engloba las entidades financieras dedicadas a ofrecer formas alternativas de pagar por bienes o servicios, sufre grandes pérdidas debido al fraude: transacciones realizadas con el medio de pago que luego no se pueden cobrar. Estas entidades destinan personal especializado a reducir sus pérdidas por fraude, usualmente con la ayuda de sistemas informáticos que generan alertas al detectar una transacción sospechosa, conocidos como sistemas de detección de fraude. Este tipo de sistema asume que los fraudadores siguen ciertos patrones de comportamiento que permiten detectar el fraude. *PayTrue Solutions*, cliente del presente proyecto, vende un sistema de detección de fraude que se configura en forma manual, por lo que puede perder efectividad cuando los fraudadores modifican sus patrones de comportamiento. El objetivo principal del presente proyecto es encontrar enfoques que permitan actualizar automáticamente esta configuración, para lo que aplicamos dos técnicas de Aprendizaje Automático: árboles de decisión y *naive bayesian*. A su vez, indicamos el proceso a seguir para aplicar estas técnicas, que incluye el uso de varias herramientas desarrolladas como parte del proyecto. Los resultados obtenidos al aplicar estas técnicas y herramientas sobre conjuntos de datos reales comprueban la factibilidad del enfoque elegido.

Palabras clave: detección de fraude, medios de pago, árboles de decisión, *naive bayesian*, Aprendizaje Automático, clasificador.

Índice general

Capítulo 1

Introducción

Desde mediados del siglo XX, el negocio de los medios de pago ha florecido en todo el mundo [18]. Este negocio es explotado por entidades financieras que brindan a sus clientes la posibilidad de realizar transacciones con medios de pago alternativos (como ser la tarjeta de crédito) en lugar de dinero en efectivo. Estas empresas sufren pérdidas importantes de dinero debido al fraude: transacciones realizadas con el medio de pago que luego la empresa *no* puede cobrar (ver estimaciones en [20, 9, 10, 11]).

Las entidades financieras pueden ser emisores de los medios de pago que usan sus clientes (tarjeta habientes), adquirentes que afilian los comercios donde se pueden usar los medios de pago, o marcas internacionales, que permiten interactuar a adquirentes y emisores de distintos países.

Naturalmente, las entidades financieras desean reducir sus pérdidas por fraude, por motivos tanto económicos como de imagen ante sus clientes. Para esto destinan personal especializado, llamados analistas de riesgo, a controlar la actividad fraudulenta, quienes trabajan con dos objetivos principales:

- Reducir las pérdidas por fraude, detectando la actividad fraudulenta lo más tempranamente posible. Esto permite tomar acciones correctivas tales como bloquear la cuenta o investigar el comercio.
- Desalentar a los fraudadores buscando que elijan otra entidad financiera, menos eficiente en el control de fraude, como blanco de su crimen.

Usualmente los analistas utilizan algún tipo de sistema informático que genere alertas cuando estima que una transacción es fraudulenta. Este tipo de sistema se conoce como sistema de detección de fraude. Los hay de diversa índole. Algunos deben configurarse manualmente, mientras que otros mantienen su propia

configuración automáticamente. En general, se basan en la premisa de que el uso fraudulento de un medio de pago sigue ciertos patrones de comportamiento, distinto del usual, y generan alertas al identificar que una transacción sigue estos patrones.

Los sistemas que se configuran manualmente suelen perder efectividad con el tiempo, debido a que las personas que cometen fraude modifican su operativa cada vez que las entidades financieras comienzan a detectarlos. De esta forma, las tácticas más conocidas han dado lugar a patrones mucho más complejos y difíciles de detectar. En contraposición, los sistemas completamente automatizados prometen mantenerse al día con la evolución de los patrones de fraude, pero tienen algunas desventajas:

1. Un sistema de detección de fraude con buena capacidad predictiva contiene un gran conocimiento acerca los patrones de fraude corrientes. Si es de tipo «caja negra», a la entidad financiera se le dificulta aprovechar este conocimiento para tareas de prevención que escapen al alcance del mismo, como ser incrementar las medidas de seguridad al realizar cierto tipo de transacciones.

2. Si el sistema es completamente automático, el analista no tiene forma de incluir su propio conocimiento de patrones de fraude en la configuración del mismo, sino que está obligado a esperar que el sistema descubra estos patrones por su cuenta. Además, no tiene control sobre el funcionamiento del sistema, que podría haber descubierto los patrones equivocados.

Por esto los sistemas de detección de fraude más sofisticados buscan utilizar una configuración que pueda ser mantenida y comprendida por los analistas de riesgo, pero que a su vez admita la posibilidad de mejorarse en forma automática. En este sentido han cobrado interés los sistemas basados en reglas, que permiten evitar las desventajas mencionadas sin perder la capacidad de automatizar su configuración. Sin embargo, los sistemas basados exclusivamente en reglas no han logrado buenos resultados para la detección de fraude, por lo que suelen combinarse con otros enfoques, aunque siempre manteniendo el objetivo de ser comprensibles para un analista de riesgo.

La empresa *PayTrue Solutions* [8], cliente del proyecto, ofrece una *suite* de soluciones informáticas orientadas al control de fraude en entidades financieras, llamada *Risk Center*. El *Risk Center* incluye el *Detection Engine*, un sistema de detección de fraude híbrido, que evalúa cada transacción recibida por la entidad financiera según una configuración en tres fases: evaluación de reglas, cálculo de la probabilidad estimada de que la transacción sea fraudulenta (puntaje de riesgo), y comparación del puntaje de riesgo contra un umbral de activación. Si el puntaje de riesgo supera el umbral, se genera una alerta. En el *Detection Engine*, esta configuración debe mantenerse manualmente, por lo que *PayTrue Solutions* desea desarrollar un producto capaz de sugerirle mejoras, al que llamará *Smart*

Analyzer. Para esto debe encontrarse una forma de automatizar la definición de esta configuración. Con el objetivo de resolver este problema se definió el presente proyecto, con los siguientes resultados esperados:

- Documentación acerca de las características del negocio de medios de pago y de patrones de fraude comunes en este negocio.
- Propuesta de métricas que permitan evaluar el desempeño de una configuración del *Detection Engine*.
- Definición de un enfoque apropiado para resolver el problema de automatización de la configuración del *Detection Engine*.
- Implementación de un prototipo que demuestre la factibilidad del enfoque elegido.

El problema de detección automática de fraude ha sido estudiado en profundidad, inclusive en su resolución mediante reglas, en el área de Aprendizaje Automático (*Machine Learning*), que trata del desarrollo de programas que son capaces de aprender a partir de un conjunto de ejemplos. Para resolver el aprendizaje de reglas decidimos usar una técnica de Aprendizaje Automático conocida como árboles de decisión; en particular, los programas `QUEST` y `C4.5`. En cambio, para el aprendizaje del cálculo del puntaje de riesgo utilizamos una técnica distinta: un clasificador *naive bayesian*.

El uso de estas técnicas para obtener una configuración del sistema de detección de fraude involucra varias etapas, en las que es necesaria la intervención del analista de riesgo. Para facilitar su trabajo al seguir estas etapas, desarrollamos varias herramientas con propósitos específicos, que en su conjunto conforman el prototipo esperado. Los resultados obtenidos al utilizar estas herramientas con varios conjuntos de datos demuestran la factibilidad del enfoque elegido, al adecuarse a las expectativas del cliente.

Trabajos relacionados

La detección de fraude en general, y en medios de pago en particular, es uno de los problemas de interés en el área de Aprendizaje Automático. Las bases de esta área se pueden ver en el libro de Mitchell [31] o en el libro de Friedman *et al.* [27].

En un extenso relevamiento realizado por Phua *et al.* [34] se reporta el uso de variados enfoques para atacar este problema: redes neuronales, redes bayesianas, árboles de decisión, regresión estadística, sistemas expertos, *support vector machines* o meta-algoritmos usando combinaciones de los anteriores, entre otros.

Sin embargo, es difícil reproducir los resultados reportados en otras investigaciones, ya que usualmente los conjuntos de datos utilizados y los resultados concretos obtenidos no se publican debido a la necesidad de confidencialidad. Bolton y Hand [20] indican, además, que muchos de los trabajos publicados usan la detección de fraude como campo de prueba para nuevos enfoques, sin centrarse en resolver realmente el problema. Por otra parte, los trabajos suelen aplicar técnicas genéricas de Aprendizaje Automático, sin analizar en profundidad las características particulares del negocio de forma de generar un enfoque especializado según estas características.

Por otra parte, estos trabajos en general desarrollan el sistema de detección de fraude en conjunto con la automatización de su configuración. En contraposición, en el presente proyecto las características del sistema de detección están fijas, y es necesario encontrar enfoques de Aprendizaje Automático que resulten apropiados a las restricciones impuestas por su operativa.

Considerando el interés comercial directo de los sistemas de detección de fraude, es de esperarse que existan empresas dedicadas a comercializarlos. El sistema comercial más exitoso para detección de fraude, de la multinacional FairIsaac, usa una combinación de redes neuronales, árboles de decisión, y otras técnicas (no especificadas) para su sistema Falcon [4]. Este sistema es utilizado, por ejemplo, por VISA Europa [16]. La empresa FutureRoute ofrece su sistema iHex basado en programación lógica inductiva (*inductive logic programming*, ILP) [6]. Cabe destacar que el costo de puesta en producción de estos sistemas puede alcanzar varios cientos de miles de dólares al año y requerir un proyecto de varios meses de duración para adaptar el sistema a la realidad de cada entidad financiera.

Contribuciones de este trabajo

El objetivo principal de este proyecto es resolver el problema de automatizar la configuración del *Detection Engine*. Para esto es necesario buscar una técnica que permita aprender reglas y otra que permita calcular un puntaje de riesgo para una transacción.

En este sentido, este trabajo realiza las siguientes contribuciones:

- Analiza en profundidad la forma de aplicar árboles de decisión al aprendizaje de reglas en el contexto de la detección de fraude
- Describe el uso de *naive bayesian* para estimar la probabilidad de que una transacción sea fraudulenta
- Documenta resultados comparativos de aplicar dos programas de árboles de decisión distintos en las mismas condiciones

- Sugiere métricas de evaluación de sistemas de detección de fraude específicas para el negocio de medios de pago, que no aparecen en la literatura abarcada de Aprendizaje Automático
- Aplica métricas de evaluación usuales de forma de permitir evaluar por separado cada componente del sistema de detección de fraude (reglas, cálculo de puntaje de riesgo y umbral)
- Sugiere la necesidad de un programa eficiente para aplicar transformaciones sobre el conjunto de transacciones disponibles inicialmente para el aprendizaje, y describe una implementación de un tal programa
- Sugiere formas, razonables para el negocio, de mejorar la proporción de ejemplos fraudulentos en los conjuntos de datos utilizados para el aprendizaje
- Documenta las características del fraude en medios de pago

Estructura del documento

El resto de este documento se estructura de la siguiente manera. El capítulo 2 describe el negocio de los medios de pago y las características particulares del sistema de detección de fraude de *PayTrue Solutions*. El capítulo 3 analiza la resolución del problema mediante las técnicas de árboles de decisión y *naive bayesian*, y sugiere varias formas de evaluar una configuración del *Detection Engine*. El capítulo 4 describe las herramientas desarrolladas para asistir en el uso de las técnicas. El capítulo 5 documenta los resultados obtenidos al aplicar las técnicas mencionadas a distintos conjuntos de datos según las métricas de evaluación descritas en el capítulo 3. Por último, el capítulo 6 presenta las conclusiones y delinea posibles trabajos futuros de este proyecto.

Capítulo 2

El negocio de los medios de pago

El negocio de los medios de pago es un área de servicios financieros dedicada a proveer formas alternativas de pago por un bien o servicio. Esta área ha crecido considerablemente desde sus comienzos en la primera mitad del siglo XX, donde el negocio consistía en permitir hacer compras con una tarjeta plástica, respaldada por una cuenta de crédito. Actualmente existen múltiples variantes, aunque ciertas figuras permanecen constantes.

Este capítulo describe el negocio de los medios de pago, haciendo énfasis en aquellos aspectos relevantes para la detección de fraude. La sección 2.1 describe la mecánica del negocio junto con sus variantes más comunes. La sección 2.2 contiene un resumen del protocolo utilizado para intercambiar transacciones electrónicas (ISO 8583), que resulta de interés ya que define la información conocida de cada transacción. La sección 2.3 describe las características del fraude en medios de pago. La sección 2.4 describe las características del sistema de detección de fraude de *PayTrue Solutions*, el *Detection Engine*, que son de gran influencia para el proyecto ya que restringen las técnicas aplicables de Aprendizaje Automático. Por último, la sección 2.5 describe las funcionalidades esperadas del *Smart Analyzer*, que refinan los resultados esperados del proyecto.

2.1. Mecánica del negocio

La mecánica del negocio es básicamente siempre la misma: el usuario (conocido como tarjeta habiente o *cardholder*) puede obtener bienes o servicios presentando su medio de pago en lugar de dinero en efectivo o cheques. Este acto es

denominado transacción. A quien acepta el medio de pago se le llama comercio o, a veces, *card acceptor*.

Para que el comercio acepte como válido el pago del tarjeta habiente se requiere la participación de dos entidades financieras:

Adquirente El adquirente o *acquirer* afilia a los comercios que aceptan los medios de pago y se encarga de pagarles a cambio de las transacciones que los tarjeta habientes hayan efectuado. Esto le garantiza a los comercios el pago de sus transacciones y una mayor participación de mercado entre los tarjeta habientes. El negocio del adquirente se basa en cobrar a los comercios a cambio de estas ventajas, usualmente en la forma de un arancel por cada transacción, el cual suele estar alrededor del 3 % pero puede llegar a ser tanto como 10 %.

Emisor El emisor o *issuer* emite los medios de pago, los entrega a sus clientes y se hace cargo de las transacciones realizadas por ellos ante el adquirente del comercio correspondiente. Luego cobra al tarjeta habiente por estas transacciones, según el contrato establecido. Es común que el contrato involucre la disponibilidad de crédito para realizar transacciones, en cuyo caso el emisor asume el riesgo de que el tarjeta habiente no las pague. 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 tarjeta habiente no paga el total del saldo, y cobro de intereses por financiación de compras realizadas en cuotas.

El comercio suele cobrar sus transacciones antes (en el tiempo) que el tarjeta habiente las pague al emisor, inclusive en las compras en cuotas. El ciclo de pagos, mostrado en la figura 2.1, se puede resumir de la siguiente manera: primero, el adquirente le paga al comercio; luego, el emisor le paga al adquirente; por último, el tarjeta habiente le paga al emisor. Los números reflejan el orden en que se realizan los pagos.

Cuando se trata de trascender fronteras aparece una tercera figura: la marca internacional. Mediante la definición de estándares y protocolos, la marca internacional permite la interacción de tarjeta habientes y comercios de distintos países. Existen varias marcas internacionales que operan actualmente en el mercado: VISA, MasterCard, Diners, entre otras.

Si bien es común que una misma organización actúe en los roles de emisor y adquirente simultáneamente, también es muy común que existan organizaciones especializadas en sólo uno. Por ejemplo, en Uruguay operan OCA Card, que es a la vez adquirente y emisor, y VISANET Uruguay, adquirente de VISA

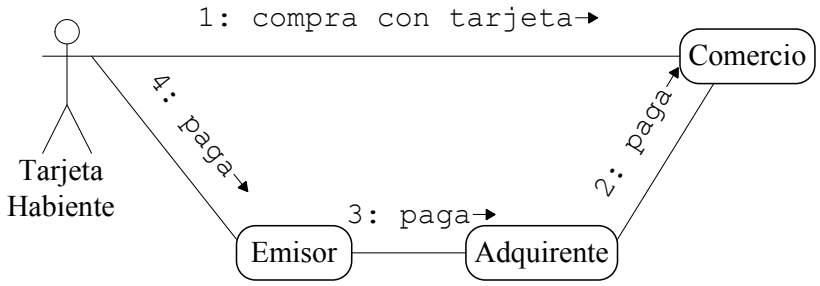

Figura 2.1: Ciclo de negocio típico

para comercios de Uruguay. A su vez, la mayoría de los bancos importantes son emisores de VISA y MasterCard.

A lo largo de este documento nos referiremos a cualquiera de estas tres entidades (emisor, adquirente o marca internacional) mediante el término «entidad financiera».

Es posible que en un mismo mercado convivan múltiples emisores y adquirentes. A su vez, es posible que un mismo comercio esté afiliado a varios adquirentes. Cuando un comercio despliega un aviso indicando que 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.

El negocio de medios de pago no consiste únicamente en hacer compras con una tarjeta de crédito. Existen múltiples variantes, tal como se describe a continuación.

Soporte físico. 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 [2]), un teléfono celular, u otros. En general, cualquier dispositivo capaz de contener una pequeña cantidad de información en forma segura puede potencialmente ser soporte físico para un medio de pago. 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.

Producto. El tipo de contrato entre el emisor y el tarjeta habiente se conoce como producto y determina, entre otras cosas:

- El respaldo financiero de las transacciones realizadas por el tarjeta habiente, que puede ser una línea de crédito otorgada por el emisor, débito contra una cuenta bancaria del tarjeta habiente, una cuenta de prepago, u otras.

- El tipo de transacciones que se pueden realizar, entre compras al contado, compras en cuotas (variando la cantidad de cuotas y el interés cobrado), devoluciones, retiros de efectivo o transferencias de dinero entre dos medios de pago.
- Las condiciones comerciales del servicio, principalmente naturaleza y monto de los cargos cobrados y límite del crédito otorgado al tarjeta habiente (si corresponde).
- La forma comercial concreta que toma el servicio proveído por el emisor, que puede ser cualquiera de las mostradas en la figura 2.2.

Procedimiento. El procedimiento utilizado para realizar una transacción depende del soporte físico, de la infraestructura encontrada en el comercio y de factores operativos. En general tiene dos fases:

- Pedido de autorización, que no tiene efecto financiero y debe ser aprobada tanto por el adquirente como por el emisor. Para pedir autorización se requieren varios datos del tarjeta habiente, que incluyen, mínimamente, el número y fecha de vencimiento de su tarjeta. La forma de capturar esta información (lectura de la banda magnética, ingreso manual, etc.) está muy relacionada con el riesgo de que la transacción sea fraudulenta. El emisor autoriza la transacción si los datos del tarjeta habiente son correctos y la tarjeta tiene dinero disponible según el monto de la transacción. El concepto de «disponible» es fundamental ya que limita lo que puede gastar un fraudador al acceder ilícitamente a una cuenta. El adquirente autoriza la transacción si los datos del comercio son correctos, y el comercio está habilitado para realizar la transacción solicitada.
- Confirmación de la transacción, idealmente refrendada mediante la firma del tarjeta habiente, que sí tiene efecto financiero. Sólo las transacciones con efecto financiero se cobran al tarjeta habiente. Es posible que algunas autorizaciones nunca sean confirmadas y no lleguen a tener efecto financiero, lo que a los efectos prácticos es equivalente a que nunca se hubiera hecho la transacción. También es posible que existan confirmaciones sin autorización previa.

2.2. ISO 8583

El ISO 8583 es el estándar para el intercambio de pedidos de autorización electrónicos hechos con medios de pago. Es utilizado por la amplia mayoría de las marcas internacionales, emisores y adquirentes en el negocio. El estándar no es público, sino que debe comprarse a la organización ISO en [3]. De todas formas en [12] se puede ver un buen resumen.

Crédito

Tarjetas de prepago Existen muchas variantes:

- Tarjetas de regalo (como un prepago no recargable)
- Tarjeta específica para viajes
- Tarjeta para uso de hijos adolescentes, o *teen card*
- Incentivos
- Envío de dinero, o *remittance*
- Pagos de cuenta
- Pago de salarios

Tarjetas de prepago específicas Telefonía o combustibles, por ejemplo

Débito

Sistemas de lealtad También conocidos como sistemas de puntos, p.e. por compras en un cierto supermercado

Figura 2.2: Posibles servicios sobre un medio de pago

Cada mensaje ISO 8583 incluye información acerca de la tarjeta usada, el comercio y el monto, entre otros campos que se describen más adelante. A su vez, no sólo pueden ser pedidos de autorización, sino también respuestas, reversas, anulaciones, términos definidos en [5]. Algunos tipos de mensaje exigen respuesta, en particular los pedidos de autorización, que deben ser aprobados o rechazados. Las respuestas deben volver por la misma ruta hasta la terminal electrónica (POS, de *Point Of Sale*) donde se hizo la solicitud original.

El ISO 8583 define el formato de mensajes y flujo de comunicación especificando cómo deben interoperar los sistemas involucrados. La amplia mayoría de las transacciones hechas en cajeros automáticos (ATM) usan ISO 8583 en algún punto de la cadena de comunicaciones, al igual que las compras con tarjeta de crédito en un comercio cualquiera. En particular, las marcas internacionales MasterCard y Visa basan sus transacciones en este estándar, al igual que muchas otras entidades financieras.

A continuación se muestra la lista de datos de una transacción intercambiados en cada mensaje ISO 8583. Estos datos son los comúnmente conocidos de cada transacción, por lo que representan la información que un sistema de detección de fraude puede considerar para decidir si generar una alerta o no. La lista contiene los campos que consideramos más significativos para el negocio, dentro de los cerca de 200 presentes en los mensajes. Los campos no descritos llevan datos de control interno o auditoría, y no consideramos que sean de interés para la detección de fraude.

Número de tarjeta: También conocido como PAN (*Primary Account Number*). A partir del PAN se obtiene el número identificador del emisor / producto. Para identificar una tarjeta se requieren también su fecha de vencimiento y número adicional (ambos son campos del ISO 8583, que se explican más adelante).

Identificador de tipo de mensaje: Llamado `MTI` por sus siglas en inglés.

Código de procesamiento: El *processing code* y el `MTI` definen el tipo de transacción (compra, retiro en efectivo, depósito, devolución, etc.).

Monto de la transacción.

Moneda de la transacción: Código ISO de la moneda correspondiente.

Fecha y hora de la transacción.

Identificador único de la transacción.

Fecha de vencimiento de la tarjeta: Identifica parcialmente la tarjeta.

MCC: Indica el rubro del comercio (estación de servicio, supermercado, ATM, hotel, etc.). En [17] se puede ver una lista de valores posibles.

Código ISO del país del adquirente: En la gran mayoría de los casos el país del adquirente coincide con el país del comercio debido a restricciones legales. De todas formas, como el país del comercio es desconocido, este campo es la mejor aproximación posible.

Código ISO del país del emisor: Es independiente de la nacionalidad del tarjeta habiente.

Modo de captura de datos del POS: Conocido como *entry mode*, identifica el método utilizado para obtener el número y fecha de vencimiento de la tarjeta cuando se usa una terminal electrónica:

- `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: Indica los modos de captura de datos soportados por el POS. Si no es compatible con el campo anterior, que es el modo de captura realmente utilizado al hacer la transacción, sugiere que la transacción se realizó en condiciones sospechosas. Los valores posibles son:

- `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: Usado en algunas redes para diferenciar entre tarjetas con el mismo número o tarjetas adicionales a la principal. Al renovar una tarjeta a veces se imprime una nueva con el mismo número pero distinto valor en este campo. Es importante diferenciarlas ya que su período de validez puede solaparse.

Código de condición del POS: Identifica las condiciones en las que se realizó la transacción:

- `00` Transacción normal
- `01` Tarjeta habiente no presente
- `02` Terminal automática operada por el tarjeta habiente (p.e. ATM)
- `03` Comercio o tarjeta sospechosos
- `05` Tarjeta habiente presente pero tarjeta no disponible
- `08` Compra por teléfono o correo; incluye transacciones recurrentes (como cobro automático de facturas de servicios)
- `10` Se verificó la identidad del cliente
- `51` Significa que no se está pidiendo autorización para una transacción, sino que únicamente se solicita la verificación del número de tarjeta, o del número de tarjeta y dirección indicados en el mensaje
- `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.

Código de respuesta: Corresponde a mensajes de respuesta a un pedido de autorización. Indica si la transacción fue aprobada (valor 00) o si fue rechazada (el valor indica la razón de rechazo, que puede ser, por ejemplo, que no se puede identificar la cuenta, que la cuenta no tiene suficiente disponible, etc.).

Número identificador de la terminal de venta.

Número identificador del comercio.

Nombre y ubicación del comercio: Además del nombre se indica la ciudad, estado y país del comercio. Todos estos datos son *strings* sin validar, por lo que es común que sean inconsistentes.

2.3. Fraude en medios de pago

Esta sección describe las características del fraude en medios de pago. La subsección 2.3.1 explica el procedimiento básico que siguen los fraudadores y algunas características generales de sus patrones de comportamiento. La subsección 2.3.2 explica más detalladamente algunos patrones de fraude vistos en la práctica. La subsección 2.3.3 describe algunos de los desafíos con los que se enfrentan las entidades financieras en sus esfuerzos por controlar el fraude. Por último, la subsección 2.3.4 explica la particularidad de la visión parcial de transacciones que tiene el negocio y que complica la detección de fraude.

2.3.1. Características

El ciclo de negocio habitual descrito en el comienzo del capítulo asume que todos los participantes actúan de buena fe. Cualquiera de estos participantes puede actuar de mala fe, no sólo un tarjeta habiente o un comercio; puede suceder que un empleado del emisor o del adquirente realice fraude interno, cuyas pérdidas pueden llegar a superar las de fraude externo.

Estos actos de mala fe están destinados en última instancia a realizar transacciones que no le serán pagadas al emisor, conformando así la figura de fraude. A la persona que lo comete se le llama defraudador o fraudador. El fraude se puede producir luego de que hayan ocurrido varios actos de mala fe; el acto concreto de no pagar una transacción es sólo su punto final.

Las entidades financieras suelen crear un área destinada exclusivamente a combatir el fraude, a la que le llaman área de administración de riesgo. Su objetivo es bloquear aquellas cuentas o comercios que cometan fraude, de forma que no puedan realizar más transacciones. Para esto es importante conocer qué características o patrones tienen en común las transacciones fraudulentas. Aunque existen tantos patrones como pueda imaginar un fraudador, todas las formas de realizar fraude siguen las etapas descritas a continuación:

1. **Obtener información de una cuenta (compromiso de la información).**
   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:

   - *Skimming*, que es el proceso de copiar subrepticiamente la información de la banda magnética de una tarjeta.
   - Robo de tarjetas.
   - Datos comprometidos por diversas fallas de seguridad, como ser 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.
   - Solicitudes de tarjeta con datos falsos.
   - Solicitud de cambio de dirección haciéndose pasar por un tarjeta habiente, para luego solicitar una reimpresión de la tarjeta o emisión de una tarjeta adicional, que será enviada a la nueva dirección.
   - Se comete fraude con la propia cuenta del fraudador, situación conocida como autofraude.
   - ... y muchas más.

2. **Utilizar esta información para realizar fraude.**
   Consiste en intentar realizar transacciones que no serán pagadas, usando los datos disponibles. En realidad las actividades concretas dependen en cierta medida de la información que tenga el fraudador:

   - Si conoce el disponible.
   - Si tiene una tarjeta física con la cual ir a un comercio, o tiene sólo 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.
   - Si cuenta con un comercio cómplice, que puede ser inclusive un comercio creado temporalmente con el único propósito de cometer fraude.

La operativa concreta seguida por los fraudadores determina patrones de comportamiento que se pueden observar a partir de los datos de las transacciones. Estos tienen las siguientes características generales:

- El volumen de transacciones legales y fraudulentas varía independientemente, por lo que la proporción de fraude sobre el total puede cambiar en el tiempo.

- Simultáneamente se pueden observar múltiples patrones de fraude. Cada patrón puede ser ocasional, regular, estacional, o seguir varios tipos de comportamiento en el tiempo.
- El comportamiento legítimo puede variar en el tiempo.
- Luego de que se descubre y bloquea el *modus operandi* de los fraudadores profesionales ellos reaccionan modificando su comportamiento, hasta que logran evitar los controles, causando una evolución constante en los patrones de fraude.

2.3.2. Patrones de fraude

Una vez que el fraudador está en posesión de la información necesaria, pasa a intentar realizar transacciones de forma tal que la misma sea aprobada y pueda obtener una ventaja económica.

La forma en que el fraudador realiza estas transacciones depende de qué datos concretos tenga. Para realizar una transacción se requiere como mínimo el número y fecha de vencimiento de la tarjeta. Un fraudador que logre obtener estos dos simples números (quizás simplemente mirando por encima del hombro de su víctima en una tienda) puede, por ejemplo, hacer compras por algunos sitios de Internet o por teléfono, ya que son escenarios de «tarjeta no presente»: el comercio no tiene forma de comprobar si el cliente realmente tiene la tarjeta en su poder.

Para evitar esta situación, muchos emisores colocan un número aleatorio impreso en la parte trasera de la tarjeta, y exigen al comercio que solicite este número en escenarios de tarjeta no presente. Otra medida de seguridad es exigir que la dirección de entrega coincida con la dirección del cliente. De todas formas no todos los comercios hacen estas verificaciones. Por lo tanto, en general se considera que este escenario es mucho más riesgoso que cuando la información de la tarjeta es capturada por una terminal electrónica.

Los fraudadores no cometen fraude con cualquier tipo de mercancía, sino que les interesan aquellas de fácil reventa para obtener un beneficio directo: mercancía electrónica, joyas, ropa de marca, o similares. Recordar que el tipo de mercancía se identifica mediante el `MCC`. Otros `MCC` comúnmente relacionados con el fraude son las estaciones de servicio, telecomunicaciones, agencias de viaje, hoteles y supermercados. Un comercio puede estar registrado con varios `MCC` si tiene sectores o departamentos con diferentes POS, lo cual permite al fraudador (si un empleado del comercio es cómplice suyo) hacer pasar sus transacciones por `MCC` de apariencia menos riesgosa. Por ejemplo, en estaciones de servicio que tienen un mini-supermercado, pasar una compra de combustible por la caja del supermercado, para evitar muchos controles definidos para el `MCC` de combustible, que suele ser el que concentra la mayoría del fraude.

A su vez, un número y fecha de vencimiento de tarjeta cualquiera pueden ser inválidos: quizás la tarjeta no esté activa, o no tenga suficiente disponible. Para comprobar la validez de la tarjeta, los fraudadores suelen intentar realizar transacciones sencillas por poco dinero. Una vez comprobada la validez de la tarjeta, el fraudador probablemente intentará hacer todas las compras posibles hasta agotar el disponible de la tarjeta.

Podríamos entonces definir un primer patrón de fraude, muy básico, que se compone de al menos dos transacciones (de tarjeta no presente): una transacción por un monto pequeño en un cierto `MCC`, seguida de varias compras en comercios de otros `MCC`. Es común inclusive que todas las compras siguientes sean en el mismo comercio o en comercios del mismo `MCC`.

Otra forma de conseguir los datos de una tarjeta es el *skimming*. Usando un lector de banda magnética, esencialmente muy similar a un POS, un comerciante deshonesto, o empleado deshonesto de un comercio honesto, puede copiar la información de la banda magnética de las tarjetas que se usan en su comercio. Esta información puede ser luego impresa en una nueva tarjeta o inclusive copiada por encima de la banda de una tarjeta ya existente. Esto último tiene la ventaja (para el fraudador) de que los elementos físicos de seguridad de la tarjeta, similares a los de un billete, no necesitan ser falsificados.

Las tarjetas así creadas pueden ser usadas para hacer compras fraudulentas en escenarios de tarjeta presente, normalmente considerados como más seguros gracias a la información codificada en la banda magnética de la tarjeta, que contiene el número y fecha de vencimiento de la tarjeta pero además un código aleatorio de verificación.

En este escenario, es muy probable que el fraudador pueda realizar las transacciones que desee sin obstrucción alguna. Sin embargo, si intenta sencillamente gastar el disponible de la tarjeta lo más rápido posible, seguirá un patrón de compra fácil de identificar, similar al ya mencionado: varias compras en un mismo comercio, o en en distintos comercios cuyo `MCC` está dentro de aquellos más relacionados con el fraude. Recordar que lo interesante para la entidad financiera es identificar este patrón lo antes posible; detectarlo a la segunda transacción es bueno, a la tercera o cuarta es aceptable, más allá es demasiado tarde, en general.

Este patrón de compra tan simple probablemente esté en el sistema de detección de fraude del emisor de la tarjeta, en un tipo de control conocido como *velocity check* (chequeo de velocidad, porque controlan la velocidad con la que se hacen transacciones). Esto significa que a la segunda o tercera compra hecha en estas condiciones probablemente el emisor ya haya bloqueado la tarjeta. El fraudador debe entonces intentar engañar al sistema de detección utilizando un patrón de compras más difícil de detectar.

Una alternativa es recorrer distintos comercios lo más rápido posible, de forma de poder hacer varias compras sin dar tiempo al emisor de detectar la situación.

Esto es fácil de hacer en los grandes centros de compra que agrupan muchos comercios: realiza una compra en una perfumería, otra en una joyería, otra en una tienda de electrodomésticos, etc. Aquí se puede detectar el uso fraudulento por la escasa diferencia de tiempo entre cada compra, o por una desviación respecto al uso normal del tarjeta habiente: quizás no suela gastar 500 dólares al mes en perfumes, o nunca use su tarjeta más de tres veces en un período de una hora, etc. Por esto es importante controlar las transacciones contra el uso promedio del tarjeta habiente.

¿Qué ocurriría si el sistema de detección de fraude fuese tan sofisticado que detecta estos casos? Algunos fraudadores se limitan a utilizar sólo una vez cada tarjeta clonada mediante *skimming*, con lo cual evitan cualquier posibilidad de ser detectados mediante *velocity checks*. Sin embargo, es posible detectar la presencia de fraude gracias a la concentración de fraude en un cierto emisor, producto o comercio en el tiempo respecto a lo normal.

2.3.3. Control de fraude

El control de fraude se realiza en 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.

Existen circunstancias que dificultan tanto la tarea de control de fraude como la simple recolección de información al respecto. Esto sucede debido a la forma relativamente indirecta en que las entidades financieras pueden detectar que fueron víctimas de fraude. Si la entidad financiera no utiliza un sistema de detección de fraude, sólo se entera del fraude cuando un tarjeta habiente denuncia que no realizó una transacción. Recordar que el propio tarjeta habiente puede ser el fraudador, con lo cual habría realizado la transacción pero la denunciaría como fraudulenta. Al acto de denunciar una transacción como fraudulenta se lo conoce también como repudiar la transacción.

En los capítulos posteriores se usan los términos «fraude» o «denuncia de fraude» siempre para denotar el reconocimiento por parte de la entidad financiera de que ciertas transacciones fueron fraudulentas. Este reconocimiento tarda regularmente entre uno y tres meses (o a veces más) por razones operativas. Un tarjeta habiente sólo puede denunciar que no hizo una transacción luego de que el emisor se la intente cobrar en el estado de cuenta siguiente a la transacción, lo cual tarda aproximadamente un mes. A su vez estas denuncias pasan por un proceso de verificación, donde se coteja la información disponible para comprobar si el tarjeta habiente efectivamente realizó la transacción o no. Como mínimo normalmente se verifica si el comercio tiene un comprobante de venta firmado por el tarjeta habiente antes de considerar que la denuncia se corresponde realmente a un acto fraudulento.

La identificación de fraudes no es completamente efectiva. A continuación se describen algunas situaciones que se dan en la realidad y dificultan esta tarea.

- Puede suceder que un tarjeta habiente que usa mucho su tarjeta reciba un estado de cuenta con transacciones que él no realizó, pero no se de cuenta y las pague al emisor. En este caso el fraude queda sin detectar.
- A veces los tarjeta habientes denuncian como fraudulentas transacciones de rubros cuestionables (apuestas o pornografía, por ejemplo). En realidad las transacciones fueron realizadas en forma legítima pero por vergüenza o para evitarse problemas familiares las denuncian como fraudulentas. Este tipo de fraude suele ser puntual por lo que es casi imposible de predecir.
- Es común que en algunos comercios (normalmente donde los empleados reciben sueldos muy bajos) se obvien los controles más básicos de seguridad, lo que facilita la aparición de fraudes fácilmente evitables. A título anecdótico, ver [1].
- En su afán por incrementar su base de clientes, a veces las entidades financieras aceptan solicitudes de tarjeta de legitimidad dudosa, lo que facilita el robo de identidad. Ver el experimento social de [13].
- Ha sucedido también en varias oportunidades que se registran accesos no autorizados a bases de datos de empresas de entidades financieras, mediante lo cual se comprometen varios cientos de miles de números de tarjeta válidos. Un caso es el de CardSystems, una empresa procesadora de tarjetas. En junio de 2005 se supo que, durante una falla de seguridad, se comprometieron entre 200,000 y 40 millones de números de tarjetas, con sus correspondientes códigos de verificación y nombres de clientes [15].

Más allá de los patrones de fraude introducidos en la subsección anterior, a continuación se resumen varios puntos considerados como básicos en el control de fraude en medios de pago.

- 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 cuenta en la que hubo un cambio de dirección.
- Transacciones de una misma cuenta en cortos periodos de tiempo (caso conocido como «ensayo de cuenta»).
- Pagos adelantados en cheques de terceros o contra otras cuentas, previos al vencimiento.
- Transacción originada en un país considerado como riesgoso.

- 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: escenario de tarjeta no presente; modo de captura de los datos inconsistente con la capacidad de la terminal (terminal con capacidad de lectura de banda pero transacción digitada manualmente); 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 restaurants, 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»).
- 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.

2.3.4. Visión parcial de los datos

Al dedicarse a la detección de fraude, las entidades financieras deben enfrentar las dificultades causadas por su visión parcial de los datos. Sólo aquellas entidades que son a la vez emisor y adquirente conocen todos los datos de las transacciones realizadas. La figura 2.3 muestra un escenario posible de marca internacional que tiene asociados un emisor, un adquirente, y una entidad que es a la vez emisor y adquirente. Los tarjeta habientes de ambos emisores pueden comprar en los comercios afiliados por los dos adquirentes.

En este escenario, ninguna entidad conoce toda la información acerca de las transacciones.

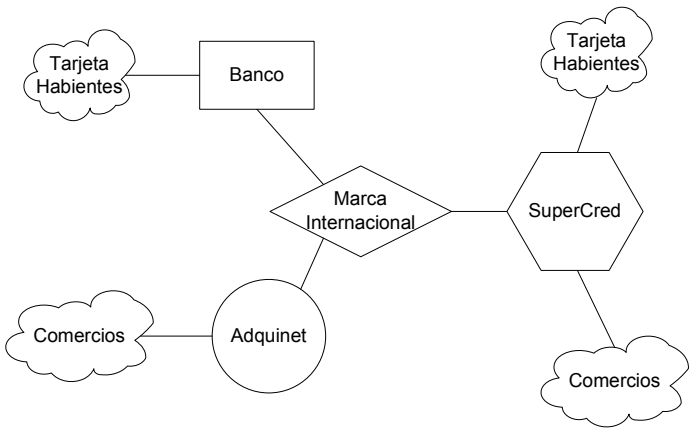

Figura 2.3: Ejemplo típico de marca internacional

- La marca internacional conoce:
  - Las transacciones realizadas por tarjeta habientes de *Banco*
  - Las transacciones realizadas por tarjeta habientes de *SuperCred* en comercios de *Adquinet*
- *SuperCred* conoce:
  - Las transacciones realizadas por tarjeta habientes de *Banco* en sus comercios
  - Las transacciones realizadas por sus tarjeta habientes en cualquier comercio
- *Banco* y *Adquinet* sólo conocen las transacciones realizadas por sus tarjeta habientes y en sus comercios respectivamente.

En general, los adquirentes sólo conocen aquellas transacciones realizadas en sus comercios, pero no todas las transacciones de los tarjeta habientes que compraron allí. Simétricamente, los emisores sólo conocen aquellas transacciones realizadas por sus tarjeta habientes, 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. En este protocolo, los datos tales como número de tarjeta, nombre del comercio, etc. son simples atributos de cada transacción.

Los datos adicionales a los de este protocolo son conocidos únicamente por el emisor o adquirente al que pertenece el tarjeta habiente o comercio respectivamente. Algunos de estos datos pueden ser de interés para la detección de fraude, por ejemplo:

- Monto disponible en la cuenta.
- Límite de crédito de la cuenta.
- Identificación de la persona física responsable de la cuenta, de forma de poder asociar comportamiento de distintas cuentas que pertenecen a la misma persona.
- Información de la dirección del tarjeta habiente, incluyendo el historial de cambios de dirección.
- Con qué `MCC` puede vender el comercio, lo que permitiría considerar todos los `MCC` posibles del comercio y no sólo el de la transacción.
- Fecha de fundación del comercio, que es importante por la problemática de los comercios «migratorios» (creados temporariamente con el único fin de cometer fraude).
- Información referente al nombre y ubicación física del comercio más fidedignos que los intercambiados en la mensajería ISO.
- Historial de pagos realizados por el tarjeta habiente, incluyendo si existieron pagos rechazados (por haber sido hechos con cheques sin fondos, por ejemplo).
- Fecha de vencimiento del próximo pago pendiente del tarjeta habiente.

2.4. El *Risk Center*

El *Risk Center* de *PayTrue Solutions* es una *suite* de varios productos destinados a una entidad financiera que desee controlar el fraude cometido con medios de pago. Uno de ellos, el *Detection Engine*, es un sistema de detección fraude. *PayTrue Solutions* desea agregar al *Risk Center* un producto llamado *Smart Analyzer* que permita automatizar el proceso de configuración del *Detection Engine*. Uno de los resultados esperados de este proyecto es una propuesta de cómo resolver los requerimientos del *Smart Analyzer*, por lo que es importante entender su relación con los demás productos del *Risk Center*, que son:

Detection Engine Orientado a la detección de fraude mediante la generación de alertas para transacciones posiblemente fraudulentas.

Advanced Console Orientado al procesamiento de alertas generadas ya sea por el *Detection Engine* u otro sistema.

Common Purchase Point* (CPP) *Finder Identifica potenciales CPPs: comercios en los que ha sido comprometida información de tarjetas que luego podría ser utilizada en transacciones fraudulentas.

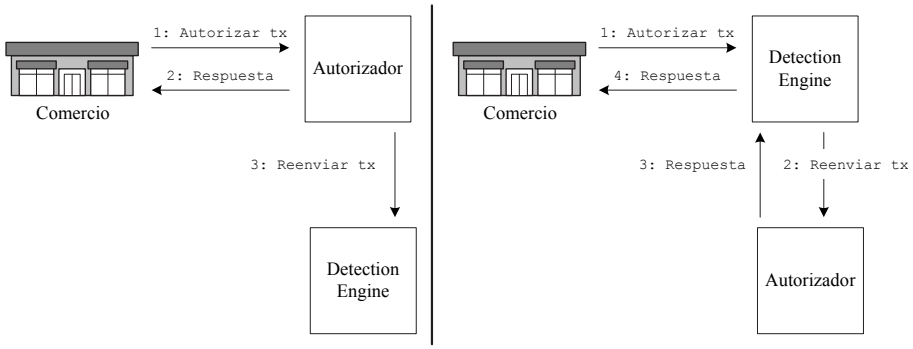

Figura 2.4: Interacción entre el autorizador y el *Detection Engine*

Suspicious Activity Manager Permite realizar un análisis de datos sobre el histórico de transacciones y fraudes para la detección de actividad sospechosa en cuentas y comercios.

Statistics 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.

El *Detection Engine* se instala en una entidad financiera de forma tal que le lleguen todas las autorizaciones, ya sea antes o después de pasar por el autorizador de la entidad financiera. La figura 2.4 ilustra estas dos posibilidades. Aún en caso de recibir la autorización después que pase por el autorizador, la misma llega al *Detection Engine* para su evaluación en forma casi inmediata. En este sentido decimos que su funcionamiento siempre es *online*.

El *Detection Engine* evalúa cada transacción así recibida para decidir si generar una alerta o no. Esta decisión se toma de acuerdo al proceso de evaluación en tres fases ilustrado en las figuras 2.5 y 2.6.

Este proceso requiere una configuración (definición de reglas, forma de cálculo del puntaje de riesgo, umbrales) que debe realizarse en forma manual. El *Detection Engine* no infiere o determina por sí mismo esta configuración. Como los fraudadores están continuamente modificando su comportamiento, es fundamental contar con una forma de evitar que decaiga la efectividad de las alertas generadas por el *Detection Engine*. El objetivo del *Smart Analyzer* es asistir en esta tarea.

Si el *Detection Engine* genera una alerta, los analistas de riesgo de la entidad financiera pueden iniciar un proceso de investigación que, eventualmente, culmine en el bloqueo de la cuenta correspondiente. Este proceso puede incluir pasos

Fase 1: Evaluación de reglas El *Detection Engine* tiene configurado un conjunto de reglas o predicados $r{:}Tx \rightarrow \{fraude, legitima\}$, que son siempre de la forma:

si (condiciones) $\rightarrow$ *fraude*
si no $\rightarrow$ *legitima*

Cuando una transacción *tx* cumple que *r(tx) = fraude* decimos que *r* clasificó a *tx* como fraudulenta.

Fase 2: Cálculo de puntaje de riesgo Esta fase, también llamada *scoring*, consiste en calcular un puntaje de riesgo para aquellas transacciones que fueron clasificadas como fraudulentas por alguna regla.

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

Figura 2.5: Fases del proceso de generación de alertas del *Detection Engine*

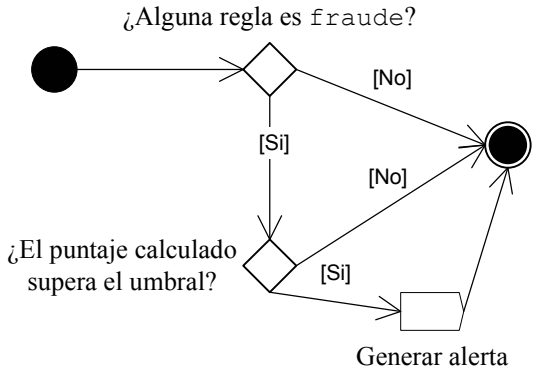

Figura 2.6: Proceso de generación de alertas del *Detection Engine*

tales como verificar el historial de transacciones y pagos del tarjeta habiente, el historial de transacciones del comercio, llamar telefónicamente al tarjeta habiente para comprobar si realmente realizó la transacción, etc.

Si el *Detection Engine* se instala por delante del autorizador, puede tener la posibilidad de indicar a éste que conteste «referido» a una solicitud de autorización. Esto significa que el emisor desea contactarse telefónicamente con el comercio para comprobar la identidad del tarjeta habiente, consultando datos personales, tales como el apellido de soltera de su madre o similares.

En cambio, si se instala por detrás del autorizador, no existe esta posibilidad, por lo que al momento de generar la alerta ya se autorizó la transacción. Por lo tanto, por más que resultase fraudulenta, ya fue realizada cuando los analistas de riesgo reciben la alerta. En esta modalidad, el objetivo de la generación de

alertas y las investigaciones no es impedir una transacción fraudulenta concreta sino bloquear las cuentas que realizan fraude.

A continuación se explican detalladamente las fases del proceso de generación de alertas del *Detection Engine*.

2.4.1. 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:

si (condiciones) → *fraude*
si no → *legitima*

Por ejemplo, una regla muy sencilla podría ser:

```
MCC = 6010 AND MONTO > 10000
```

Si una transacción es clasificada como fraudulenta por alguna regla, se considera que la transacción es sospechosa y se evalúan las siguientes fases. Si todas las reglas devuelven *legitima*, la transacción no es sospechosa y no se genera una alerta para ella. Las reglas expresan condiciones sobre las propiedades de cada transacción. 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: nombre del tarjeta habiente, dirección del comercio, etc. 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 este tipo de propiedades 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 que se está calculando la propiedad). De esta forma se pueden 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

Este tipo de propiedades es más difícil de caracterizar que las reglas de nivel 2. De hecho el *Detection Engine* no las provee directamente. Para subsanar este problema 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 AND
TX_1.MONTO > TX_2.MONTO AND
TX_2.MONTO > TX_3.MONTO
```

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

2.4.2. Cálculo de 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. Pueden llegar a haber reglas que tengan tan poco como un 3 % de aciertos, es decir, que sólo aciertan al clasificar una transacción como fraudulenta aproximadamente 1/30 de las veces.

Para afinar más el resultado, el *Detection Engine* genera alertas sólo para aquellas transacciones cuyo puntaje de riesgo o *risk score* supere un cierto umbral. Este puntaje $p \in [0, 1]$ 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. El *Detection Engine* cuenta con un mecanismo de extensiones para sustituir el componente que realiza este cálculo.

El umbral se usa para ajustar la proporción de falsas alertas generadas por el *Detection Engine*. Al subir el umbral, se generan menos falsas alertas pero se acepta más fraude, mientras que al bajarlo, 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 se podría llegar a predicciones contradictorias acerca de la transacción. En esta situación el *Detection Engine* genera una alerta si al menos una de las predicciones indica que la transacción es fraudulenta.

2.5. El *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*. Esta

sección explica los principales requerimientos del *Smart Analyzer*, para cuya resolución se definió el presente proyecto.

Los usuarios del *Smart Analyzer* son responsables de reducir las pérdidas por fraude de la entidad financiera. Con este objetivo utilizarán el *Smart Analyzer* de forma periódica (por ejemplo semanal o mensualmente) para descubrir posibles modificaciones en la configuración vigente del *Detection Engine*. Esta operativa admite que la ejecución del *Smart Analyzer* insuma varias horas de ser necesario. A su vez, los análisis necesarios no deben afectar el funcionamiento normal de la entidad financiera. Por esto, el *Smart Analyzer* debe funcionar de modo *offline*, es decir, desconectado del flujo de transacciones de la entidad financiera. Recordar que el *Detection Engine* funciona de modo *online*.

Para poder ejecutar el *Smart Analyzer* es razonable esperar que su usuario tenga acceso a 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.

El *Smart Analyzer* sugiere modificaciones en la configuración del *Detection Engine* que el analista de riesgo considerará; aquellas modificaciones que le resulten interesantes o novedosas las incluirá en la configuración del *Detection Engine*.

Por otra parte, el analista puede conocer reglas que considere interesantes, por ejemplo porque le fueron sugeridas a partir de nuevos patrones de fraude observados en otras entidades financieras. Es importante que pueda experimentar con estas reglas para comprobar si mejoran o no la configuración del *Detection Engine*. Además de las reglas, puede interesarle ajustar el cálculo de puntaje de riesgo o umbrales definidos. Para esto, el *Smart Analyzer* debe permitirle probar el funcionamiento de estas reglas sobre su base de datos de transacciones, en una modalidad de uso conocida como *sandbox*.

En definitiva el analista de riesgo es el responsable final de manejar la configuración del *Detection Engine*. Para esto le sirven de entrada tanto las sugerencias del *Smart Analyzer* como las recomendaciones de analistas de riesgo de otras entidades financieras.

Todas estas funcionalidades deben ser ofrecidas por el prototipo del *Smart Analyzer* desarrollado como parte del presente proyecto.

Capítulo 3

Automatización de la configuración

El objetivo principal del proyecto es resolver la automatización de la configuración de un sistema de detección de fraude híbrido, basado en una combinación de reglas y comparación del puntaje de riesgo contra un umbral de activación. El recurso disponible para lograr este objetivo es la base de datos de la entidad financiera, donde se encuentran todas las transacciones realizadas previamente, marcadas como fraudulentas o legítimas. Como los fraudadores siguen ciertos patrones de conducta, es de esperarse que de esta información se puedan extraer reglas que los revelen.

Por definición, el desarrollo de programas que pueden aprender a partir de ejemplos o de la experiencia concierne al área de Aprendizaje Automático o *Machine Learning*. Por lo tanto buscamos cómo alcanzar los objetivos utilizando técnicas de esta área.

Una definición amplia, dada por Mitchell [31], de qué significa que un programa aprenda es la siguiente: se dice que un programa aprende cuando mejora su desempeño (respecto a alguna medida) en una cierta tarea a través de la experiencia.

En este proyecto, interesa un programa que mejore su desempeño en la tarea de detección de fraude a partir de la experiencia dada por la base de datos de transacciones marcadas. La medida de desempeño es la calidad de las alertas generadas por el programa, en cuanto a su precisión (qué porcentaje de alertas se corresponden con transacciones que finalmente resultaron fraudulentas) y cobertura (qué porcentaje de las transacciones fraudulentas se detectó). Son posibles otras medidas de desempeño, que se discuten en la sección 3.4.

La tarea de detección de fraude se puede caracterizar como una tarea de clasificación: dada una transacción, clasificarla en fraudulenta o legítima (a los efectos de generar una alerta si se la clasifica como fraudulenta). Es un tipo de tarea muy común en Aprendizaje Automático, donde a los programas dedicados a resolverla se conocen como clasificadores.

Las técnicas de Aprendizaje Automático definen un modelo o representación interna de la realidad y un mecanismo de aprendizaje para mejorar su desempeño a través de la experiencia. Las distintas técnicas utilizan diversas representaciones y mecanismos de aprendizaje, que los hacen más apropiados para cierto tipo de problemas.

El funcionamiento del *Detection Engine* determina que sean necesarias dos técnicas de Aprendizaje Automático: una para el aprendizaje de reglas y otra para el aprendizaje del cálculo del puntaje de riesgo. Las secciones 3.1 y 3.2 describen las técnicas aplicadas en cada caso. Luego, la sección 3.3 discute cómo seleccionar un umbral de activación apropiado. Por último, la sección 3.4 discute cómo evaluar clasificadores, y sugiere algunas formas de evaluación particulares para el problema de detección de fraude en medios de pago.

3.1. Aprendizaje de reglas

Para elegir el enfoque a aplicar para el aprendizaje de reglas, analizamos en primer lugar las técnicas utilizadas en otros sistemas de detección de fraude.

En un extenso relevamiento realizado por Phua *et al.* [34] se reporta el uso de variados enfoques para atacar el problema de la detección de fraude: redes neuronales, redes bayesianas, árboles de decisión, regresión estadística, sistemas expertos, *support vector machines* o meta-algoritmos usando combinaciones de los anteriores, entre otros.

El sistema comercial más exitoso para detección de fraude, de la multinacional FairIsaac, usa una combinación de redes neuronales, árboles de decisión, y otras técnicas (no especificadas) para su sistema Falcon [4]. Por su parte, la empresa FutureRoute ofrece su sistema iHex basado en programación lógica inductiva (*inductive logic programming*, ILP) [6].

Dado que el objetivo es la producción de reglas, aquellos enfoques que no las producen no necesitan ser considerados. Así, los dos enfoques posibles son árboles de decisión e ILP. Provost y Perlich [33] sugieren que el desempeño de ILP no es bueno en dominios donde los atributos numéricos son importantes, o que dependen fuertemente de atributos complejos creados a partir de agregaciones. Conviene recordar que, en los medios de pago, el atributo más importante (el monto) es numérico, y que los atributos creados a partir de agregaciones (llamados propiedades de nivel 2 en el capítulo anterior) son fundamentales. Por otra

parte, en el relevamiento de Phua *et al.* [34] hay varias aplicaciones de árboles de decisión a la detección de fraude, pero ninguna de ILP. En base a los resultados de estos trabajos decidimos utilizar árboles de decisión.

Algunas observaciones generales respecto a los árboles de decisión indicadas por Friedman *et al.* en [27] son:

- Son notablemente robustos frente a la presencia de muchos atributos, incluyendo atributos no interesantes para la clasificación. Como es posible crear múltiples atributos de nivel 2 y 3, los cuales no se sabe a priori si son relevantes o no, es importante poder aplicar el algoritmo a conjuntos descritos por varias decenas de atributos, de los que sólo unos pocos son los relevantes.
- No tienen problemas al usar atributos heterogéneos (diferente semántica, distintas escalas de medida, distinto tipo, atributos que admiten valores nulos). Esto es necesario dada la disparidad de atributos que describen una transacción, según se explicó en el capítulo anterior. Otros enfoques de Aprendizaje Automático pueden tener problemas con algunos de estos casos.
- Producen modelos interpretables, lo cual es fundamental en este caso dada la necesidad de producir reglas.
- Indican claramente los atributos relevantes (aquellos elegidos para la condición de una regla), lo cual es interesante para el analista de riesgo ya que le permite comprender fácilmente los puntos vulnerables para el fraude.

3.1.1. Representación de la realidad

Como se mencionó en la introducción de este capítulo, las técnicas de Aprendizaje Automático utilizan una cierta representación de la realidad para realizar su tarea. En el caso de los árboles de decisión esta representación es muy simple, y puede verse dentro del Modelo Relacional como una única relación en primera forma normal, o `1NF`. Cada tupla de la tabla es un ejemplo de la realidad. Uno de los atributos de la relación es especial: la clase objetivo, o *target class*, que es el atributo para el que se desea aprender reglas a partir de los demás. En este caso, la clase objetivo es la clasificación de la transacción como fraudulenta o legítima.

Esta representación es adecuada a nuestra realidad, ya que debemos aprender reglas cuyo dominio es, precisamente, el conjunto de atributos de una transacción. Por lo tanto, los atributos disponibles para describir ejemplos son los intercambiados en el protocolo ISO 8583 (ver sección 2.2). Además, si la entidad financiera es un emisor o un adquirente, es posible disponer de atributos adicionales, como ser el nombre del tarjeta habiente o la dirección del comercio.

Sin embargo, estos atributos no son suficientes para expresar reglas de nivel de complejidad mayor a 1 (ver sección 2.4.1). Las reglas complejas requieren atributos calculados a partir de varios ejemplos. Como estos atributos no están disponibles en la estructura de datos original, es necesario agregarlos explícitamente, es decir, transformar la estructura de original de los atributos. Se puede ver esta transformación como otro problema de Aprendizaje Automático: a partir de un conjunto de transacciones clasificadas, determinar atributos complejos (calculados a partir de varias transacciones) que sean relevantes para aprender reglas que clasifiquen correctamente transacciones desconocidas.

El enfoque de árboles de decisión no permite solucionar este nuevo problema, sino que se requiere un algoritmo de aprendizaje capaz de considerar operadores complejos de agregación. En su forma más general, no es una funcionalidad disponible actualmente en ningún clasificador, como concluyen Provost y Perlich en [33], donde analizan en forma rigurosa las características y complejidad de los atributos calculados a partir de relaciones entre distintos objetos.

Si no existe un enfoque capaz de resolver este problema, es necesario confiar en la experiencia del analista de riesgo para determinar qué atributos complejos conviene definir. Es posible definir cuantos atributos complejos se desee, ya que los árboles de decisión manejan bien conjuntos de ejemplos descritos por muchos atributos. Por lo tanto, para suplir la carencia de un algoritmo capaz de sugerir atributos complejos, el analista de riesgo puede seguir un proceso de prueba y error, descartando aquellos atributos complejos que no aparezcan en ninguna regla, hasta encontrar los que realmente son relevantes para la detección de fraude.

3.1.2. Mecanismo de aprendizaje

Los árboles de decisión tienen dos etapas claramente diferenciadas en su funcionamiento:

1. Etapa de entrenamiento, donde se aprende a partir de un conjunto de ejemplos clasificados, generando como salida un árbol de decisión
2. Etapa de evaluación o predicción, donde se utiliza el árbol para clasificar un ejemplo no clasificado

En la etapa de entrenamiento se forma una estructura jerárquica de preguntas en forma de árbol. Una vez terminada, se obtiene un árbol que compara un atributo con un valor en cada nodo interno y asigna un valor de la clase objetivo a cada hoja. La clase objetivo es el atributo que se desea predecir a partir de los demás; en el caso de la detección de fraude, la clasificación de una transacción como fraudulenta o legítima.

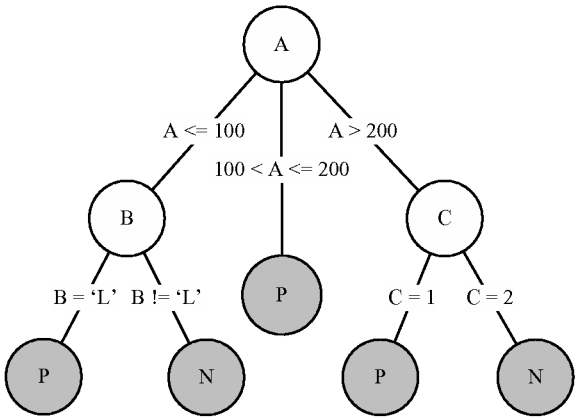

Figura 3.1: Ejemplo de árbol de decisión

En la figura 3.1 se muestra un ejemplo de árbol de decisión. Tiene tres nodos internos, marcados en blanco, y cinco hojas, marcadas en gris. La estructura de los ejemplos debe incluir al menos cuatro atributos: *A, B, C* y la clase objetivo, que (en este ejemplo) puede tomar los valores *P* y *N*.

Transformar un árbol en reglas es directo. Por cada rama se determina una regla, formada por las conjunciones de las comparaciones hechas en el camino de la raíz a la hoja. El conjunto de las reglas es la disyunción de las reglas así obtenidas. De esta forma se llega al siguiente conjunto de reglas (donde e es el ejemplo a clasificar):

- `si` $e.A <= 100$ `y` $e.B = L \rightarrow$ `predecir la clase` P `para` e
- `si` $e.A <= 100$ `y` $e.B != L \rightarrow$ `predecir la clase` N `para` e
- `si` $100 < e.A <= 200 \rightarrow$ `predecir la clase` P `para` e
- `si` $e.A > 200$ `y` $e.C = 1 \rightarrow$ `predecir la clase` P `para` e
- `si` $e.A > 200$ `y` $e.C = 2 \rightarrow$ `predecir la clase` N `para` e

Un árbol se construye en forma recursiva. Existen muchos algoritmos para esto, pero todos presentan una estructura similar a la de la figura 3.2. Esta figura muestra el seudocódigo de ID3 [35], uno de los primeros algoritmos para aprendizaje de árboles de decisión.

El seudocódigo de la figura 3.2 deja varias preguntas pendientes de respuesta: ¿Cómo se elige el mejor atributo (punto 5)?; ¿Cómo se manejan atributos continuos (como los de la figura 3.1)?; ¿Se pueden tener atributos con valores nulos?. A su vez, admite múltiples variantes, algunas de las cuales serían necesarias

1. Dado un conjunto de ejemplos E y un conjunto de atributos A
2. Crear la raíz
3. Si todos los ejemplos de E tienen el mismo valor v en su clase objetivo, la raíz es terminal y se etiqueta con v
4. Si A es vacío, la raíz es terminal y se etiqueta con la clase más común entre los ejemplos
5. Si quedan atributos y los ejemplos son de distintas clases, elegir el mejor atributo a para hacer una comparación
6. La raíz pregunta por el valor de a
7. Por cada valor v_i posible de a
   a) Crear una rama
   b) $E_{vi} = \{$Ejemplos de E en los que $a = v_i\}$
   c) Si E_{vi} es vacío, etiquetar con el valor más probable
   d) Si no, $A_{vi} = A - a$, invocar recursivamente ID3 con E_{vi} y A_{vi}

Figura 3.2: Seudocódigo de ID3

para construir el árbol de la figura 3.1. Estos temas son discutidos en profundidad por Quinlan en [36], Loh y Shih en [30], o Breiman *et al.* en [22]. No se reproducen aquí, ya que no tienen influencia directa sobre el proyecto, y una discusión detallada de los algoritmos de árboles de decisión escapa al alcance de este documento. Sin embargo, algunas consideraciones acerca de la forma de crear ramas sí son relevantes y se discuten más adelante.

Por otra parte, el seudocódigo explica solamente la fase básica de crecimiento del árbol, en la que el mismo crece todo lo posible. Se ha observado que esta estrategia crea árboles que clasifican muy bien el conjunto de entrenamiento pero luego no tiene buen desempeño frente a nuevos conjuntos de ejemplos. Esta situación, conocida como sobreajuste o *overfitting*, ha dado lugar a las siguientes estrategias para reducir el tamaño del árbol, como discute Mitchell en el capítulo 3 de [31]:

- Detener el crecimiento del árbol cuando se cumple un cierto criterio
- Permitir que el árbol crezca sin límites, y luego recortar algunas de sus ramas, estrategia conocida como *pruning*

La segunda estrategia ha dado mejores resultados en la práctica. También hay una discusión completa de este tema en la sección de árboles de decisión de [51] o en su versión impresa, de Hill y Lewicki [28].

Otra alternativa es permitir que el árbol crezca sin límites, transformarlo en reglas de la forma directa descrita anteriormente, y luego recortar las reglas, lo que se conoce como *rule post-pruning*. Esta estrategia ofrece una mayor flexibilidad al no mantener más la estructura de árbol, y es la utilizada en el programa `C4.5` de Quinlan [36].

Es importante resaltar que en el punto 6 del seudocódigo de la figura 3.2, siempre se compara un único atributo contra una constante. Esto significa que:

1. Nunca se comparan dos atributos distintos entre sí
2. Nunca se compara una combinación lineal de varios atributos contra una constante, si bien este punto ha sido objeto de investigación, dando lugar por ejemplo al algoritmo OC1 [7] (ver Murthy *et al.* [32])
3. Se crean tantas ramas como valores posibles del atributo, a menos que se use un algoritmo distinto (ver alternativas en este punto sugeridas por Loh y Shih en [30]), lo cual puede causar problemas si un atributo tiene muchos valores posibles (cientos, miles o más)

Si se considera que una comparación entre (o combinación lineal de) dos o más atributos es relevante para la detección de fraude, es necesario crear nuevos atributos calculados a partir de los originales.

Por otra parte, si existen atributos con muchos valores posibles, se debe considerar la conveniencia de utilizarlos para el aprendizaje. El uso de un atributo con más de unos pocos cientos de valores, combinado con el hecho de que se genere una rama por cada uno de esos valores, puede oscurecer la interpretación de las reglas obtenidas, y además (dependiendo del algoritmo) puede causar problemas tan graves como que el algoritmo nunca termine debido a la proliferación de ramas.

Incluso, los atributos con muchos valores son, probablemente, demasiado específicos como para ser de interés en una regla. Ejemplos típicos son los atributos únicos de una transacción (como la fecha), tarjeta habiente (como el número de tarjeta) o comercio (como su nombre). Estos atributos deben ser eliminados, de forma que al crecer el árbol no se pregunte por ellos. De lo contrario se podrían aprender reglas tales como «si la fecha de la transacción es 10/10/2005 15.31 entonces es fraudulenta», o el resultado de la ejecución de un algoritmo del estilo del de la figura 3.2 podría resultar ilegible debido a la gran cantidad de reglas generadas. Más aún, el interés de aprender reglas específicas de una cuenta o un comercio es discutible; se trata de un dato obtenible fácilmente mediante reportes, sin necesidad de usar técnicas de Aprendizaje Automático.

Un último comentario respecto al mecanismo de aprendizaje de los árboles de decisión es que son apropiados para escenarios con ruido en los datos de entrenamiento. Esta cualidad es esencial para la detección de fraude, donde se sabe que

por razones operativas los datos suelen estar distorsionados (ver sección 2.3), y se debe fundamentalmente a que se entrenan respecto a *todo* el conjunto de datos. La desventaja es que siempre que se desee considerar un nuevo atributo o un nuevo ejemplo es necesario reentrenar el árbol desde cero. Los árboles de decisión no se pueden entrenar en forma incremental.

De todas formas, la recolección de transacciones marcadas para el entrenamiento no es inmediata. Como se discutió en la sección 2.3.3, pueden pasar varios meses entre que se realiza la transacción y se sabe si la misma fue fraudulenta o no. En consecuencia, el entrenamiento incremental no es un requerimiento vital.

3.1.3. Programas utilizados

Utilizamos dos programas de aprendizaje mediante árboles de decisión: `C4.5` y `QUEST`.

`C4.5` es uno de los programas de árboles de decisión más conocidos. Sus raíces están en ID3, respecto al que incluye mejoras orientadas a aceptar atributos continuos y valores nulos, una fase de *pruning* para evitar el sobreajuste, y un generador de reglas.

El código fuente de `C4.5` (de distribución limitada) puede obtenerse de la página web de Ross Quinlan [37]. En [14] hay un tutorial de uso con explicación de los parámetros que acepta y el formato de los archivos de entrada y salida.

`C4.5` ha llegado a su octava versión, en la que Quinlan detuvo su desarrollo para centrarse en su versión comercial, `C5` o `See5` [38]. El llamado *release 8* de `C4.5` incluye varios programas. Dos de estos programas son de interés para el aprendizaje de reglas:

- `C4.5`, que aprende un árbol de decisión a partir de un archivo de ejemplos. Genera dos archivos de salida, uno con el árbol completo y otro con el resultado de aplicar *pruning*.
- `c4.5rules`, que genera reglas a partir del árbol sin *pruning* generado por `C4.5`, en un proceso que incluye el recorte de reglas (*rule post-pruning*).

Como el objetivo es la generación de reglas, utilizamos primero `C4.5` e inmediatamente `c4.5rules`.

Por su parte, `QUEST` es un programa de árboles de decisión desarrollado por Wei-Yin Loh (Universidad de Wisconsin-Madison) y Yu-Shan Shih (National Chung Cheng University, Taiwan). `QUEST` significa *Quick, Unbiased and Efficient Statistical Tree*. Está basado en las ideas de CART [22], por lo que su base teórica es diferente de la de `C4.5`.

En la página web de `QUEST` [49] se puede obtener la implementación del programa junto con un manual de uso [50]. Los conceptos de estadística en los que se basa `QUEST` están explicados en profundidad en la sección de árboles de clasificación del libro de Hill y Lewicki, en [51, 28].

El uso de `QUEST` es similar al de `C4.5`, con la salvedad de que carece de fase de *rule post-pruning* (genera un árbol de decisión sin transformar a reglas). Por lo tanto es necesario transformar el árbol a reglas siguiendo el procedimiento directo descrito previamente en la sección 3.1.2.

Las otras diferencias entre `C4.5` y `QUEST` refieren a la forma de hacer las comparaciones para generar los nodos intermedios y en el uso de *pruning*.

3.2. Aprendizaje de puntaje de riesgo

Recordar que, como se mencionó en la sección 2.4, el *Detection Engine* calcula un puntaje de riesgo para aquellas transacciones que fueron clasificadas como fraudulentas por alguna regla. Este puntaje de riesgo debe estar entre 0 y 1, y ser más cercano a 1 cuanto mayor sea la probabilidad de que la transacción sea realmente fraudulenta.

Para este problema elegimos un enfoque *naive bayesian*, también conocido como *Idiot's Bayes*. Se trata de un clasificador probabilístico sencillo, en el que la denominación *naive* surge de asumir que los atributos son independientes entre sí. Sin embargo, Hand y Yu indican que, a pesar de lo irreal de este supuesto, el enfoque produce buenos resultados [24]. Mitchell da una descripción completa de los métodos de aprendizaje bayesianos en el capítulo 6 de [31]. La implementación utilizada en este proyecto es la de [21].

Esta técnica tiene la ventaja de representar ejemplos en forma muy similar a los árboles de decisión, como se comenta en la subsección 3.2.1. En consecuencia, se trata de un enfoque aplicable con facilidad sobre conjuntos de datos ya preparados para su uso con árboles de decisión. En definitiva, decidimos utilizar esta técnica principalmente debido a su facilidad de uso y a los buenos resultados obtenidos con ella en la bibliografía.

Considerando que una transacción puede haber sido clasificada como fraudulenta por varias reglas, es necesario considerar cómo aplicar *naive bayesian*, para lo que surgen las siguientes alternativas:

- Entrenar un único clasificador *naive bayesian* a partir de todas las transacciones, y utilizarlo para calcular el puntaje de riesgo de cualquier transacción, sin importar qué regla o reglas la clasificaron como fraudulenta.

- Entrenar un clasificador *naive bayesian* distinto para cada regla, utilizando sólo las transacciones clasificadas como fraudulentas por esta regla para el entrenamiento.

La segunda alternativa resulta interesante ya que las transacciones usadas al entrenar cada clasificador bayesiano fueron preclasificadas por las reglas. Esto da lugar a un conjunto de transacciones con las siguientes características:

- Tiene menos transacciones que el conjunto original, ya que se consideran sólo aquellas transacciones clasificadas como fraudulentas por la regla.
- La proporción de fraude es igual a la precisión de la regla, que representa la proporción de transacciones fraudulentas dentro de las que la regla clasifica como tales. Por lo tanto, para cualquier regla de precisión mayor a la proporción de fraude en conjuntos reales (alrededor de 0.1 %), este conjunto tiene una proporción de fraude mayor al conjunto original. Esta afirmación se justifica formalmente en la sección 3.4.2.

Estas características resultan apropiadas para el entrenamiento de clasificadores. En general, el entrenamiento de clasificadores resulta mejor cuanto más equilibradas sean las cantidades de los distintos valores de la clase objetivo del entrenamiento, como se discute en varios trabajos (por ejemplo [52]). Por esto nos decidimos por esta alternativa.

Una observación importante es si se deben considerar todas las reglas aprendidas previamente o no. Al entrenar clasificadores *naive bayesian* se toman en cuenta únicamente aquellas reglas cuya precisión (según se define en la subsección 3.4.1), medida sobre el conjunto de entrenamiento para el aprendizaje de reglas, sea mayor a un cierto valor. De esta forma se descartan las que sean muy poco confiables. En todas las pruebas realizadas, el valor utilizado fue 3 %, que es un valor aceptable para el negocio según el cliente del proyecto.

Cabe destacar que cualquiera de las dos alternativas anteriores descartan el dato de cuántas reglas clasificaron a una transacción como fraudulenta. Podría pensarse que cuanto mayor esta cantidad, mayor es la probabilidad de fraude. Este punto no fue considerado y constituye una de las posibles mejoras a considerar, según se discute en la sección 6.3.

3.2.1. Representación de la realidad

Los programas bayesianos utilizan una representación de la realidad sencilla e igual a la usada por los árboles de decisión (una única relación en `1NF`). En general permiten utilizar cualquier tipo de atributos. Los atributos continuos, como fechas o números, deben ser discretizados, en forma similar a como se hace

con los árboles de decisión. También soportan valores desconocidos o nulos y no tienen inconvenientes frente a la presencia de muchos atributos o atributos con muchos valores.

Consideramos entonces que este enfoque es compatible con el tipo de atributos que usamos con los árboles de decisión, por lo cual no es necesario hacer ningún tipo de transformación en los conjuntos de ejemplos.

3.2.2. Mecanismo de aprendizaje

Los clasificadores *naive bayesian* tienen las mismas etapas que los árboles de decisión en su funcionamiento:

1. Entrenamiento a partir de un conjunto de ejemplos clasificados
2. Uso del clasificador entrenado para clasificar ejemplos no clasificados

La forma de entrenamiento es la siguiente. Dada una transacción t con un conjunto de atributos $A = \{a_1, a_2, \ldots, a_n\}$, el clasificador *naive bayesian* lo clasifica como fraudulento si

$$P(fraude|t) > P(legitimo|t)$$

o lo que es equivalente

$$P(fraude|t) > 0{,}5$$

dado que

$$P(fraude|t) + P(legitimo|t) = 1$$

Según el teorema de Bayes (ver capítulo 6 de Mitchell [31]), estas probabilidades se calculan como

$$P(v|t) = \frac{P(t|v)P(v)}{P(t)}$$

donde $v \in \{fraude,\ legitimo\}$.

El clasificador *naive bayesian* asume que los atributos de t son independientes entre sí, por lo tanto calcula $P(t|v)$ como el producto de las n probabilidades individuales

$$P(t|v) = \prod_{i=1}^{n} P(a_i|v)$$

Luego, estimando las probabilidades condicionales de cada uno de los valores de los atributos a_i con simples técnicas de conteo sobre el conjunto de transacciones disponibles, se está en condiciones de calcular la probabilidad $P(t|v)$ para futuros ejemplos.

Si el supuesto de independencia no se cumple, la aproximación de $P(t|v)$ puede tener errores importantes, lo cual podría causar que la clasificación sea inadecuada. Sin embargo, *naive bayesian* ha demostrado un desempeño sorprendentemente bueno, aún cuando los atributos no son independientes entre sí, como indican Hand y Yu [24] o Domingos y Pazzani [25].

Este mecanismo de aprendizaje tiene la particularidad que no realiza ningún tipo de búsqueda sobre los ejemplos, como hacen los árboles de decisión, sino que se limita a contar la ocurrencia de los valores de cada atributo en los casos de fraude y legitimidad. En consecuencia el algoritmo es lineal en la cantidad de ejemplos. Por otra parte, también utiliza todo el conjunto para entrenar, por lo cual tiene características similares a los árboles de decisión, en cuanto a que tolera el ruido en los datos y debe reentrenarse completamente ante la aparición de nuevos atributos o ejemplos. Además su evaluación es muy simple y computacionalmente tan eficiente como la evaluación de reglas.

El objetivo en esta fase es aprender a determinar un valor numérico de riesgo, donde el umbral para clasificar un ejemplo como fraudulento podría ser distinto de 0.5. Por lo tanto la salida necesaria del algoritmo es directamente la estimación de $P(fraude|e)$ en lugar del $v \in \{fraude, legitimo\}$.

3.3. Estimación del umbral de activación

Luego de haber calculado el puntaje de riesgo, resta definir cómo estimar un umbral adecuado. El valor elegido es el que optimiza la medida F_α, la cual se define en la sección 3.4. Se decidió utilizarla porque combina precisión y cobertura en una única medida, y además brinda la posibilidad de ponderar una sobre otra. Es importante destacar que, dadas las características del negocio, es fundamental tener alta cobertura (capturar la mayor parte del fraude posible), aún a costa de tener una baja precisión (generar muchas falsas alertas).

El objetivo del cálculo de puntaje de riesgo y su comparación contra el umbral de activación es refinar la predicción de las reglas, con el objetivo de mejorar su precisión (que suele ser muy baja). Modificando el valor de α se logra que en el cálculo de la medida F_α pese más la cobertura o la precisión. En particular, luego de algunas pruebas decidimos utilizar un $\alpha = 0{,}05$, con el que la cobertura pesa 20 veces más que la precisión. Como posible mejora a este trabajo, puede ser interesante implementar una forma automática de estimar el valor de α o utilizar otra estrategia para estimar el umbral adecuado.

Por último, resta definir qué hacer con una transacción que es evaluada como fraudulenta por varias reglas, pero luego de aplicar sus respectivos *naive bayesian* y umbrales se obtienen predicciones contradictorias. De acuerdo a la semántica definida por el *Detection Engine*, en este caso se clasifica una transacción como fraudulenta si al menos un *naive bayesian* con su respectivo umbral la clasifica como fraudulenta.

3.4. Evaluación de clasificadores

Dado un clasificador es importante estimar el desempeño que tendrá al clasificar ejemplos no utilizados en su entrenamiento. Para esto, se lo utiliza para clasificar un conjunto de ejemplos ya clasificados pero no usados en su entrenamiento, conocido como conjunto de evaluación. Al comparar las predicciones obtenidas con la clasificación real del ejemplo se pueden obtener varias medidas numéricas que dan una idea del desempeño del clasificador.

Las medidas comúnmente usadas en la bibliografía son genéricas, en el sentido de que se basan exclusivamente en la cantidad de errores y aciertos obtenidos por el clasificador al evaluar el conjunto de evaluación. Por esto, uno de los resultados esperados del proyecto es una propuesta de medidas específicas para el problema de la detección de fraude. Por otra parte, el *Detection Engine* tiene un funcionamiento en fases (evaluación de reglas, puntaje de riesgo y umbral), al que llamaremos modelo de detección. También es importante considerar cómo evaluar el funcionamiento de cada fase del modelo de detección por separado.

En esta sección se describen varias medidas utilizadas regularmente para estimar el desempeño de un clasificador. Además se sugieren otras medidas, tanto particulares para la detección de fraude como específicas para el funcionamiento del *Detection Engine*. Estas medidas no aparecen en la bibliografía, lo cual representa una de las contribuciones de este proyecto. El capítulo 5 presenta el desempeño logrado por los clasificadores desarrollados según las técnicas de Aprendizaje Automático descritas en este capítulo, evaluados contra distintos conjuntos de datos y según todas las medidas discutidas en esta sección.

El resto de esta sección se organiza como sigue. La subsección 3.4.1 presenta las medidas utilizadas normalmente para evaluar clasificadores. La subsección 3.4.2 aplica estas medidas sobre distintos conjuntos de transacciones para evaluar el desempeño de cada fase del modelo de detección por separado. La subsección 3.4.3 describe cómo evaluar la variación de desempeño del modelo de detección al variar los umbrales de detección mediante gráficas de ROC, de uso común en la bibliografía. La subsección 3.4.4 propone otras medidas para estimar cuánto tarda un modelo de detección en descubrir una cuenta fraudulenta. La subsección 3.4.5 sugiere medidas adicionales para estimar cuánto dinero se recupera y cuántas cuentas fraudulentas se descubren mediante un cierto modelo de detección. Por último, la sección 3.4.6 hace un resumen de todas las medidas descritas.

3.4.1. Medidas genéricas

En problemas de clasificación donde la clase objetivo sólo tiene dos valores posibles se usan los términos «positivo» y «negativo» (P y N). En el caso de

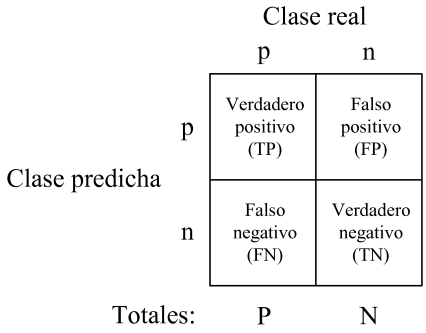

Figura 3.3: Matriz de confusión

la detección de fraude, los ejemplos fraudulentos son positivos y los legítimos negativos. Generalmente el valor «positivo» es el que interesa encontrar.

Dado un clasificador y un ejemplo, hay cuatro resultados posibles según si el ejemplo es positivo o negativo y el valor predicho por el clasificador:

- Ejemplo positivo clasificado como positivo
  Cuenta como verdadero positivo (*true positive* o TP).
- Ejemplo positivo clasificado como negativo
  Cuenta como falso negativo (*false negative* o FN).
- Ejemplo negativo clasificado como negativo
  Cuenta como verdadero negativo (*true negative* o TN).
- Ejemplo negativo clasificado como positivo
  Cuenta como falso positivo (*false positive* o FP).

Al evaluar un conjunto es común utilizar los cuatro contadores acumulados sobre cada uno de los ejemplos para construir una matriz de 2×2, que resume el comportamiento del clasificador respecto al conjunto. A esta matriz se le llama «matriz de confusión» y es la base de muchas medidas comunes. La estructura de la matriz de confusión se muestra en la figura 3.3.

En general, en Aprendizaje Automático se usan las siguientes medidas para estimar el desempeño de un clasificador:

$$exactitud = \frac{TP + TN}{P + N}$$

$$precision = \frac{TP}{TP + FP}$$

$$cobertura = \frac{TP}{P}$$

$$medidaF_1 = \frac{2 \times \text{precision} \times \text{cobertura}}{\text{precision} + \text{cobertura}}$$

$$medidaF_\alpha = \frac{(1 + \alpha) \times \text{precision} \times \text{cobertura}}{\text{precision} + \alpha \times \text{cobertura}}$$

Figura 3.4: Medidas genéricas de evaluación de clasificadores

- Exactitud (*accuracy*)
  Proporción global de aciertos sobre el total.
- Precisión (*precision*)
  Proporción de aciertos sobre las predicciones de «positivo».
- Cobertura (recuperación, efectividad o *recall*)
  Proporción de casos positivos encontrados.
- Medida-F (*F-measure*)
  Combina la precisión y cobertura en una sola medida. Depende de un parámetro α, el cual se puede usar para dar más peso a la cobertura o a la precisión.

Estas medidas se calculan a partir de los componentes de la matriz de confusión según las ecuaciones de la figura 3.4.

Idealmente, un clasificador es, a la vez, exacto, preciso y con buena cobertura. Una explicación más detallada de estos conceptos, así como también de varias medidas posibles, se puede ver en el reporte técnico de Fawcett acerca de evaluación de clasificadores [26]. Un ejemplo de matriz de confusión se presenta en la figura 3.5.

Estas herramientas permiten cuantificar los resultados esperados del proyecto. ¿Qué porcentajes de cobertura y precisión es razonable esperar? La detección de fraude es un problema muy difícil. *PayTrue Solutions* considera que es importante lograr coberturas de alrededor de 80 %, aún si esto reduce la precisión de las reglas a tan poco como 2 % o 3 %. Además es importante observar que estos dos objetivos entran en conflicto, ya que en general al incrementar la cobertura se reduce la precisión y viceversa.

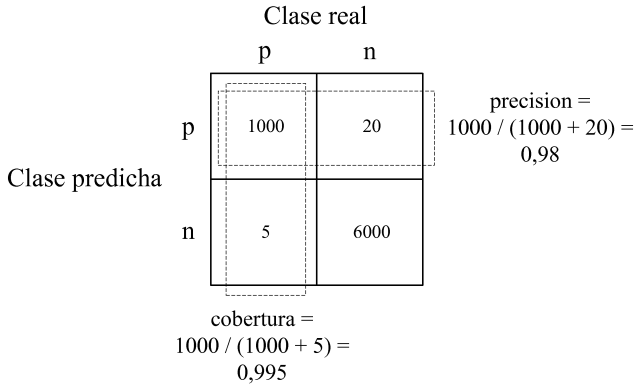

Figura 3.5: Ejemplo de matriz de confusión

3.4.2. Evaluación de cada fase

El sistema de detección de fraude utilizado por *PayTrue Solutions* (*Detection Engine*) tiene tres fases bien definidas. La comparación del umbral de activación contra el puntaje de riesgo refina la predicción realizada por las reglas en la primera fase.

Al evaluar el modelo de detección contra un conjunto de ejemplos no utilizado para entrenar, es interesante conocer el desempeño de cada elemento del modelo por separado, además del desempeño del modelo completo. Es decir, importa conocer el desempeño de:

- Cada regla r_i.
- Cada regla luego de aplicar el umbral de activación al *naive bayesian* entrenado para ella, al que llamaremos r_i+. Permite ver la mejora introducida sobre cada r_i por el refinamiento de la segunda fase.
- El modelo completo, M, o sea el conjunto de todas las reglas con sus respectivos *naive bayesian* y umbrales. Recordar que M es distinto de $\{r_i+\}$, ya que clasifica una transacción tx como fraudulenta si $\exists r_j \in \{r_i+\} \mid r_j(tx) = fraude$.
- El conjunto de todas las reglas $\{r\}$, sin calcular el puntaje de riesgo, pero considerando la misma semántica que M: clasificar una transacción tx como fraudulenta si $\exists r \in \{r\} \mid r(tx) = fraude$. A este modelo de detección le llamaremos $M-$. Permite evaluar el desempeño del conjunto de reglas.

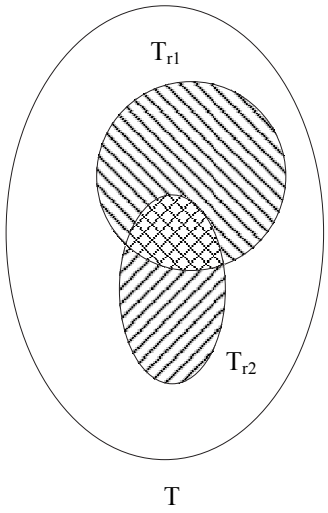

Figura 3.6: Conjuntos de transacciones cortados por cada regla

La variación de desempeño al variar los umbrales de activación se evalúa mediante gráficas de ROC, explicadas más adelante.

A su vez, es posible evaluar cada elemento del modelo de detección contra distintos conjuntos. Considérese el conjunto de todas las transacciones del conjunto de evaluación, T. Cada regla r determina un subconjunto de este conjunto, $T_r = \{tx \in T \mid r(tx) = fraude\}$. Recordar que los distintos T_r no tienen porqué ser disjuntos, ya que una transacción puede ser evaluada como fraudulenta por varias reglas, como se muestra en la figura 3.6.

Luego, al calcular las distintas medidas (precisión, cobertura, exactitud y medida F) a partir de la matriz de confusión, se pueden obtener distintos resultados según si se hace la evaluación sobre el conjunto T o T_r. Esto da lugar a las siguientes posibilidades:

- Evaluación de cada r_i sobre T
- Evaluación de cada r_i sobre T_r
- Evaluación de cada r_i+ sobre T
- Evaluación de cada r_i+ sobre T_r
- Evaluación de M sobre T
- Evaluación de $M-$ sobre T

No todas las combinaciones anteriores aportan información interesante. En particular, evaluar cada r_i sobre su respectivo conjunto T_r no aporta nueva información. En efecto, considérese la definición de $T_r = \{tx \in T \mid r_i(tx) = fraude\}$.

Por lo tanto, por definición, r_i no clasifica ninguna transacción de T_r como legítima. En consecuencia, los valores FN y TN de r_i sobre T_r son iguales a 0. Por lo tanto, además, sabiendo que (por definición)

$$P = TP + FN$$
$$N = FP + TN$$

resulta que

$$P = TP$$
$$N = FP$$

Por lo tanto, la exactitud resulta igual a la precisión. Además, la cobertura es igual a 1, como se muestra a continuación. Por definición,

$$precision = \frac{TP}{TP + FP} \qquad exactitud = \frac{TP + TN}{P + N}$$

Sabiendo que TN = 0 y P = TP,

$$exactitud = \frac{TP}{TP + FP}$$

Por definición,

$$cobertura = \frac{TP}{P}$$

Sabiendo que TP = P,

$$cobertura = 1$$

Observar que el cálculo de la precisión no se modifica. Entonces la precisión de r_i sobre T_r es igual a la precisión de r_i sobre T. Como además la cobertura de r_i sobre T_r es siempre igual a 1, resulta que no se obtiene ningún dato nuevo al evaluar r_i sobre T_r. Además, la proporción total de fraudes es

$$\frac{P}{P + N}$$

lo que es igual a la precisión de r_i sobre T_r, ya que se reduce directamente a

$$\frac{TP}{TP + FP}$$

Se puede concluir entonces que siempre que r_i tenga una precisión mayor a la proporción de fraudes de T, T_r tiene una mayor proporción de fraude que T, dato que se utilizó para definir la forma de entrenamiento del *naive bayesian*, según se explicó en la sección 3.2.

En cambio, es de particular interés comparar el desempeño de M sobre T con el de $M-$ sobre T, lo que permite tener una visión global de la mejora en el desempeño del clasificador debida a las fases 2 y 3. Esta misma visión, pero regla por regla en lugar de global, se puede lograr comparando la evaluación de cada r_i sobre T con la evaluación de cada r_i+ sobre T. La misma información, en forma simplificada dadas las características mencionadas previamente de T_r, se puede ver mediante la evaluación de cada r_i+ sobre T_r.

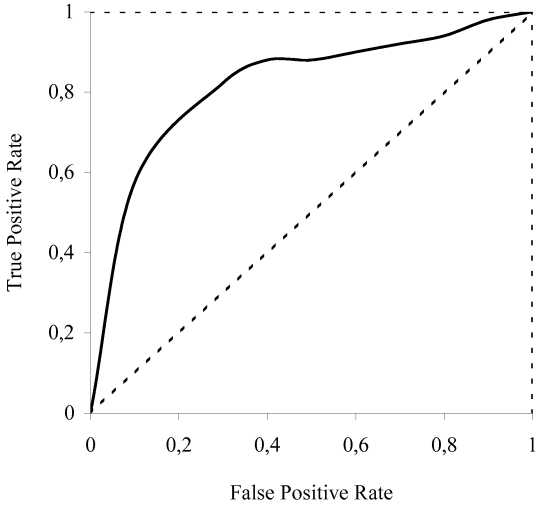

Figura 3.7: Ejemplo de gráfica de ROC

3.4.3. Gráficas de ROC al variar los umbrales

Otra medida que aporta visión acerca del comportamiento del modelo de detección es cómo varía el desempeño de cada regla al variar su correspondiente umbral. Para esto usamos una gráfica de ROC, o *Receiver Operating Characteristic*. Las gráficas de ROC son una forma alternativa de examinar el desempeño de un clasificador. Grafican el ratio de falso positivo, $\frac{FP}{N}$, contra el ratio de verdadero positivo, $\frac{TP}{P}$. El punto (0,1) es el clasificador perfecto: clasifica todos los casos positivos y negativos correctamente; es (0,1) porque el ratio de falso positivo es 0 (nada) y el de verdadero positivo es 1 (todo). El punto (0,0) representa un clasificador que predice que todos los ejemplos son negativos, mientras que el punto (1,1) corresponde a un clasificador que predice que todos los ejemplos son positivos. El punto (1,0) es el clasificador que se equivoca en todas sus predicciones. Los clasificadores aleatorios se ubican sobre la diagonal de la gráfica.

Al aumentar el valor del umbral, se incrementa el verdadero positivo pero al costo de incrementar el falso positivo. Cada valor del umbral produce un punto $(\frac{FP}{N}, \frac{TP}{P})$; al variar el valor del umbral entre 0 y 1 se puede hacer la gráfica de ROC completa. Las gráficas de ROC tienen las siguientes propiedades (ver el reporte técnico de Fawcett [26]):

- Son independientes de las proporciones de ejemplos negativos y positivos
- Cada punto de la gráfica encapsula la información completa de la matriz de confusión correspondiente a un valor del umbral

- Dan una herramienta visual para examinar el balance entre el falso positivo y el verdadero positivo

En definitiva, las gráficas de ROC permiten comprender claramente cómo varían el verdadero y falso positivo al variar el valor del umbral. La figura 3.7 muestra un ejemplo de este tipo de gráfica. A partir de esta interpretación geométrica del desempeño de un clasificador es posible definir un criterio sencillo para determinar el mejor valor del umbral: aquel que determine el clasificador más cercano (según la distancia euclidiana) al punto (0,1). Este criterio es válido y es de uso común. Sin embargo, tiene la desventaja de que no permite ponderar los distintos tipos de error (FP o FN). En cambio, la medida F_α usada por nosotros nos permite ponderar la cobertura sobre la precisión. Otra posibilidad podría haber sido utilizar este criterio geométrico pero con un cálculo de distancia euclidiana que pondere el ratio de verdadero positivo sobre el ratio de falso positivo.

3.4.4. Velocidad de detección

Otra medida que interesa conocer acerca del modelo es qué tan rápido se detecta el comienzo de actividad fraudulenta en una cuenta. Es decir, quizás un cierto modelo tenga excelente cobertura de cuentas, pero si en lograr esto tarda tanto que le da tiempo al fraudador de agotar todo el disponible de la cuenta no sirve de nada. En el caso general el disponible no se conoce; sólo el emisor conoce los disponibles de sus tarjeta habientes. No obstante, se pueden dar indicadores interesantes para estimar qué tan rápido se detecta el comienzo de la actividad fraudulenta. Las transacciones fraudulentas entre el primer fraude de una cuenta y la primer alerta de la cuenta (incluyendo la alerta) son pérdida para la entidad financiera. Las transacciones siguientes a la primer alerta se podrían haber evitado al bloquear la cuenta luego de la primer alerta. Estos conceptos se ejemplifican en la figura 3.8, donde las transacciones legítimas y fraudulentas se indican con una L y F respectivamente, y las alertas generadas con una A. Luego se pueden calcular los siguientes indicadores acerca de las pérdidas de la entidad financiera:

- Cantidad de transacciones fraudulentas entre el primer fraude de una cuenta y la primer alerta generada (vale 3 en el ejemplo de la figura 3.8).
- Suma de los montos de transacciones, $\sum pne.monto$, siendo pne una transacción cuya pérdida no fue evitada, según se mostró en la figura 3.8.

A los efectos de simplificar este cálculo ante la presencia de transacciones realizadas en distintas monedas, los montos se considerarán siempre consolidados a dólares.

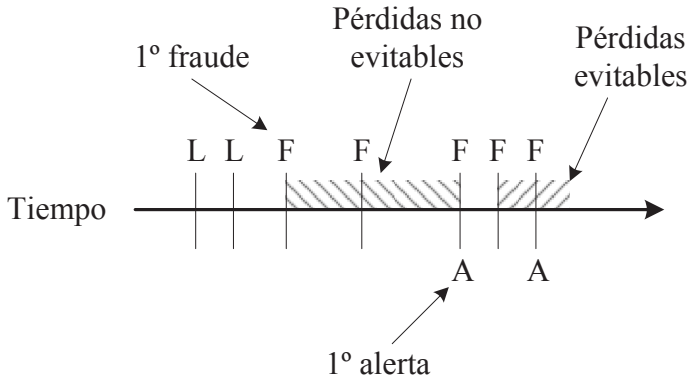

Figura 3.8: Secuencia de fraude

3.4.5. Cobertura de cuentas y montos

Un punto importante a considerar es que la cobertura se puede medir tanto sobre las transacciones como sobre las cuentas. Esta distinción es relevante. Considérese por ejemplo que podrían existir 100 transacciones fraudulentas de 6 cuentas, donde en realidad 80 de las transacciones pertenecen a la misma cuenta c, y las restantes 20 transacciones se reparten equitativamente entre las 5 cuentas restantes. Un modelo de detección que detecte sólo las 80 transacciones de c tiene 80 % de cobertura sobre transacciones, pero sólo 17 % de cobertura sobre cuentas.

Desde el punto de vista del negocio son importantes ambas coberturas. La cobertura de cuentas es importante para reducir las pérdidas por fraude de la entidad financiera. La cobertura de transacciones es importante para descubrir los patrones de fraude que provocan estas pérdidas.

Por otra parte, las medidas tal cual fueron dadas previamente están basadas en cantidades de transacciones. Sin embargo, el problema del fraude es económico. Es de interés contar con una evaluación de cuánto dinero se ahorra al aplicar un cierto modelo de detección. Por lo tanto estimaremos también la cobertura (de transacciones) basada en el monto en lugar de la cantidad de los fraudes. La versión natural de este cálculo, hecha a partir de la fórmula de cobertura, es

$$cobM = \frac{\sum TP.monto}{\sum P.monto}$$

Sin embargo, según lo discutido en la subsección anterior, el monto recuperado es más que $\sum TP.monto$, ya que los montos de todas las transacciones fraudulentas posteriores a la primer alerta son recuperados, sin importar si se generaron alertas para ellas o no. En consecuencia la cobertura de montos se calcula como

$$cobM = \frac{\sum P.monto - \sum pne.monto}{\sum P.monto}$$

donde $\sum pne.monto$ fue definido en la subsección anterior.

3.4.6. Medidas de evaluación

Resumiendo, al evaluar el desempeño del modelo de detección sobre un cierto conjunto de datos daremos las siguientes medidas:

- Precisión y cobertura de cada regla calculadas contra el conjunto de evaluación total
- Precisión y cobertura de cada regla luego de aplicar su *naive bayesian* y umbral, calculadas contra el conjunto de evaluación total y también contra el conjunto T_r
- Gráfica de ROC de cada regla con su *naive bayesian* al variar el valor de su umbral, calculadas contra el conjunto de evaluación total
- Precisión y cobertura globales del modelo de detección completo contra el conjunto de evaluación total
- Promedio, desviación estándar e histograma de las siguientes medidas tomadas sobre las transacciones fraudulentas entre la primer transacción fraudulenta de una cuenta y la primer alerta generada:
  - Cantidad
  - Suma de montos
- Cobertura de montos de fraude (del modelo de detección completo contra el conjunto de evaluación total), calculada como

$$cobM = \frac{\sum P.monto - \sum pne.monto}{\sum P.monto}$$

- Cobertura de cuentas fraudulentas, calculada como

$$cobC = \frac{\#(cuentas\ fraudulentas\ detectadas)}{\#(cuentas\ fraudulentas\ totales)}$$

Capítulo 4

Herramientas desarrolladas

En el capítulo anterior se discutió cómo resolver el aprendizaje de una configuración del *Detection Engine* (reglas, cálculo de puntaje de riesgo y umbrales de activación) a partir de un conjunto de transacciones marcadas como legítimas o fraudulentas. Para resolver este problema se aplicaron las técnicas de Aprendizaje Automático de árboles de decisión y *naive bayesian*. A su vez, se discutieron varias medidas posibles para estimar el desempeño de una cierta configuración del *Detection Engine*.

La aplicación de las técnicas mencionadas de Aprendizaje Automático requiere seguir un proceso de varias etapas, cuya salida consiste en una configuración completa del *Detection Engine* y una estimación de su desempeño, como se describe en la sección 4.1. Seguir estas etapas no es trivial, sino que se requiere un esfuerzo importante de preparación de los conjuntos de datos necesarios, archivos requeridos por las implementaciones utilizadas de las técnicas de Aprendizaje Automático, y evaluación de una configuración contra un conjunto de transacciones marcadas. Por este motivo, como parte del proyecto se desarrollaron varias herramientas, orientadas a facilitar estas tareas. La figura 4.1 resume el proceso de análisis y la interacción entre las herramientas.

Este capítulo describe brevemente las herramientas desarrolladas. Para más detalles, ver la documentación del sistema [39] o las guías de uso de cada una de las herramientas [41, 40, 47, 45, 46, 42, 44, 43]. El resto de esta sección se organiza como sigue. La sección 4.1 describe brevemente el propósito de cada herramienta y la forma en que se relacionan entre sí. La sección 4.2 describe el funcionamiento de cada herramienta en forma más detallada, indicando sus objetivos y características concretos. Por último, la sección 4.3 describe el plan de verificación utilizado para comprobar el correcto comportamiento de las herramientas.

4.1. Vista global de las herramientas

El punto de partida del proceso de análisis es una tabla de una base de datos relacional con transacciones marcadas como legítimas o fraudulentas, que llamaremos `tx.original`. De acuerdo a lo discutido en el capítulo anterior, en la sección 3.1, es necesario realizar varios tipos de transformaciones sobre los datos de `tx.original` (cálculo de atributos complejos, eliminación de atributos innecesarios, etc.). Sería ideal contar con una herramienta que sugiera qué atributos complejos son necesarios; sin embargo, llegamos a la conclusión de que al momento el estado del arte de Aprendizaje Automático no cuenta con enfoques capaces de resolver este problema (ver subsección 3.1.1). De todas formas, bautizamos una posible herramienta que sugiera atributos complejos como `aggs`.

La tarea de calcular atributos complejos a partir de los atributos de `tx.original` es compleja, por lo que consideramos necesario contar con una herramienta que los calcule. Esta herramienta, a la que llamaremos `dprep`, es capaz de tomar un conjunto de transacciones de `tx.original` y calcularles las transformaciones que se consideren necesarias. Estas transformaciones se describen en un archivo de configuración. De existir, la función de `aggs` sería generar este archivo. El resultado de aplicar `dprep` a `tx.original` es un conjunto de transacciones transformadas al que llamaremos `tx.trans`.

A partir de `tx.trans` se está en condiciones de pasar al análisis de estos datos con el objetivo de obtener sugerencias para la configuración del *Detection Engine*. Para esto definimos dos herramientas, `rlearn` y `slearn`. Por un lado, `rlearn` está basada en algoritmos de árboles de decisión y puede sugerir reglas. En cambio, `slearn` está basada en un algoritmo *naive bayesian* y puede sugerir la configuración de puntaje de riesgo y umbrales. Ambas aprovechan la configuración de `dprep` para extraer la *metadata* que requieren para acceder a `tx.trans`.

El analista de riesgo querrá evaluar estas sugerencias, así como también reglas obtenidas por otros medios, para estimar su desempeño. A su vez puede desear experimentar con diversos ajustes manuales sobre las sugerencias del *Smart Analyzer*. Por lo tanto requiere una herramienta que le permita estimar el desempeño de una cierta configuración del *Detection Engine*. A esta herramienta le llamamos `sbox`. Permite evaluar una configuración del *Detection Engine* contra `tx.trans`, obteniendo como resultado las métricas descritas en la sección 3.4. El resultado de esta evaluación puede visualizarse en forma amigable mediante la herramienta `mview`.

En caso de que el analista de riesgo haya obtenido varias configuraciones del *Detection Engine* y desee unirlas para generar una configuración unificada, puede hacerlo mediante la herramienta `mmerge`. Esta posibilidad se discute en la sección 5.1.

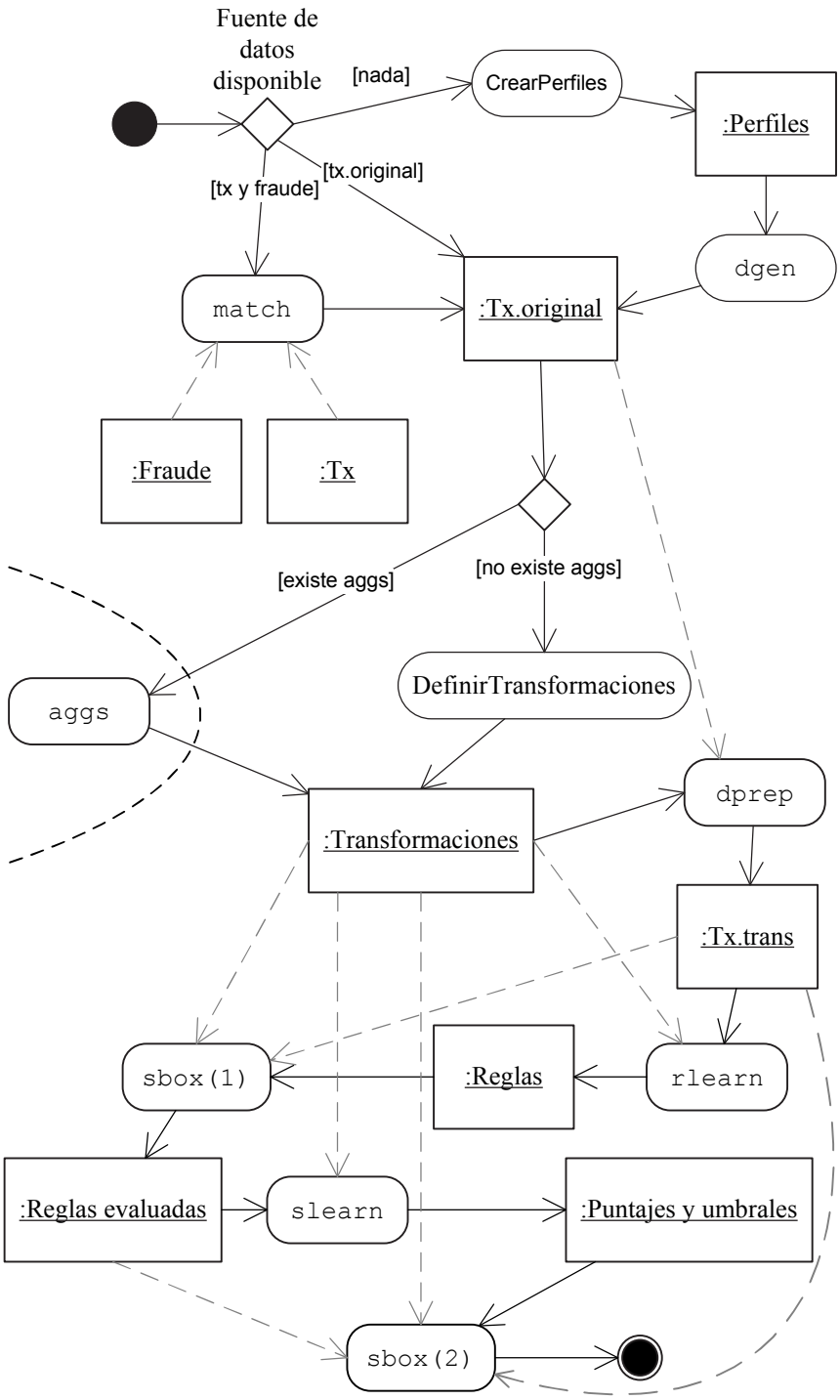

Figura 4.1: Herramientas desarrolladas y sus relaciones

Es posible que, en una entidad financiera que nunca pasó por una etapa de control de fraude, el analista de riesgo no tenga disponible `tx.original`. Si en lugar de esto tiene acceso a una fuente de datos separada que tiene sólo las denuncias de fraude, debe dedicarse a generar `tx.original` a partir de sus transacciones no marcadas y sus denuncias de fraude. Para facilitar esta tarea puede usar la herramienta `match`.

Es posible también que ni siquiera exista una base con datos de fraude, pero de todas formas se desee poder utilizar las herramientas para comprobar su funcionamiento. Al encontrarnos en esta situación desarrollamos la herramienta `dgen` que permite generar `tx.original` a partir de perfiles configurados en archivos XML. Cabe destacar que esta herramienta no es parte del uso normal del *Smart Analyzer*, y de hecho no es del interés de *PayTrue Solutions*. Fue desarrollada para paliar la carencia de datos de prueba.

En definitiva, este conjunto de herramientas son independientes pero se comunican entre sí a través de fuentes de datos comunes, como lo son `tx.original`, `tx.trans` y la configuración de `dprep`. Cada una tiene su lugar en el proceso de análisis. Este proceso, junto con las herramientas que intervienen en cada paso, se resume en la figura 4.1. Las herramientas `mview` y `mmerge` no se muestra en el diagrama de forma de reducir su complejidad. El uso de `mview` ocurre luego de finalizado el proceso de análisis, para visualizar sus resultados, mientras que el de `mmerge` es una alternativa a utilizar antes de la actividad `sbox(2)`.

Como se mencionó previamente, no desarrollamos la herramienta `aggs` por lo que aparece apartada de las demás.

4.2. Herramientas

4.2.1. `dgen`

`dgen` es una herramienta de consola que permite generar transacciones que pueden ser usadas como `tx.original` para comenzar un proceso de análisis. Para esto toma como entrada un conjunto de archivos de configuración que describen patrones de comportamiento de un tarjeta habiente.

Los archivos de configuración tienen una flexibilidad tal que permiten simular comportamientos reales en los datos generados. `dgen` usa dos tipos de archivo: un archivo de configuración global, en XML, y muchos archivos de patrones de compra, en texto plano.

El archivo de configuración global dirige la generación de datos, para lo que consta de dos partes. En la primera parte se define la cantidad N de transacciones a generar, un conjunto de números de tarjeta $\{T\}$ y una distribución d. En la segunda parte se declara una lista de perfiles $\{P\}$; cada uno de ellos

1. Para i de 1 a N
   a) Elegir aleatoriamente $t \in T$ según la distribución d
   b) Para cada perfil $p \in P$
      1) Elegir aleatoriamente c según la distribución discreta definida en p
      2) Generar c transacciones siguiendo la configuración de p, que indica cómo calcular un valor para cada atributo de una transacción

Figura 4.2: Seudocódigo de `dgen`

está acompañado de una distribución discreta que define la probabilidad de que se le generen c transacciones. La definición concreta de cada perfil está en un archivo de texto aparte. `dgen` genera datos de acuerdo al seudocódigo de la figura 4.2.

En base al análisis que hicimos de las características del negocio generamos varios juegos de archivos de configuración para generar conjuntos típicos de transacciones.

Usamos esta herramienta principalmente para generar datos de simulación. Esto nos permitió ir enfrentando paulatinamente obstáculos y limitaciones de cálculo y procesamiento de los algoritmos, antes de comenzar las pruebas con los datos reales provistos por *PayTrue Solutions*. En este sentido, cumplió con el objetivo de la simulación, dejando la validación del enfoque y de los resultados a cargo de las pruebas con los datos reales.

En un ambiente real, `dgen` puede servir para que los analistas de riesgo puedan crear escenarios de prueba. De esta forma, se puede usar para generar patrones de comportamiento específicos o emergentes dentro de los datos reales.

Para más detalles véase la guía de uso de `dgen` [40], que incluye ejemplos de configuración.

4.2.2. `aggs`

El objetivo de esta herramienta es sugerir atributos complejos que convenga agregar a `tx.original` para utilizar en la generación de reglas. Como se explicó en el capítulo anterior (sección 3.1.1), no hemos encontrado ningún enfoque que permita atacar este problema. Por esta razón no hemos implementado esta herramienta, la cual queda para futuras extensiones a este proyecto, como se discute en el capítulo 6.

Para suplir la falta de esta herramienta el analista de riesgo puede recurrir a otras fuentes de conocimiento, como pueden ser:

- Su propia experiencia en detección de fraude
- Reportes estadísticos que revelen mayor concentración de fraude en ciertas áreas
- Recomendaciones de otras entidades financieras
- Sucesivos ciclos de experimentación con distintas posibilidades, eliminando aquellos que `rlearn` no identifique como relevantes

Los atributos complejos que surjan de este tipo de consideraciones pueden agregarse a `tx.trans` usando `dprep` para luego seguir con el resto del proceso de análisis. Recuérdese que los árboles de decisión son buenos selectores de atributos relevantes. Si `rlearn` no sugiere ninguna regla que use los nuevos atributos entonces se puede concluir que no son relevantes para la detección de fraude.

4.2.3. `dprep`

`dprep` es una herramienta de consola que permite transformar el conjunto de transacciones `tx.original` en el conjunto transformado `tx.trans`, a partir del cual se puede entrenar un clasificador para detección de fraude. Para esto, `dprep` toma como entrada dos fuentes de datos:

- Un conjunto `tx.original` de transacciones marcadas como fraudulentas o legítimas. `dprep` lee esta información de una tabla en una base de datos SqlServer.
- Un archivo de configuración definiendo transformaciones a aplicar sobre las transacciones, que lee de un archivo en XML. Este archivo también contiene *metadata* para describir la estructura de `tx.original`.

Respecto a las columnas de `tx.original`, se exige que las mismas incluyan como mínimo una clase objetivo (que indica si son fraudulentas o legítimas), la fecha, el monto de la transacción y el identificador de la cuenta o tarjeta correspondiente.

En cuanto a las transformaciones, pueden ser de distintos tipos, según se describe a continuación. La necesidad de las mismas fue discutida en la sección 3.1.

Conversión de tipo Transforma el tipo de una columna de `tx.original` a un tipo distinto en `tx.trans`.

Clasificación de columnas Indica si la columna es la fecha, el monto, el identificador de cuenta o la categoría objetivo en `tx.trans`.

Tabla destino Indica el nombre de la tabla donde guardar `tx.trans`.

Filtros Indica condiciones de filtrado sobre `tx.original`, para que no todas sus tuplas aparezcan en `tx.trans`.

Copiado de columnas Indica que una columna de `tx.original` se copia sin modificaciones en `tx.trans`, opcionalmente modificando su nombre.

Split de columnas Separa una columna de `tx.original` en varias columnas de `tx.trans`. Por ejemplo, separar una fecha en cuatro columnas con el día, mes, año y hora.

Comparación de columnas Crea una columna en `tx.trans` que se calcula a partir de dos columnas de `tx.original` o de una columna comparada contra una constante aplicando alguna expresión.

Acumuladores Crea una columna en `tx.trans` que resulta de aplicar una función agregada (simple o compuesta) a algún dato de `tx.original` de aquellas tuplas que cumplen ciertas condiciones. Sirven para crear atributos de nivel 2, según la clasificación descrita en la subsección 2.4.1.

Ventanas Crea una columna en `tx.trans` que resulta de comparar algún valor de una tupla de `tx.original` contra el mismo valor de una tupla previa (en el tiempo) de la misma tabla cumpliendo alguna condición de filtrado, por ejemplo que sea de la misma tarjeta. Sirven para crear atributos que aproximen el nivel 3, según la clasificación descrita en la subsección 2.4.1.

Sampling Hace que `dprep` agregue una columna con un valor aleatorio que puede servir para elegir un subconjunto de `tx.trans` al azar.

Por detalles de otras funcionalidades y opciones de configuración, ver la guía de uso de `dprep` [41].

4.2.4. `rlearn`

`rlearn` es una herramienta de consola que ejecuta un programa de generación de reglas a través de árboles de decisión. El resultado queda en un archivo ASCII. El formato de este archivo varía según el algoritmo utilizado. La herramienta `sbox` es capaz de procesar directamente la salida de todos los programas integrados con `rlearn` (`C4.5` y `QUEST`). `rlearn` toma como entrada dos fuentes de datos:

- Un conjunto de transacciones marcadas como fraudulentas o legítimas, que lee de una tabla en una base de datos. Usualmente es el conjunto `tx.trans` transformado mediante `dprep`.

- El archivo de configuración de `dprep` usado para generar la tabla. Si no se generaron los datos mediante `dprep`, de todas formas es necesario contar con este archivo, el cual se puede crear de forma que describa la estructura de las transacciones.

`rlearn` acepta varios parámetros para modificar su comportamiento. Entre ellos destacamos el parámetro `/a`, que define el programa de árboles de decisión a utilizar. Este parámetro puede tomar los valores `c` o `q`, para indicar que se use `C4.5` o `QUEST` respectivamente. Ambos programas funcionan a partir de archivos de datos en formato ASCII, que `rlearn` genera antes de invocarlos.

Por detalles del resto de los parámetros, ver la guía de uso de `rlearn` [45].

4.2.5. `sbox`

`sbox` es una herramienta de consola que permite evaluar el desempeño de una configuración del *Detection Engine* sobre una cierta base de datos de transacciones `tx.trans`. Tiene dos modalidades de ejecución:

Evaluación de reglas Evalúa el desempeño de un conjunto de reglas, que puede estar en el formato de cualquiera de los programas de generación de árboles de decisión soportados por `rlearn`. Esta modalidad aparece como `sbox(1)` en el diagrama de la figura 4.1.

Evaluación del modelo completo Evalúa el desempeño del modelo completo de detección de fraude (reglas, puntaje de riesgo y umbral de activación). Esta modalidad aparece como `sbox(2)` en el diagrama de la figura 4.1.

En ambas modalidades, `sbox` toma como entrada el archivo de configuración de `dprep`, de donde obtiene la *metadata* de `tx.trans` y su ubicación física en una base de datos SqlServer. Luego evalúa el modelo dado (ya sea parcial - sólo reglas - o total) contra todas las transacciones. A partir de su salida se pueden calcular las medidas de evaluación definidas en la sección 3.4:

- Precisión y cobertura de cada regla calculadas contra el conjunto de evaluación total
- Precisión y cobertura de cada regla luego de aplicar su *naive bayesian* y umbral, calculadas contra el conjunto de evaluación total y también contra el conjunto T_r
- Precisión y cobertura globales del modelo de detección completo contra el conjunto de evaluación total

- Promedio, desviación estándar e histograma de las siguientes medidas tomadas sobre las transacciones fraudulentas entre la primer transacción fraudulenta de una cuenta y la primer alerta generada:
  - Cantidad
  - Suma de montos
- Cobertura de montos de fraude (del modelo de detección completo contra el conjunto de evaluación total)
- Cobertura de cuentas fraudulentas

4.2.6. `slearn`

`slearn` es una herramienta de consola que sugiere la configuración de las fases 2 y 3 del procesamiento de generación de alertas del *Detection Engine*. Recordar que estas fases consisten en calcular un puntaje de riesgo $p \mid p \in [0, 1]$ y luego comparar p contra un umbral. El *Detection Engine* sólo genera alertas para aquellas transacciones cuyo puntaje de riesgo p supera el umbral.

`slearn` toma como entrada el archivo de configuración de `dprep` y la tabla con las reglas evaluadas resultante de la aplicación de `sbox(1)`. Se permite filtrar las reglas que cuenten con una precisión mayor a un valor dado, quedándose así únicamente con las que hayan sido identificadas como buenas mediante el uso de `sbox`. Por más información respecto a la elección de las reglas ver la guía de uso de `slearn` [47].

`slearn` genera un clasificador *naive bayesian* para cada regla. Este clasificador se entrena sólo sobre aquellas transacciones clasificadas como fraudulentas por alguna regla. La salida de `slearn` es un archivo comprimido que contiene un archivo de texto para cada clasificador entrenado de esta manera, junto con su regla correspondiente y el umbral elegido.

El analista de riesgo puede utilizar la herramienta gráfica `mview` para modificar manualmente alguna regla, regenerar clasificadores *naive bayesian*, o modificar los umbrales si lo considera necesario. Esto puede ser interesante para dar mayor relevancia a algunos atributos o poder ajustar la relación precisión - cobertura del modelo en forma manual.

`slearn` también genera la información necesaria para construir las gráficas de ROC de cada regla con su *naive bayesian* al variar el valor de su umbral, calculadas contra el conjunto T_r. Estas gráficas pueden analizarse con la herramienta `mview`.

La efectividad del clasificador completo (reglas más cálculo de puntaje de riesgo y umbrales) se puede medir utilizando la herramienta `sbox`.

1. Dadas la tolerancia en el monto m y la tolerancia en la fecha d
2. Para cada denuncia de fraude f
   a) Seleccionar la primera transacción t que cumple que:
   - t.tarjeta = f.tarjeta
   - t.monto entre f.monto - m y f.monto + m
   - t.fecha entre f.fecha - d y f.fecha + d

   b) Marcar t como fraudulenta y f como usado

Figura 4.3: Seudocódigo del marcado automático de transacciones

4.2.7. `match`

`match` es una herramienta gráfica orientada a facilitar la preparación del conjunto `tx.original` en una entidad financiera donde las transacciones y las denuncias de fraude se registran por separado.

En una tal situación es muy difícil marcar las transacciones de `tx.original` como legítimas o fraudulentas, ya que no existe un identificador que relacione cada denuncia de fraude con su transacción respectiva. Además, como los datos se ingresan por separado, se pueden cometer errores de digitación que compliquen aún más el procedimiento.

`match` toma como entrada una tabla con transacciones sin marcar y una tabla con denuncias de fraude. Tiene dos modalidades: marcado automático y marcado manual.

La modalidad de marcado automático acepta una tolerancia en el monto y otra tolerancia en la fecha. Marca transacciones como fraudulentas de acuerdo al seudocódigo de la figura 4.3.

Es aconsejable utilizar esta modalidad en forma iterativa, subiendo paulatinamente las tolerancias desde cero hasta algún tope razonable, por ejemplo 15 dólares en el monto y 10 días en la fecha.

La modalidad de marcado manual permite recorrer todas las tarjetas fraudulentas una por una. Al seleccionar una tarjeta, despliega todas las transacciones sin marcar y todas las denuncias de fraude sin usar de la tarjeta. Se le permite al usuario marcar como fraudulenta cualquier transacción, y marcar como usada cualquier denuncia de fraude. Esta modalidad requiere que el usuario tenga un buen conocimiento del dominio para marcar las transacciones correctamente.

Las transacciones así marcadas forman un `tx.original` de buena calidad para utilizar en el proceso de análisis.

4.2.8. `mview`

`mview` es una aplicación gráfica que permite visualizar el resultado de una ejecución de la herramienta `sbox` en su segunda modalidad, es decir, evaluación del modelo completo, con reglas, puntajes de riesgo y umbrales.

Además, permite editar las reglas o modificar los umbrales de detección. Esto último se hace a través de la visualización de las gráficas de ROC.

4.2.9. `mmerge`

En el capítulo 5 se discute una sutil modificación en el proceso de análisis descrito en la sección 4.1 de este capítulo. Esta modificación consiste en crear distintos subconjuntos de `tx.trans`, con el objetivo de mejorar la calidad de los resultados obtenidos. Luego se analiza cada subconjunto por separado, desde la actividad «`rlearn`» en adelante, obteniendo así una instancia distinta del modelo de detección entrenada para cada uno de estos subconjuntos.

Al trabajar de esta forma, es necesario luego poder unir estos modelos de detección en uno solo. La herramienta de línea de comando `mmerge` provee esta funcionalidad. El modelo de detección resultado de usar `mmerge` puede evaluarse mediante `sbox` de la misma manera que cualquier otro modelo obtenido mediante `slearn`.

4.3. Verificación

Para comprobar el correcto funcionamiento de las herramientas se siguió una estrategia de «caja negra» (ver [19]), es decir que no se verificaron los componentes internos de las herramientas por separado.

Se definió un conjunto de casos de prueba para cada una de las herramientas. La ejecución de estos casos de prueba está automatizada para las herramientas de línea de comandos (`dgen`, `dprep`, `rlearn`, `sbox`, `slearn` y `mmerge`). Para las herramientas gráficas la ejecución es manual (`match` y `mview`).

Las herramientas de consola comparten la característica de ser no interactivas, lo cual simplifica su verificación. La estrategia seguida en general fue definir explícitamente, para cada caso de prueba, los datos de entrada y salida de cada herramienta. El procedimiento de verificación se puede resumir según se muestra en la figura 4.4.

En el caso de la herramienta `rlearn`, cabe destacar que parte de su salida es un archivo con reglas, generado por un programa externo de árboles de decisión (`C4.5` o `QUEST`). Por lo tanto no se verificó la correctitud de este archivo.

1. Para cada caso de prueba p
   - *a*) Ejecutar la herramienta correspondiente haciendo que sus datos de entrada sean los definidos para p
   - *b*) Comprobar que la salida de la herramienta (datos en la base de datos o en archivos de salida) sea igual a la definida en el caso de prueba p

Figura 4.4: Procedimiento de verificación

Para más detalles, incluyendo la especificación de todos los casos de prueba, ver el plan de verificación completo, que es parte de la documentación anexa [48].

Capítulo 5

Resultados obtenidos

A modo de resumen de lo discutido hasta el momento, en el presente proyecto se busca aprender reglas y calcular puntajes de riesgo que permitan detectar fraude.

Para el aprendizaje de reglas aplicamos árboles de decisión, en particular los programas `C4.5` y `QUEST`, mientras que para el cálculo de puntajes de riesgo utilizamos un clasificador *naive bayesian*. Recordar que se entrena una instancia distinta del clasificador *naive bayesian* por cada regla. El puntaje de riesgo estimado por el *naive bayesian* se compara contra un umbral, el cual elegimos optimizando la medida F. Nos referimos al conjunto de reglas, puntaje de riesgo y umbral como modelo de detección de fraude, que debe ser entrenado sobre un conjunto de transacciones clasificadas como fraudulentas o legítimas.

Este capítulo documenta los resultados obtenidos al entrenar modelos de detección de fraude sobre dos conjuntos de transacciones:

- Transacciones generadas semialeatoriamente
- Transacciones reales a las que tuvimos acceso gracias al cliente del proyecto

Los atributos que describen ambos tipos de transacciones son un subconjunto de los del protocolo ISO 8583, que se describió en la sección 2.2. Los datos generados se generan ya clasificados, mientras que las transacciones reales estaban sin clasificar, debido a que su entidad financiera aún no ha destinado recursos a combatir el fraude. Más adelante se describen las tareas realizadas para clasificar estas transacciones. Recordar además que es necesario aplicar varias transformaciones sobre estos conjuntos de datos, como se discutió en la sección 3.1.2. Por lo tanto, para entrenar el modelo de detección se debe proceder de la siguiente forma:

- Generar transacciones ya clasificadas, o clasificar las transacciones reales
- Aplicar transformaciones sobre las transacciones
- Generar reglas sobre el subconjunto de entrenamiento
- Evaluar las reglas sobre este mismo subconjunto
- Entrenar los clasificadores *naive bayesian* que calculan el puntaje de riesgo para las transacciones que son evaluadas como fraudulentas por cada regla
- Optimizar los umbrales de activación (uno por cada regla)

Luego de entrenar el modelo es necesario evaluarlo sobre ejemplos no utilizados en el entrenamiento para estimar su desempeño. De esta forma podemos validar el enfoque utilizado para resolver el problema, comparando los resultados obtenidos en la evaluación contra los esperados por el cliente. Además, al aplicar algoritmos de clasificación se pueden generar varios posibles clasificadores para el mismo problema, ya sea variando los parámetros de un algoritmo o usando distintos algoritmos. En particular, al haber utilizado `C4.5` y `QUEST` es interesante determinar cuál resulta en un modelo de detección de mejor desempeño. Para seguir el proceso anterior se pueden utilizar las herramientas implementadas en el proyecto, que fueron descritas en el capítulo 4.

El resto de este capítulo está separado en dos secciones, que describen los resultados obtenidos según estas medidas de desempeño sobre los conjuntos de transacciones reales y transacciones generadas (5.1 y 5.2 respectivamente).

En la sección 6.2 se analizan los resultados documentados en este capítulo, extrayendo varias conclusiones.

5.1. Transacciones reales

El conjunto de transacciones reales utilizadas para la evaluación del modelo de detección fue provisto por la empresa cliente. Por razones de confidencialidad, sólo se pueden dar características generales y estadísticas del conjunto, sin mencionar datos adicionales de su origen ni valores específicos. Contiene las transacciones hechas a lo largo de tres meses en un cierto país. Los atributos sensibles de las transacciones, como el número de tarjeta o fecha de vencimiento, estaban ofuscados para proteger la privacidad de la información. Recordar que estos datos son suficientes para cometer fraude, según se explicó en la sección 2.3, por lo que de ninguna manera puede ser comprometida su seguridad.

Estas transacciones no estaban clasificadas explícitamente; en lugar de esto, existían dos archivos, uno con los fraudes y otro con las transacciones. En consecuencia, en primer lugar hubo que clasificar las transacciones para poder entrenar el modelo de detección. Un análisis preliminar sobre la calidad de los

%(*cuentas*) fraudulentas	0,31 %
%(tx) fraudulentas	0,15 %
%(tx) legítimas	99,85 %

Figura 5.1: Estadísticas del conjunto de transacciones reales

datos indicó que solamente el 5 % de los fraudes se podía aparear con su correspondiente transacción original en forma automática. Esto se debe fundamentalmente a que el reporte del fraude se realiza en forma manual, sin visualización previa de la transacción original. Por lo tanto es común que los datos se ingresen incorrectamente, o al menos diferentes al original.

La figura 5.1 muestra las estadísticas de este conjunto. Obsérvese el alto sesgo del conjunto, que tiene sólo un 0,15 % de fraude. Este porcentaje se da sobre una gran cantidad de transacciones, que no podemos mencionar. Es importante discutir la posibilidad de que un programa de aprendizaje de reglas no pueda manejar conjuntos arbitrariamente grandes. Al respecto conviene considerar que el orden de ejecución de `c4.5rules` es $O(n^3)$, como indica Cohen en [23], mientras que su versión comercial, `C5` o `See5` [38], tiene mejoras específicamente en este punto. En general, según el estudio empírico de Lim *et al.*[29], la mayoría de los algoritmos de árboles de decisión requieren un tiempo $O(n \log n)$ para su entrenamiento. En cambio, el *naive bayesian* es lineal en la cantidad de ejemplos.

Por otra parte, los algoritmos en general tienen problemas con conjuntos desproporcionados. En un estudio empírico, Provost recomienda que los conjuntos de entrenamiento tengan proporciones similares de ejemplos negativos y positivos [52].

Para que el tamaño del conjunto de datos permita ejecutar un programa de árboles de decisión y sea menos desproporcionado, buscamos formas de obtener subconjuntos con menor proporción de transacciones legítimas, técnica conocida como *sampling*.

En primer lugar utilizamos el enfoque sencillo de seleccionar todas las transacciones fraudulentas y un muestreo al azar (sin reemplazo) de las transacciones legítimas. Las pruebas con subconjuntos creados con este enfoque, que tenían un porcentaje de fraude sobre el total entre 1 % y 5 %, revelaron que no es una buena solución. Al seleccionar al azar alrededor del 15 % de las transacciones legítimas se pierden demasiadas características del conjunto original, lo cual lleva al aprendizaje de reglas irreales. Por ejemplo: «si el comercio es un restaurant entonces la transacción es fraudulenta», regla aprendida debido a que al muestrear no quedó ninguna transacción legítima hecha en un restaurant.

Por otra parte, este muestreo tiene una falencia conceptual grave desde el punto de vista del negocio, ya que es a nivel de transacción. Cualquier muestreo razonable debe estar hecho a nivel de cuenta; se deben elegir las cuentas que cumplen

una cierta condición, y luego utilizar para el entrenamiento todas las transacciones de estas cuentas. Por lo tanto, es fundamental definir qué criterios de selección de cuentas o «cortes» son razonables, considerando las características del negocio.

Los «cortes» son razonables en función de que representen comportamientos diferenciados de los tarjeta habientes en relación con el fraude. En este sentido, identificamos los siguientes «cortes» posibles:

1. Cuentas con actividad fraudulenta.
2. Cuentas con transacciones en el extranjero, al que llamaremos *crossborder*.
3. Cuentas pertenecientes a ciertos productos, de forma de analizar por separado el comportamiento de las cuentas con productos destinados a mercados con distinto nivel adquisitivo.
4. Cuentas pertenecientes a distintos tipos de producto, lo cual permite, por ejemplo, analizar las transacciones de prepago separadas de las de crédito.
5. Cuentas con transacciones hechas por Internet

Recordar que el conjunto de transacciones reales estaba ofuscado, es decir, modificado para preservar la privacidad de los datos sensibles, notablemente el número de tarjeta. Como el emisor y producto se conocen a partir del número de tarjeta, no hay información suficiente para utilizar los conjuntos basados en el producto (tercer y cuarto conjuntos del listado anterior). Por otra parte, en los atributos de este conjunto faltaba el código de condición del POS, el cual permite distinguir las transacciones hechas por Internet, lo que impide utilizar el quinto subconjunto. En consecuencia sólo es posible aplicar las técnicas descritas a los conjuntos de cuentas fraudulentas y de *crossborder*. El resto de esta sección describe los resultados obtenidos sobre estos conjuntos.

El conjunto de todas las transacciones de aquellas cuentas que tienen alguna transacción fraudulenta tiene un 27,02 % de fraude. Sus estadísticas se muestran en la figura 5.2. El conjunto de *crossborder* tiene un 0,82 % de fraude. Sus estadísticas se muestran en la figura 5.3. Dado que el conjunto sigue siendo muy sesgado, en realidad utilizamos su respectivo subconjunto de cuentas fraudulentas, es decir: aquellas cuentas con transacciones en el extranjero y que además tienen alguna transacción fraudulenta. De esta forma se obtiene otro conjunto cuyas estadísticas se muestran en la figura 5.4.

A su vez realizamos un estudio manual de las características de este conjunto de transacciones para identificar la presencia de patrones de fraude conocidos. De esta forma nos fue posible comprobar si las reglas aprendidas identifican algunos de estos patrones, como una forma adicional de validar el resultado más allá de las medidas de desempeño descritas anteriormente. Los patrones identificados por el análisis manual a nivel de cuenta son los siguientes:

%(*cuentas*) fraudulentas	100 %
%(tx) fraudulentas	27,02 %
%(tx) legítimas	72,98 %

Figura 5.2: Estadísticas del conjunto de cuentas fraudulentas

%(*cuentas*) fraudulentas	0,78 %
%(tx) fraudulentas	0,82 %
%(tx) legítimas	99,18 %

Figura 5.3: Estadísticas del conjunto *crossborder* completo

1. Excesiva concentración de transacciones en un mismo día.
2. Excesiva concentración de transacciones en un mismo día dentro de un mismo rubro.
3. Excesiva concentración de transacciones en un mismo día siendo además en un mismo comercio.
4. Transacciones por montos elevados.
5. Transacciones con montos similares un mismo día.
6. Transacciones con montos por debajo del límite de piso.
7. Transacciones rechazadas en un mismo día.
8. Transacciones cuyo modo de entrada fue manual en terminales con capacidad de lectura de banda.

Para cada uno de los patrones listados se obtuvieron una o más reglas que evaluaban es tipo de actividad sospechosa. También se obtuvieron otras reglas que no se corresponden con ninguno de los patrones conocidos.

Los atributos complejos utilizados son similares a los que se muestran más adelante para las transacciones generadas. Una vez generados estos atributos complejos, se está en condiciones de comenzar con el entrenamiento. Recordar que utilizamos los subconjuntos de cuentas fraudulentas y *crossborder*, entrenando así dos modelos de detección distintos, que llamaremos M_{cf} y M_{ce} respectivamente. En ambos casos entrenamos el modelo de detección usando los dos

%(*cuentas*) fraudulentas	100 %
%(tx) fraudulentas	56 %
%(tx) legítimas	44 %

Figura 5.4: Estadísticas del conjunto *crossborder* (sólo cuentas con fraude)

```
precision = 0.10
cobertura = 0.47
cobM = 0.37
cobC = 0.53
```

```
avg(ctx) = 1.68
σ(ctx) = 1.44

avg(mtx) = 220.72
σ(mtx) = 392.05
```

Figura 5.5: Evaluación del desempeño de M_{cf}

primeros meses. Luego evaluamos el modelo de detección así entrenado sobre *todas* las transacciones del tercer mes, para estimar su desempeño sobre otros conjuntos de transacciones.

Cabe destacar que estos modelos de detección son el resultado de análisis realizados a partir de visiones parciales de la realidad. Por lo tanto, las reglas obtenidas para cada uno de ellos son complementarias. Entonces es interesante generar un modelo de detección a partir de la unión de estos dos modelos, al que llamaremos M_m.

Así como no es posible dar información detallada acerca del conjunto de transacciones, tampoco se pueden detallar las reglas obtenidas a partir de ellas ni su matriz de confusión (que revelaría las cantidades absolutas de transacciones legítimas y fraudulentas). Solamente podemos dar las medidas de desempeño evaluadas para los tres modelos de detección mencionados. Las medidas dadas corresponden, en general, a modelos de detección entrenados utilizando `C4.5` para la generación de reglas. El desempeño de los modelos entrenados utilizando `QUEST` fue siempre muy inferior, por lo que damos solamente algunas de las medidas, correspondientes al modelo M_m, al final de esta sección.

La figura 5.5 indica las medidas tomadas respecto al desempeño del modelo de detección completo entrenado a partir del subconjunto de las cuentas fraudulentas, M_{cf}. En ella, *ctx* es la cantidad de transacciones fraudulentas entre el primer fraude y la primer alerta de una cuenta (inclusive), y *mtx* es la suma de los montos de las transacciones fraudulentas entre el primer fraude y la primer alerta de una cuenta (inclusive). También se utilizan los términos *cobC* y *cobM* para denotar la cobertura de cuentas y montos, respectivamente.

Recordar que las medidas *ctx*, *mtx* y *cobM* consideran que se evitan todos los fraudes ocurridos luego de la primer alerta exitosa en cada cuenta, como se explicó en la subsección 3.4.4. Por lo tanto, el mejor *ctx* posible es 1, ya que siempre hay al menos un fraude en cada cuenta. De la misma manera, no es posible llegar a 1 en *cobM*, dado que en la suma del monto recuperado nunca se considera el correspondiente a la primer alerta exitosa.

La figura 5.6 indica las medidas tomadas respecto al desempeño del modelo de detección completo entrenado a partir del subconjunto de cuentas del extranjero, M_{ce}. Se usan *ctx*, *mtx*, *cobC* y *cobM* de la misma manera que en la figura 5.5.

```
precision = 0.16
cobertura = 0.29
cobM = 0.23
cobC = 0.26
```

```
avg(ctx) = 1.21
σ(ctx) = 0.69

avg(mtx) = 188.67
σ(mtx) = 347.21
```

Figura 5.6: Evaluación del desempeño de M_{ce}

```
precision = 0.10
cobertura = 0.50
cobM = 0.38
cobC = 0.55
```

```
avg(ctx) = 1.61
σ(ctx) = 1.38

avg(mtx) = 215.29
σ(mtx) = 362.42
```

Figura 5.7: Evaluación del desempeño de M_m

La figura 5.7 indica las medidas tomadas respecto al desempeño del modelo de detección completo obtenido al unir los modelos M_{ce} y M_{cf}, llamado M_m. Se usan *ctx*, *mtx*, *cobC* y *cobM* de la misma manera que en la figura 5.5.

La figura 5.8 muestra el histograma de *ctx* (cantidad de transacciones fraudulentas entre el primer fraude y la primer alerta de una cuenta), mientras que la figura 5.9 muestra el histograma de *mtx* (suma de los montos de las transacciones fraudulentas entre el primer fraude y la primer alerta de una cuenta).

La figura 5.10 indica las medidas tomadas respecto al desempeño del modelo de detección completo, M_m, entrenado a partir de reglas aprendidas usando `QUEST`. Se usan *ctx*, *mtx*, *cobC* y *cobM* de la misma manera que en las figuras anteriores.

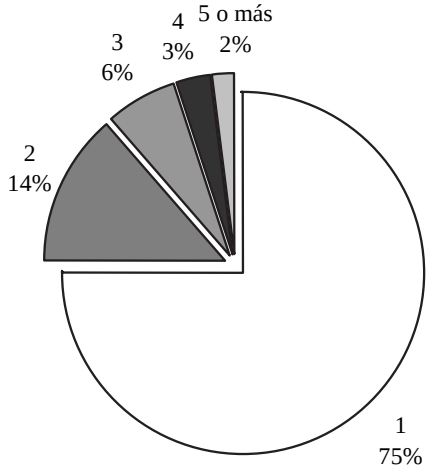

Figura 5.8: Histograma de *ctx*

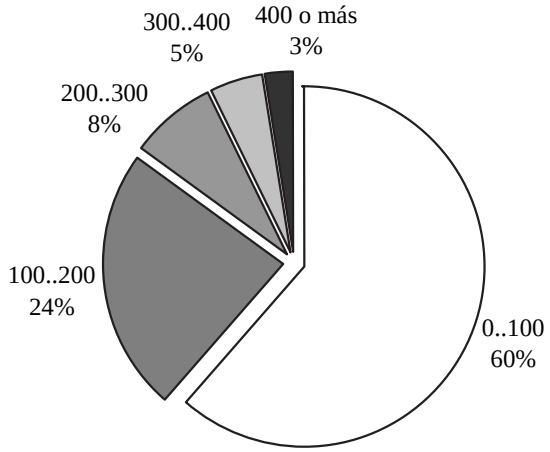

Figura 5.9: Histograma de *mtx*

```
precision = 0.05
cobertura = 0.24
cobM = 0.23
cobC = 0.22
```

```
avg(ctx) = 1.61
σ(ctx) = 1.38

avg(mtx) = 303.20
σ(mtx) = 642.07
```

Figura 5.10: Evaluación del desempeño de M_m entrenado usando `QUEST` para la generación de reglas

#(*cuentas*) legítimas	3 300
#(*cuentas*) fraudulentas	2 800 (24,24 %)
#(*tx*) aprobadas	85 708
#(*tx*) fraudulentas	13 031 (15,20 %)
#(*tx*) legítimas	76 677 (84,80 %)
#(*tx*) rechazadas	2 068

Figura 5.11: Estadísticas del conjunto de transacciones generadas

MCC	Descripción	Monto fraude	#(fraudes)
6011	Cajeros automáticos	1 547 168	5166
5072	Componentes electrónicos	213 624	466
5722	Electrodomésticos	194 302	266
7994	Videojuegos	193 031	279
5611	Tiendas de ropa	10 000	180
7995	Apuestas	9 170.66	100

Figura 5.12: Listado de 5 `MCC` con su monto perdido por fraude

5.2. Transacciones generadas

Al dudar de la disponibilidad de datos reales, inicialmente implementamos un generador de transacciones que nos permitiera realizar pruebas. Esto requirió realizar un análisis del comportamiento típico de los tarjeta habientes, para poder generar transacciones de acuerdo a tal comportamiento. Sobre este conjunto no se creó ningún «corte», sino que se analizó el conjunto completo.

El conjunto de transacciones generadas mediante la herramienta `dgen` (en forma completamente independiente a las transacciones reales) contiene las transacciones hechas a lo largo de tres meses en un cierto país. La figura 5.11 resume sus estadísticas; además, la figura 5.12 indica la distribución de fraude entre los `MCC` de las transacciones de este conjunto. En las figuras 5.13 y 5.14 se muestran las cantidades y montos de transacciones legítimas y fraudulentas por mes.

Al generar datos podemos definir atributos que no estaban disponibles en el conjunto de transacciones reales debido al ofuscamiento, lo que permite definir más atributos complejos. En particular, un atributo importante que no estaba presente en las transacciones reales es el BIN emisor, que identifica el emisor / producto correspondiente al medio de pago. A partir de este atributo se definió el atributo complejo `ISSUER_BIN_COUNT_REJECTED_DAY`.

Los datos generados presentan los siguientes patrones de fraude:

- Ensayo de `BIN`
- Ensayo de cuenta

Mes	Monto	Cantidad de transacciones
A	4 857 216	30 564
B	3 740 209	26 015
C	5 272 725	31 197

Figura 5.13: Listado de montos y cantidades de transacciones legítimas por mes

Mes	Monto	Cantidad de transacciones
A	1 574 348	30 564
B	553 341	26 015
C	2 096 072	31 197

Figura 5.14: Listado de montos y cantidades de transacciones fraudulentas por mes

- Concentración por cuenta
- Concentración por comercio
- Montos decrecientes o crecientes
- Concentración por `MCC` y por cuenta
- Autorizaciones rechazadas por montos pequeños en cajeros automáticos
- Transacciones por montos superiores a 20 000 dólares

Su generación se hizo en dos etapas. En primer lugar generamos únicamente datos legítimos, de acuerdo a patrones de uso típicos. Luego generamos datos simulando el comportamiento fraudulento para las mismas cuentas, de forma de que se presenten simultáneamente ambos tipos de comportamiento en las mismas cuentas, de la misma forma que en la realidad. Los archivos con la definición de los perfiles utilizados para la generación son entregados como parte de los recursos del proyecto.

A continuación se muestran los atributos complejos que definimos sobre este conjunto:

`DAY_OF_WEEK`**:** Calcula el día de la semana en que se realizó la transacción.

`ISSUER_CTY_EQ_MERCHANT_COUNTRY`**:** Calcula si el país del emisor es igual al país del comercio (o sea, si es internacional o no). Este atributo permitirá identificar las cuentas internacionales, que comúnmente son más riesgosas que las domésticas.

`ISSUER_BIN_COUNT_REJECTED_DAY`**:** Calcula la cantidad de autorizaciones rechazadas para un mismo tipo de producto en el día. Este atributo permitirá detectar el ensayo de `BIN`.

`SUM_PAN_AMOUNT_MONTH`: Calcula la suma del monto gastado por la cuenta en el mes corriente. Permite comparar el uso corriente del disponible de la cuenta contra un posible uso excesivo.

`PAN_COUNT_MONTH`: Calcula la cantidad de transacciones de la cuenta en el mes corriente, logrando un objetivo similar al atributo anterior.

`SUM_PAN_AMOUNT_WEEK`: Suma el monto de las transacciones de la cuenta para la semana corriente. Permite identificar un uso excesivo de la cuenta en forma semanal.

`PAN_COUNT_REJECTED_DAY`: Calcula la cantidad de transacciones rechazadas para la cuenta en las últimas 24 horas antes de esta transacción sea aprobada. Permite identificar el ensayo de cuenta o del disponible de la cuenta.

`SUM_PAN_AMOUNT_DAY`: Suma el monto gastado para la cuenta en el día actual. Permite identificar uso excesivo diario de la cuenta.

`PAN_COUNT_DAY`: Calcula la cantidad de transacciones para la cuenta en el día de la transacción actual. Permite identificar uso excesivo diario de la cuenta.

`MCC_PAN_COUNT_48HOURS`: Calcula la cantidad de transacciones de la cuenta bajo el mismo `MCC` en las últimas 48 horas. Permite identificar concentración excesiva de compras bajo un mismo rubro poco habitual en la actividad legítima.

`MERCHANT_PAN_COUNT_48HOURS`: Calcula la cantidad de transacciones de la cuenta que realizó en el mismo comercio en las últimas 48 horas. Permite identificar una concentración excesiva de compras bajo un mismo comercio, también poco habitual en la actividad legítima.

`PAN_COUNT_WEEK_SMALL_AMOUNT_FOR_CASH_WITHDRAWAL`: Calcula la cantidad de transacciones con monto inferior a 10 para retiros de efectivo en cajeros (`MCC` 6011 o 7011) en la última semana. Identifica transacciones de montos pequeños, usualmente realizadas para saber si la cuenta tiene disponible o si ha sido bloqueada, para luego efectuar una compra o retiro por un monto elevado.

`MONTHLY_AVG_PAN_AMOUNT_LAST_2MONTHS`: Monto promedio gastado para una cuenta en los últimos dos meses. Permite comparar el uso de la cuenta en períodos anteriores. Dependiendo de la cantidad de datos de entrenamiento o evaluación se pueden comparar períodos mayores como ser anuales o trimestrales.

`MONTHLY_AVG_PAN_COUNT_LAST_2MONTHS`: Cantidad de transacciones para la cuenta en los últimos dos meses.

`DISTINCT_MCC_PAN_COUNT_LAST_MONTH`: Cantidad distinta de rubros de comercio por los que pasó la cuenta en los últimos dos meses.

`DISTINCT_MCC_PAN_COUNT_3DAY`: Cantidad distinta de rubros de comercio por los que paso la cuenta en los últimos 3 días. También permite identificar actividad excesiva bajo un mismo rubro.

`MERCHANT_EQ_P1, P2, P3`: Compara, para cada una de las tres transacciones anteriores, si la compra actual fue en el mismo comercio. Permite detectar actividad excesiva en el mismo comercio.

`MERCHANT_CITY_EQ_P1, P2, P3`: Compara, para cada uno las tres transacciones anteriores, si la compra actual fue en la misma ciudad que las anteriores. Permite identificar si compras consecutivas y cercanas en el tiempo fueron realizadas en distintas ciudades.

`AMOUNT_DIFF_P1, P2, P3`: Calcula, para cada una de las tres transacciones anteriores, la diferencia de monto respecto a la transacción actual. Este atributo permite detectar patrones de comportamiento respecto a la diferencia entre montos de transacciones consecutivas. Una vez que se comete fraude, suele incrementarse el monto de cada transacción hasta agotar el disponible o hasta que la cuenta sea bloqueada.

`HOUR_DIFF_P1, P2, P3`: Calcula para cada una de las tres transacciones anteriores la diferencia en horas, minutos y segundos respecto a la transacción actual. Cuando una tarjeta es robada suele utilizarse rápidamente antes de que sea reportada. Por lo tanto es común que la diferencia en minutos de las transacciones consecutivas sea muy pequeña.

`MONTHLY_AMOUNT_USAGE_COMPARISON`: Compara el monto gastado para la cuenta en el último mes contra el último trimestre.

`MONTHLY_USAGE_COMPARISON`: Compara la cantidad de transacciones de la cuenta del último mes contra el último trimestre. Estos últimos dos atributos comparan el uso de la cuenta de períodos pasados contra el actual o no cerrado.

El modelo de detección entrenado usando estos atributos a partir de los dos primeros meses detectó todos los patrones de fraude introducidos durante la generación de datos. Esto no es sorprendente, ya que es probable que los datos generados no incluyan suficiente ambigüedad como para confundir al modelo de detección. De todas formas, comprueba que las reglas generadas permiten diferenciar los distintos comportamientos fraudulentos presentes en el conjunto de ejemplos. Consideramos que, al haber obtenido datos reales, no es relevante continuar incrementando la complejidad de los datos generados.

La figura 5.16 muestra algunas de las reglas aprendidas utilizando `C4.5`. El resto de las medidas corresponden también al modelo de detección entrenado utilizando `C4.5` para la generación de reglas.

Cabe destacar que las reglas aprendidas obtuvieron excelentes resultados según todas las medidas de evaluación definidas, al punto de que la mayoría de las

		Clase real P	Clase real N
Clase pred.	P	3843	1363
	N	21597	191

```
precision = 0.74
cobertura = 0.95
cobM = 0.78
cobC = 0.98
```

```
avg(ctx) = 1.11
σ(ctx) = 0.37

avg(mtx) = 315.92
σ(mtx) = 322.46
```

Figura 5.15: Evaluación del desempeño de M_{dgen}

reglas tienen precisión 1. En este escenario, la fase de puntaje de riesgo y umbral de activación no tiene sentido, ya que su función es mejorar la precisión. Por esto se utilizó un $\alpha = 0$, lo que significa que el umbral se ajusta de forma tal que maximice la cobertura, ya que la fórmula de cálculo de la medida F

$$medidaF_\alpha = \frac{(1+\alpha) \times \text{precision} \times \text{cobertura}}{\text{precision} + \alpha \times \text{cobertura}}$$

se reduce a

$$medidaF_0 = \text{cobertura}$$

en este caso.

La figura 5.15 indica las medidas tomadas respecto al desempeño del modelo de detección completo entrenado a partir del conjunto de datos generados, M_{dgen}. Se usan ctx, mtx, $cobC$ y $cobM$ de la misma manera que en la figura 5.5.

Es importante discutir cómo algunas de las reglas aprendidas, enumeradas en la figura 5.16, identifican los patrones introducidos con el generador de transacciones. Las reglas se identifican a partir de la columna «Ref.».

Las reglas «A» y «B» muestran un patrón de concentración de actividad en un mismo comercio para un misma cuenta con actividad cercana en el tiempo. Las reglas «C» y «D» también muestran concentración de actividad en un mismo comercio pero particularmente a nivel de retiros en cajeros automáticos. La reglas «E» y «F» identifican el patrón de ensayo de cuenta, ya que la primera detecta cuentas con más de tres transacciones en cajeros con montos pequeños, mientras que la segunda identifica cuentas que tuvieron más de tres transacciones rechazadas en el día. Las próximas dos reglas, «G» y «H», identifican cuentas con actividad riesgosa por los montos elevados de sus transacciones. «I», «J» y «K» identifican transacciones dentro de un rubro de actividad de alto riesgo (estaciones de servicio). Estas reglas presentan un uso excesivo mensual, transacciones cercanas en el tiempo, o inclusive transacciones en el mismo rubro y el mismo día pero en ciudades distintas. Cada una de las reglas mencionadas identifican actividad que, mínimamente, resulta sospechosa. La última regla identifica varias compras realizadas en restaurantes, con escaso tiempo transcurrido entre ellas, lo cual resulta también poco creíble en el uso corriente.

Ref.	Correctos	Incorrectos	Precisión	Cobertura	Medida F
(AMOUNT > 59) AND (DAY > 14) AND (DAY ≤ 23) AND (MERCHANT_PAN_COUNT_48HOURS > 0) AND (HOUR_DIFF_P1 ≤ -4)					
A	308	2	0.993	0.076	0.141
(ISS_BIN = 497326) AND (AMOUNT > 59) AND (DAY ≤ 24) AND (MERCHANT_EQ_P1 = E)					
B	134	0	1	0.033	0.064
(MCC = 7011) AND (MCC_PAN_COUNT_48HOURS > 3) AND (MERCHANT_PAN_COUNT_48HOURS > 3)					
C	131	24	0.845	0.032	0.062
(MCC = 6011) AND (MCC_PAN_COUNT_48HOURS > 3) AND (MERCHANT_PAN_COUNT_48HOURS > 3)					
D	100	40	0.71	0.024	0.047
(PAN_COUNT_WEEK_SMALL_AMOUNT_FOR_CASH_WITHDRAWAL > 3)					
E	8	0	1	0.001	0.003
(PAN_COUNT_REJECTED_DAY > 3)					
F	132	20	0.868	0.032	0.063
(ISS_BIN < 472812) AND (AMOUNT > 501) AND (DAY > 26) AND (DAY ≤ 29)					
G	343	8	0.977	0.085	0.156
(ISS_BIN = 497326) AND (AMOUNT > 501) AND (DAY > 21) AND (DAY ≤ 25) AND (SUM_PAN_AMOUNT_DAY ≤ 479) AND (MCC_PAN_COUNT_48HOURS ≤ 0) AND (HOUR_DIFF_P3 > -1150)					
H	18	0	1	0.004	0.008
(MCC = 5541) AND (AMOUNT > 40) AND (HOUR_DIFF_P1 ≤ -1)					
I	175	0	1	0.043	0.083
(MCC = 5411) AND (SUM_PAN_AMOUNT_MONTH ≤ 1071) AND (MERCHANT_CITY_EQ_P1 = D) AND (DAY ≤ 25) AND (PAN_COUNT_MONTH > 9) AND (HOUR_DIFF_P2 ≤ -6859)					
J	2	1	0.667	0.001	0.001
(MCC = 5411) AND (AMOUNT > 40) AND (AMOUNT ≤ 100) AND (SUM_PAN_AMOUNT_MONTH ≤ 2769) AND (PAN_COUNT_MONTH > 15) AND (HOUR_DIFF_P1 ≤ -1)					
K	2	0	1	0.001	0.001
(MCC = 5814) AND (DAY > 10) AND (HOUR_DIFF_P1 ≤ -1)					
L	2	1	0.667	0.001	0.001

Figura 5.16: Reglas aprendidas que identifican patrones de fraude conocidos

Capítulo 6

Conclusiones

El objetivo principal del proyecto es automatizar la configuración de un sistema de detección de fraude en transacciones de medios de pago, llamado *Detection Engine*. Esta configuración indica cuándo generar una alerta para que los analistas de riesgo estudien el caso. El *Detection Engine* analiza cada transacción que llega a la entidad financiera, siguiendo los siguientes pasos:

1. Evaluar un conjunto de reglas
2. Si alguna regla evalúa la transacción como posiblemente fraudulenta, calcular un puntaje de riesgo
3. Si el puntaje de riesgo supera un umbral de activación, generar alerta

6.1. Técnicas aplicadas

Para lograr automáticamente una configuración del *Detection Engine* aplicamos dos técnicas de Aprendizaje Automático: algoritmos de árboles de decisión, que aprenden reglas a partir de un conjunto de transacciones marcadas como fraudulentas o legítimas, y un algoritmo *naive bayesian*, que estima la probabilidad de que una transacción sea fraudulenta a partir de las transacciones así clasificadas por alguna regla. Al entrenar el *naive bayesian* se utilizan las mejores reglas aprendidas en la fase anterior, eligiendo aquellas cuya precisión supere un parámetro p. Elegimos un valor de 3 % para este parámetro, que representa la mínima precisión aceptable de una regla para el cliente.

Para determinar el umbral de activación óptimo, implementamos una búsqueda entre los posibles umbrales, eligiendo aquel que maximice la medida F_α. Esta

medida combina la cobertura y precisión en un solo valor numérico, permitiendo dar más peso a cualquiera de estos dos factores al variar el parámetro α. Luego de varias pruebas, elegimos un $\alpha = 0{,}05$, con el que la cobertura pesa 20 veces más que la precisión. Este valor se corresponde con la necesidad del cliente de lograr una buena cobertura, aún a costa de perder precisión.

Para simplificar la tarea del usuario, implementamos varias herramientas que permiten seguir el proceso completo de análisis, desde los conjuntos de ejemplos iniciales hasta obtener una configuración completa. Este proceso se puede resumir de la siguiente manera:

1. Obtener un conjunto inicial de ejemplos (transacciones clasificadas como fraudulentas o legítimas).
2. Aplicar transformaciones sobre el conjunto inicial para obtener un conjunto de transacciones enriquecidas con atributos complejos.
3. Separar el conjunto transformado en un conjunto de entrenamiento y otro de evaluación según la fecha de las transacciones. Además puede convenir elegir un subconjunto del conjunto de entrenamiento.
4. Ejecutar uno de los dos algoritmos de árboles de decisión soportados sobre el subconjunto de entrenamiento elegido, para obtener reglas.
5. Si se obtuvieron reglas a partir de un subconjunto del conjunto de entrenamiento, conviene evaluarlas sobre todo el conjunto de entrenamiento, para estimar qué tan bien generalizan ante nuevos ejemplos, lo cual da mayor confianza para la elección de las mejores reglas (siguiente punto).
6. Elegir las reglas de mejor precisión y utilizarlas para entrenar el clasificador *naive bayesian* (paso en el que también se optimiza el umbral), con lo que la configuración queda completa.
7. Evaluar la configuración completa contra el conjunto de transacciones de evaluación.
8. Analizar los resultados obtenidos en función de las diversas medidas generadas por la herramienta `sbox`.

La elección de subconjuntos del conjunto de entrenamiento es necesaria para reducir el elevado sesgo, común en el negocio, en donde el porcentaje de fraude suele estar alrededor del 0,1 %. También es importante para reducir la cantidad de ejemplos de entrenamiento, ya que los programas utilizados (`C4.5` y `QUEST`) pueden tener problemas con conjuntos masivos de ejemplos. Estos subconjuntos deben crearse según cortes razonables desde el punto de vista del negocio, de los que identificamos cinco:

1. Cuentas con actividad fraudulenta.
2. Cuentas con transacciones en el extranjero.
3. Cuentas pertenecientes a ciertos productos, de forma de analizar por separado el comportamiento de las cuentas con productos destinados a mercados con distinto nivel adquisitivo.
4. Cuentas pertenecientes a distintos tipos de producto, lo cual permite, por ejemplo, analizar las transacciones de prepago separadas de las de crédito.
5. Cuentas con transacciones hechas por Internet

Debido a carencias en la información disponible de transacciones reales, sólo pudimos crear los dos primeros subconjuntos. Por otra parte, al obtener distintas instancias del modelo de detección para cada subconjunto, es importante poder unir estas instancias de forma de obtener un modelo de detección unificado.

6.2. Resultados obtenidos

En el presente trabajo se define un proceso de análisis que permite obtener un clasificador de transacciones en fraudulentas o legítimas. Es fundamental poder estimar la calidad de este clasificador, de forma de saber si conviene aplicarlo o no. Precisamente el último paso del proceso de análisis es realizar esta estimación, lo cual se hace a través de un conjunto de medidas presentadas en la sección 3.4 de este trabajo.

La primera conclusión importante surge de comparar los resultados obtenidos entre `C4.5` y `QUEST`. Al respecto podemos decir con seguridad que `C4.5` fue ampliamente mejor que `QUEST`, obteniendo el doble de precisión y cobertura. Además, la legibilidad de las reglas generadas por `QUEST` es inferior, debido a que compara atributos contra una lista de valores posibles en un mismo nodo, lo cual resulta poco apropiado para atributos con varios cientos de valores.

Los resultados mencionados a continuación se obtuvieron siguiendo el proceso descrito en la sección anterior, de acuerdo a las siguientes observaciones:

- Se utilizó `C4.5` para la generación de reglas
- Se entrenaron dos modelos de detección, a partir de los subconjuntos de cuentas fraudulentas y de cuentas en el extranjero
- En ambos casos se utilizó $\alpha = 0{,}05$ y $p = 3\,\%$
- Se generó un modelo de detección unificado a partir de estos dos modelos

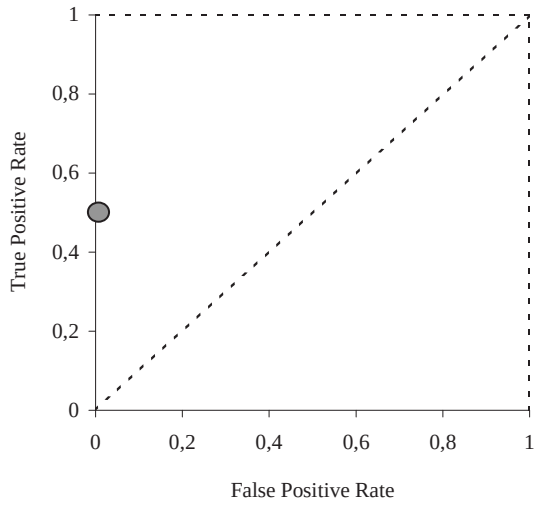

Figura 6.1: Posición del modelo de detección final en una gráfica de ROC

Acerca de los resultados obtenidos por este modelo, corresponde en primer lugar compararlos contra los obtenidos por un hipotético clasificador aleatorio. Esta comparación se puede ver en la gráfica de ROC de la figura 6.1. En este tipo de gráfica, un clasificador aleatorio puede ocupar cualquier punto sobre la diagonal. El punto ($\frac{FP}{N}$, $\frac{TP}{P}$) correspondiente al mejor clasificador entrenado, (0.006, 0.50), está sustancialmente por encima de la diagonal de la gráfica, por lo que podemos decir que son significativamente superiores a un clasificador aleatorio.

A pesar de esto, no se alcanzó la cobertura esperada por el cliente del proyecto (80 %), habiéndose logrado solamente un 50 % con el modelo final. De todas formas, se puede decir que los datos reales utilizados eran de muy baja calidad, respecto a lo que destacamos los siguientes puntos:

- Sólo el 5 % de las denuncias de fraude se pudo aparear con su correspondiente transacción en forma directa, lo cual hace dudar de la calidad de la clasificación de las transacciones utilizadas como punto de partida para el análisis.
- Ciertos datos, relevantes para el análisis de fraude, estaban ofuscados para preservar la confidencialidad de la información, notablemente el número de tarjeta.
- Otros datos, también relevantes para el análisis, no estaban disponibles, notablemente el indicador de condición del POS (que permite identificar las transacciones hechas por Internet).
- Debido a la carencia de estos datos, no pudimos crear varios de los «cortes» que identificamos como interesantes para hacer el análisis.

- Se sabe que, en general, por problemas operativos, los datos registrados en campos importantes (como ser identificación del comercio o su ubicación geográfica) son de baja calidad, lo cual causa que los atributos complejos creados a partir de este dato arrastren este ruido.
- Sólo pudimos disponer de tres meses de datos, lo cual impide el uso de atributos complejos que resumen el comportamiento promedio a lo largo de períodos extendidos de tiempo. Este problema es aún más crítico ya que es necesario utilizar una parte de los datos para entrenamiento (primeros dos meses) y el resto para evaluación (último mes), aún sabiendo que las transacciones del primer mes (sobre todo) tenían gran cantidad de sus atributos complejos sin calcular, debido a la falta de datos históricos.
- Sólo pudimos utilizar atributos básicos del protocolo ISO 8583; al aplicar las técnicas presentadas en una entidad financiera real (emisor o adquirente) se puede disponer de atributos adicionales (del tarjeta habiente o comercio, respectivamente) que pueden ser relevantes para la detección de fraude, según se discutió en la subsección 2.3.4.

Más allá de los resultados de evaluar el modelo de detección entrenado, a continuación resaltamos algunas conclusiones generales que surgen del estudio realizado.

Sesgo en los datos

Aún luego de superar los problemas de calidad de datos descritos previamente, queda el problema del sesgo. Los conjuntos de transacciones reales suelen ser muy sesgados, con porcentajes de fraude de entre 0.1 % y 1 %. Es fundamental atacar este problema, lo cual se hace comúnmente mediante el muestreo, o sea obteniendo subconjuntos del conjunto original según algún criterio.

El criterio trivial de utilizar todas las transacciones fraudulentas pero elegir un subconjunto al azar de las transacciones legítimas resultó no tener utilidad, según se explicó en el capítulo 5. De hecho, concluimos que cualquier criterio razonable debe ser hecho a nivel de cuenta, eligiendo primero algunas cuentas, y utilizando luego todas sus transacciones.

Por otra parte, las cuentas deben ser seleccionadas de formas apropiadas según las características del negocio. Destacamos entonces las sugerencias de posibles conjuntos hechas en la sección 5.1 de este trabajo.

Evaluación de clasificadores

Algunas de las medidas presentadas en la sección 3.4 no están en la bibliografía, y son específicas para el problema de detección de fraude en medios de pago:

- Promedio, desviación estándar e histograma de las siguientes medidas tomadas sobre las transacciones fraudulentas entre la primer transacción fraudulenta de una cuenta y la primer alerta generada:
  - Cantidad
  - Suma de montos
- Cobertura de montos de fraude (del modelo de detección completo contra el conjunto de evaluación total)
- Cobertura de cuentas fraudulentas

Al respecto deseamos comentar que su presentación al cliente del proyecto fue positiva, ya que permiten obtener una visión del desempeño del modelo de detección muy comprensible para los analistas de riesgo, que son los usuarios finales del sistema. En contraposición, las medidas de evaluación tradicionales son muy abstractas y no se relacionan fácilmente con el objetivo final del sistema, que es reducir las pérdidas financieras causadas por el fraude.

Desempeño del *naive bayesian*

La fase de cálculo de puntaje de riesgo, y su correspondiente comparación contra el umbral de activación, tiene el objetivo de mejorar la predicción de las reglas. Cabe preguntarse qué pasaría si no existiera esta etapa, es decir: si se tomara la predicción de las reglas como absoluta para la predicción de fraude en una transacción. Esta pregunta puede responderse evaluando el desempeño de un modelo de detección que consta sólo de reglas, llamado $M-$, según se discutió en la sección 3.4.2.

En el caso particular del modelo de detección entrenado según lo descrito al comienzo de esta sección, su correspondiente $M-$ obtuvo una `precision = 2.11%` y `cobertura = 75.05%`, es decir: 5 veces menos precisión que M, pero 50 % más de cobertura. Consultado acerca de estos resultados, el cliente del proyecto prefirió el de M, ya que la precisión indicada está por debajo del mínimo aceptable en el negocio y la cobertura obtenida no compensa esta pérdida. De todas formas, desde un punto de vista general la decisión es dudosa. En consecuencia, podemos concluir que la aplicación de *naive bayesian* al cálculo de puntaje de riesgo no resultó lo suficientemente provechosa, ya que esperaríamos una diferencia más significativa a favor de M sobre $M-$.

Herramientas implementadas

Las herramientas implementadas conforman un prototipo casi completo del *Smart Analyzer* que abarca toda la funcionalidad requerida, exceptuando la

herramienta `aggs`, cuyas funcionalidades escapan al alcance del estado del arte en el área de Aprendizaje Automático. En su conjunto, las herramientas demuestran la factibilidad de los enfoques sugeridos para resolver las funcionalidades del *Smart Analyzer*, ya que los resultados obtenidos con datos reales son alentadores, a pesar de las múltiples dificultades presentadas por las características del problema. Además, la automatización de todas las fases del proceso permitió realizar las pruebas de forma sistemática y obtener resultados precisos.

Resumen final

El resultado final del presente proyecto constituye un avance importante para el cliente en pos de obtener un *Smart Analyzer* comercializable. En efecto, hemos generado los siguientes resultados de valor para el cliente:

- Una documentación detallada acerca de las características del negocio de medios de pago en general, y del fraude en este negocio en particular
- Un análisis profundo de las diversas problemáticas asociadas a los requerimientos del *Smart Analyzer*
- Un prototipo funcional que cubre la mayoría de las funcionalidades requeridas, y puede ser probado en ambientes reales
- Una sugerencia de tareas a realizar para mejorar este prototipo

Estos resultados representan una reducción drástica en los riesgos tecnológicos de obtener una versión terminada del *Smart Analyzer*.

6.3. Trabajos futuros

El presente proyecto ha resuelto la mayoría de los objetivos planteados. Sin embargo algunos puntos quedan abiertos a una mayor investigación.

En primer lugar, si bien la alternativa de ILP para el aprendizaje de reglas fue descartada en base a lo sugerido por la bibliografía, no se hicieron pruebas utilizando ninguna implementación de ILP. Por lo tanto puede ser interesante hacer pruebas con alguna implementación que siga este enfoque, que tiene para ofrecer, principalmente, la posibilidad de poder inducir reglas de nivel tres sin tener que crear atributos complejos para ellas.

En segundo lugar, el cálculo de puntaje de riesgo se resolvió mediante un algoritmo *naive bayesian*. No se hicieron pruebas con otros enfoques, de los que hay varios capaces de aprender una función numérica. Considerando además que los resultados obtenidos con el *naive bayesian* fueron de baja calidad (según surge

de la comparación de los resultados obtenidos con M y $M-$), es importante avanzar en este camino. Por otra parte, la opción de optimizar el umbral de activación según la medida F_α (así como también el valor usado para $\alpha, 0{,}05$) no tiene sustento teórico; quizás otro criterio de optimización sería superior. Por ejemplo, se podría experimentar con el criterio de proximidad geométrica ponderada al punto (0,1) de la gráfica de ROC.

En tercer lugar, recordar que una transacción puede ser clasificada como fraudulenta por varias reglas. Podría pensarse que cuanto mayor esta cantidad, mayor es la probabilidad de fraude. Este punto no fue considerado en nuestro trabajo. Eventualmente, podría ser de interés tomarlo en cuenta para el cálculo de puntaje de riesgo, de forma de generar alertas para transacciones que se hubieran tomado como legítimas según el proceso normal. Así se podría lograr una mayor cobertura. Pensamos que es una mejora fácil de implementar y que podría arrojar mejoras en los resultados obtenidos.

En cuarto lugar, el foco del proyecto estuvo en encontrar técnicas mediante las que resolver los problemas planteados desde el punto de vista funcional. Sin embargo, no nos centramos en atacar los conjuntos masivos de datos que comúnmente manejan las entidades financieras, donde fácilmente pueden existir decenas de millones de transacciones por mes. Las herramientas implementadas por nosotros están fuertemente optimizadas para ejecutarse sobre varios millones de ejemplos. En cambio, las implementaciones que utilizamos de algoritmos de árboles de decisión funcionan cargando el conjunto completo de ejemplos en memoria, lo cual no es escalable a conjuntos arbitrariamente grandes. Por lo tanto, es importante buscar implementaciones de los algoritmos utilizados que puedan funcionar sin asumir que el conjunto de ejemplos cabe en memoria.

Por otra parte, los mejores resultados fueron obtenidos con el programa `C4.5`, el cual en su fase de generación de reglas es de orden $O(n^3)$, por lo que el tiempo de entrenamiento se hace intolerable pasada una cierta cantidad de ejemplos. Conviene entonces comprobar si su versión comercial, `See5`, incluye mejoras en este punto; de lo contrario será necesario buscar otra implementación, más eficiente.

En este camino puede valer la pena probar las implementaciones de algoritmos de árboles de decisión ofrecidas por los proveedores principales de manejadores de bases de datos relacionales. Al estar integrados con sus respectivos productos, existe la expectativa de que soporten conjuntos masivos de ejemplos.

Pasando a las posibles mejoras desde el punto de vista funcional, pensamos que los mejores resultados se obtienen en función de lo apropiado de los atributos elegidos. En consecuencia es un punto a continuar estudiando. Por ejemplo, se podría haber agregado un atributo que agrupe varios MCC similares, o más atributos complejos. Además, los atributos disponibles en el conjunto de datos inicial pueden variar, con lo cual sería necesario analizar qué atributos complejos son posibles, o potencialmente interesantes, aprovechando esa nueva información.

Otra posible mejora es estudiar otros posibles subconjuntos a utilizar para el entrenamiento, además de los mencionados de cuentas fraudulentas, cuentas del extranjero, por producto o tipo de producto, y cuentas con compras por Internet.

También es importante experimentar con datos que abarquen períodos más extensos. Al disponer de tres meses, las pruebas fueron hechas utilizando dos meses de entrenamiento y uno de evaluación. Se utilizaron cortes sobre el conjunto de evaluación para inferir reglas, y el conjunto entero de entrenamiento (incluyendo las transacciones del corte) para la primer fase de evaluación. Al trabajar sobre períodos más prolongados de tiempo es factible, por ejemplo, construir dos conjuntos disjuntos de entrenamiento para reducir la posibilidad de experimentar sobreajuste.

Por otra parte, destacamos la variedad de medidas de evaluación implementadas en la herramienta `sbox`. Sin embargo es posible que sea interesante, o útil desde el punto de vista del negocio, agregar otras. Un ejemplo sería medir la cantidad de minutos que pasan entre el primer fraude de una cuenta y la primer alerta generada para ella.

Por último, recordar que no se realizó una implementación de la herramienta `aggs`, cuya funcionalidad sería sugerir qué atributos complejos conviene utilizar. De todas formas, es posible atacar este problema en forma manual, definiendo muchos atributos complejos y eliminando aquellos que ninguna regla utiliza, gracias a que los algoritmos usados para generación de reglas no tienen problemas al manejar grandes cantidades de atributos. Aún así, un camino a investigar a largo plazo es buscar formas de resolver los requerimientos de esta herramienta.

Glosario

Adquirente Entidad financiera, banco, o empresa que acepta transacciones de crédito generadas en los comercios afiliados.

Aprendizaje Automático Área que se dedica al desarrollo de programas que pueden aprender a partir de ejemplos o de la experiencia. Se dice que un programa aprende cuando mejora su desempeño (respecto a alguna medida) en una cierta tarea a través de la experiencia.

Aprendizaje con árboles de decisión Método para aproximación de funciones con valor objetivo discreto, en el que la función objetivo es representada como un árbol de decisión.

Árbol de decisión Grafo en que las aristas representan decisiones y las hojas el resultado obtenido a partir de las decisiones tomadas previamente. Un árbol de decisión puede representarse como un conjunto de reglas *if-then-else*.

Autorización (pedido de) Forma de iniciar una transacción de tarjeta de crédito. El pedido de autorización se envía al adquirente y al emisor correspondientes, quienes deben verificar los datos de la transacción (notablemente el disponible del tarjeta habiente) para responder en consecuencia.

BIN Código de identificación internacional de entidades financieras (*Bank Identification Number* por sus siglas en inglés).

C4.5 Un programa de aprendizaje que genera árboles de decisión desarrollado por Ross Quinlan. Contiene varias mejoras respecto al que toma como base (ID3).

Clasificador Función que asocia un valor de su dominio (discreto o continuo) a una etiqueta perteneciente a un conjunto discreto.

Cobertura Proporción de casos positivos (fraudes) encontrados en las predicciones de un clasificador.

Código de respuesta Forma de codificar las respuestas a un pedido de autorización. El código de respuesta puede ser 00 (autorizado) o un código de dos caracteres indicando la razón por la que se rechaza la transacción (por ejemplo: falta de fondos).

Contracargo Proceso mediante el cual un tarjeta habiente puede quejarse por una transacción que se le imputó incorrectamente. El proceso de contracargo puede terminar en que se identifique la transacción como fraudulenta, en cuyo caso se le devuelve el dinero al tarjeta habiente y alguna de las partes involucradas (adquirente, emisor o comercio) asume el costo de la transacción y del proceso de contracargo en sí.

Disponible Monto de dinero disponible en la cuenta de crédito de un tarjeta habiente.

Emisor Entidad financiera, banco, o empresa que emite tarjetas de crédito a los tarjeta habientes.

EMV Estándar para la emisión de tarjetas basadas en un chip integrado en el plástico (en lugar de la tradicional banda magnética).

Exactitud Proporción global de aciertos de las predicciones de un clasificador sobre el total de los ejemplos.

Gráfica de ROC Gráfica que muestra el comportamiento de un clasificador ante un determinado conjunto y no es sensible al sesgo del mismo.

ID3 Uno de los primeros programas de aprendizaje de árboles de decisión. Demuestra la capacidad del enfoque a pesar de sus limitaciones en cuanto a tipos de datos permitidos y técnicas sofisticadas para recortar el árbol aprendido.

ISO-8583 Protocolo que define el formato de los mensajes de intercambio electrónico de transacciones entre entidades financieras (adquirente, emisor, marca internacional, etc.).

Límite de crédito Máximo del disponible de un tarjeta habiente.

Marca internacional Entidad financiera que asocia, y permite interactuar, varios adquirentes y emisores de distintos países.

Matriz de confusión Matriz de 2x2 que resume el comportamiento de un clasificador respecto a un cierto conjunto. Muestra verdaderos positivos, falsos negativos, verdaderos negativos y falsos positivos.

MCC Código de rubro de comercios (*Merchant Category Code* por sus siglas en inglés).

Medida-F Combina la precisión y cobertura en una sola medida. Depende de un parámetro α, el cual se puede usar para dar más peso a la cobertura o a la precisión.

Naive bayes Clasifcador probabilístico sencillo basado en el Teorema de Bayes y en supuestos de independencia que simplifican su funcionamiento.

POS Genéricamente Punto de Venta (*Point Of Sale* por sus siglas en inglés). Normalmente usado para referirse a un lector electrónico de tarjetas de crédito, ubicado en un comercio.

Precisión Proporción de aciertos de las predicciones de un clasificador sobre las predicciones positivas (fraudes).

Programación lógica inductiva Técnica para aprender programas lógicos a partir de un conjunto de ejemplos clasificados.

Puntaje de riesgo Valor numérico real comprendido entre 0 y 1 que se le asigna a una transacción; representa una estimación de la probabilidad de que la misma sea legítima o fraude. A mayor puntaje de riesgo, más probabilidad que la transacción sea fraude.

QUEST Un programa de aprendizaje que genera árboles de decisión desarrollado por Wey-Yin Loh y Yu-Shan Shih. El nombre significa *Quick, Unbiased and Efficient Statistical Tree*.

Regla Predicado lógico que al ser evaluado frente a una transacción evalúa en verdadero si considera que la misma es un fraude, y en falso si considera que es legítima.

Tarjeta habiente Persona que posee una tarjeta de crédito.

Transacción (de tarjeta de crédito) Intercambio de promesa de pago a través de la tarjeta de crédito por un bien o servicio en un comercio afiliado a un adquirente que tenga un acuerdo con el emisor de la tarjeta.

Umbral de activación Valor numérico real comprendido entre 0 y 1 que se compara contra el puntaje de riesgo asignado a una transacción. Si el puntaje de riesgo supera el valor del umbral, se genera una alerta para esa transacción.

Bibliografía

[1] The credit card prank. http://www.zug.com/pranks/credit/, último acceso en setiembre del 2006.

[2] EMVCo. http://www.emvco.com/, último acceso en setiembre del 2006.

[3] Especificación del ISO 8583. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=31628&scopelist=, último acceso en setiembre del 2006.

[4] FairIsaac. http://www.fairisaac.com/, último acceso en setiembre del 2006.

[5] Glosario de términos del iso 8583. http://www.transcan.com/glossary.html, último acceso en setiembre del 2006.

[6] iHex. http://www.futureroute.co.uk/technology_scientific_foundations.html, último acceso en setiembre del 2006.

[7] The OC1 decision tree software system. http://www.cs.jhu.edu/~salzberg/announce-oc1.html, último acceso en setiembre del 2006.

[8] PayTrue Solutions. http://www.paytrue.com, último acceso en setiembre del 2006.

[9] Pérdidas por fraude en Canadá. http://www.rcmp-grc.gc.ca/scams/ccandpc_e.htm, último acceso en setiembre del 2006.

[10] Pérdidas por fraude en Inglaterra. http://www.theregister.co.uk/2005/03/08/apacs_fraud_2004/, último acceso en setiembre del 2006.

[11] Pérdidas por fraude en U.S.A. http://www.theregister.co.uk/2005/02/02/ftc_fraud_report/, último acceso en setiembre del 2006.

[12] Resumen del ISO 8583. http://www.amarshall.com/resix/iso8583.html, último acceso en setiembre del 2006.

[13] The torn-up credit card application. http://www.cockeyed.com/citizen/creditcard/application.shtml, último acceso en setiembre del 2006.

[14] Tutorial de C4.5. http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html, último acceso en setiembre del 2006.

[15] Visa cuts CardSystems over security breach. http://www.theregister.co.uk/2005/07/19/cardsystems/, último acceso en setiembre del 2006.

[16] VISA EU launches new advanced fraud detection tool. http://www.visaeurope.com/pressandmedia/newsreleases/press178_pressreleases.jsp, último acceso en setiembre del 2006.

[17] Visa merchant category classification (MCC) codes directory. http://www.usda.gov/procurement/card/card_x/mcc.pdf, último acceso en setiembre del 2006.

[18] Wikipedia: historia de las tarjetas de crédito. http://en.wikipedia.org/wiki/Credit_card#History, último acceso en setiembre del 2006.

[19] Boris Beizer. *Software Testing Techniques*. International Thomson Computer Press, 1990.

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

[21] Christian Borgelt. Full and naive bayes classifiers. http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/bayes/bayes.html, último acceso en setiembre del 2006.

[22] Leo Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

[23] William W. Cohen. Fast effective rule induction. In Morgan Kaufmann, editor, *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.

[24] Hand D.J. and Yu K. Idiot's bayes - not so stupid after all? *International Statistical Review*, 69:385–398, 2001.

[25] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[26] Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs, 2004.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer, 2003.

[28] Thomas Hill and Paul Lewicki. *Statistics: Methods and Applications*. StatSoft, Inc., 2005.

[29] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.

[30] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica*, 7(4):815–841, 1997.

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

[32] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[33] Claudia Perlich and Foster Provost. Aggregation-based feature invention and relational concept classes. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 167–176, New York, NY, USA, 2003. ACM Press.

[34] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. In *en borrador (versión de febrero del 2005), a aparecer en Artificial Intelligence Review*, la última versión está en `http://www.bsys.monash.edu.au/people/cphua/`.

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

[36] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[37] J. Ross Quinlan. Página personal de R. Quinlan. `http://www.rulequest.com/Personal/`, último acceso en setiembre del 2006.

[38] J. Ross Quinlan. See5. `http://www.rulequest.com/see5-info.html`, último acceso en setiembre del 2006.

[39] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Documentación del sistema. Documentación suplementaria del proyecto.

[40] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `dgen`. Documentación suplementaria del proyecto.

[41] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `dprep`. Documentación suplementaria del proyecto.

[42] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `match`. Documentación suplementaria del proyecto.

[43] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `mmerge`. Documentación suplementaria del proyecto.

[44] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `mview`. Documentación suplementaria del proyecto.

[45] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `rlearn`. Documentación suplementaria del proyecto.

[46] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `sbox`. Documentación suplementaria del proyecto.

[47] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Guía rápida de uso de `slearn`. Documentación suplementaria del proyecto.

[48] Diego Rivero, Álvaro Rodríguez, and Dieter Spangenberg. Plan de verificación. Documentación suplementaria del proyecto.

[49] Yu Shan-Shih. QUEST. http://www.stat.wisc.edu/~loh/quest.html, último acceso en setiembre del 2006.

[50] Yu Shan-Shih. QUEST user manual. http://www.stat.wisc.edu/~loh/treeprogs/quest/questman.pdf, revisado en junio del 2003, último acceso en setiembre del 2006.

[51] StatSoft, Inc. Electronic Statistics Textbook. http://www.statsoft.com/textbook/stathome.html, último acceso en setiembre del 2006.

[52] Gary M. Weiss and Foster Provost. The effect of class distribution on classifier performance: an empirical study. Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.