ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Modelo predictivo de deserción de clientes de un banco usando librería Mahout del framework HADOOP”

TESINA DE SEMINARIO

Previo la obtención de los Títulos de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN SISTEMAS DE INFORMACIÒN

Presentada por:

Fernando Xavier Loor Mera

María José Loor Martínez

GUAYAQUIL – ECUADOR
2011

DEDICATORIA

Dedico este proyecto de tesis primeramente a Dios, porqueSiempre me ha estado fortaleciendo en tiempos malos y porque me ha permitido llegar con vida hasta este momento especial, y por esto me encuentro totalmente agradecido, te amo Dios, también quisiera dedicar este proyecto de tesis a mis padres que siempre estuvieron respaldándome y dándome ánimos a seguir, ahí cuando quería botar la toalla ellos estuvieron dándome consejosy ayuda, además de todas las enseñanzas que he visto por medio deellos, son muy especiales para mi, y quizá no me alcance el tiempoen esta vida para decir cuánto los amo, los amo enteramente.

Fernando Xavier Loor Mera

La presente tesis se la dedico de antemano a Dios, quien con sus bendiciones me ha permitido alcanzar una meta más en mi vida profesional. La dedico a mi familia que gracias a sus consejos me ayudaron a crecer como persona. A mis padres y hermana por su infinito apoyo, confianza y amor. Gracias por ayudarme a cumplir mis objetivos como estudiante y profesional. Un agradecimiento especial a mis jefes, quienes siempre estuvieron dispuestos a ayudarme y enseñarme. Gracias por la oportunidad de continuar y finalizar mis estudios.

María José Loor Martínez

AGRADECIMIENTO

Este trabajo ha sido un esfuerzo conjunto del equipo de trabajo María José Loor y Fernando Loor, departe de nosotros quisiéramos agradecer a nuestra profesora y guía la Ingeniera Vanessa Cedeño sin la cual nos hubiéramos perdido en este duro camino, también quisiéramos agradecer a las personas que estuvieron dándonos ánimos, entre ellas compañeros, familiares, que nunca dudaron en darnos una mano cuando más lo necesitábamos, un profundo agradecimiento a cada uno de los maestros que tuvimos en nuestra larga etapa universitaria por todas las enseñanzas ya sean estas científicas o vivenciales, un gracias a la Espol.Gracias Dios, a nuestros Padres, a la Ingeniera Cedeño y a todos los que estuvieron ahí.

DECLARATORIA EXPRESA

"La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la **Escuela Superior Politécnica del Litoral**"

(Reglamento de Graduación de la ESPOL)

_____

FERNANDO XAVIER LOOR MERA

_____

MARÍA JOSÉ LOOR MARTINEZ

TRIBUNAL DE SUSTENTACIÓN

Ing. Lenin Freire.	Ing. Vanessa Cedeño
PROFESOR DELEGADO POR EL DECANO	**PROFESORA DEL SEMINARIO**

RESUMEN

Las empresascon productos y servicios altamente competitivosse enfrentan frecuentemente a problemas de rotación de clientes. Esta problemática ha originado que las compañías dirijan su atención a los análisis de lealtad y deserción de clientes. La gran acogida de esta visión se debe a que los costos que incurre en mantener a un cliente son menores que la adquisición de nuevos.

El siguiente trabajo refiere la aplicación de métodos de Regresión Logística para determinar un modelo predictivo de deserción de clientes, a través del sistema de procesamiento masivo de datos, HADOOP. El enfoque intrínseco del presente estudio corresponde a una entidad financiera, específicamente un banco, cuyos datos históricos recolectados permitirán precisar los factores y variables demográficas que inciden en la fuga de clientes con la mayor precisión posible. De igual forma, se identificará el comportamiento transaccional de dichos clientes.

El entrenamiento y la evaluación del modelo predictivofacultará el pronóstico de futuros desertores, para con ello la empresa pueda evaluar a sus clientes actuales y preparar estrategias de retención, en el caso que lo ameriten.

INDICE GENERAL

ANEXOS

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Compañías como bancos, servicios de telefonía, proveedores de servicios, etc, suelen realizar análisis para obtener métricas que les permitan evaluar la situación actual de sus clientes. Con ello pueden desarrollar estrategias que le ayuden a retener a los clientes propensos a terminar sus contratos con estas empresas.

Los clientes suelen terminar su relación con la empresa por diferentes causas, una de ellas puede ser inconformidad con el servicio ofrecido, o más bien problemas con el recurso humano y la calidad de atención; así como también se puede considerar el costo de los productos que mantienen y que en el mercado son ofertados a un precio menor.

Uno de los objetivos de implementar un modelo de predicción es poder constatar los factores que determinan la salida o fuga de los clientes, y poder hacer un esfuerzo por mejorar cada uno de ellos. Así también se puede encontrar un patrón en el comportamiento transaccional del cliente, para comenzar a actuar sobre las alertas.El porqué de las empresas para usar estos métodosradica en los costos, puesto que esmás factible y menos costoso mantener a un cliente que buscar y atraer nuevos, por ello la importancia a la búsqueda de los mejores modelos para determinarlos.

En cada uno de los capítulos tratados daremos un paseo por cada uno de los aspectos necesarios para llegar a determinar un modelo predictivo. El contenido de este documento se distribuye de la siguiente manera: en el capítulo 1 se plantea la problemática que se desea abordar, los antecedentes ylos objetivos, así como la justificación y la definición del alcance. En el capítulo 2 se hace una exposición de los conceptos necesarios para comprender el tema planteado, se expone las técnicas a usar y los fundamentos teóricos. En el capítulo 3 se describe el análisis de la solución y se hace referencia a la metodología implementada. También se explica la estructura del conjunto de datos de entrada. En el capítulo 4 se muestra el entrenamiento y la evaluación del modelo. Y finalmente, las pruebas y resultados son mostrados en el capítulo 5.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN.

1.1. Antecedentes y Descripción del Problema

Antecedentes

Uno de los grandes problemas de las empresas se da en la competitividad, ya que los clientes se fugan en gran manera, debido a las promociones o ventajas que ofrecen los bancos X, Y o Z; estas promociones, servicios o calidad suelen ser factores de[finitorios para que un cliente decida concluir su contrato o en este caso su cuenta con una empresa y tomar el contrato de otra, esto implica un costo mucho mayor ya que si una parte de los clientes fuga de la empresa esta, se encuentra en la obligación de obtener nuevos clientes, y este es un gasto mucho mayor al que se hubiera invertido tratando de mantener a los clientes desertores.

Descripción del Problema

El problema en la deserción de clientes es identificar las variables principales que causan esta deserción a partir de datos históricos

recolectados, para elaborar un modelo que explique el comportamiento de la deserción y permita pronosticar a posibles desertores. El pronóstico servirá para que la empresa tome medidas preventivas.

.

1.2. Justificación

Con este proyecto se pretende obtener un modelo predictivo de deserción de clientes, el cual no solo identificará algunas de las variables más relevantes e influyentes en la fuga de la clientela, sino que, también proporcionará un procedimiento para evaluar fácilmente a los clientes actuales y así determinar quiénes podrían ser futuros desertores. Con la información obtenida, la empresa estará en la facultad de tomar decisiones efectivas sobre los aspectos que se denotan en los resultados y así fidelizar a sus clientes.

1.3. Objetivos

- Identificar los posibles factoresdemográficos que intervienen en la fuga de clientes.
- Analizar el comportamiento transaccional de los clientes que han abandonado la empresa.

- A través de algoritmos de clasificación como Regresión Logística de la librería Mahout, determinar un modelo matemático que permitirá la predicción de futuros desertores.

1.4. Alcance y Limitaciones

Alcance

El alcance de esta aplicación está destinado a colaborar con la entidad financiera para poder obtener información de los clientes que posiblemente quieran cancelar sus servicios y productos de la institución. Debido a que existen muchas razones a considerar para que un cliente cancele sus productos y servicios, y como no existe una retroalimentación de todas las causas de abandono, se ha limitado el alcance de este proyecto a la parte transaccional.

Limitaciones

Existen algunas limitaciones referentes a la aplicación en sí y al proyecto en general, que se detallan a continuación:

- La obtención de los datos para las pruebas respectivas de la aplicación no se las consigue de una manera sencilla, debido a su naturaleza confidencial. La base de datos obtenido es solo una

pequeña muestra con algunos de los clientes desertores del banco.

- El tiempo destinado al proyecto siempre es una limitación, ya sea por eventos ajenos a nosotros que pueden suceder.
- La disposición del lugar o espacio para hacer las pruebas respectivas de la aplicación ya sea por la ubicación, las maquinas, etc.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

2.1. Marco Teórico Conceptual

2.1.1. El concepto de deserción de clientes

No existe una definición única sobre los problemas de deserción, pero generalmente, el término deserción se refiere a todos los tipos de abandono de clientes ya sea voluntario o involuntario [1]. Como lo reconocemos en la parte práctica, pues depende del tipo de compañía. En nuestro estudio, el cliente es tratado como desertor si el cliente tuvo un producto (cuentas, tarjetas de crédito, créditos) en un periodo de tiempo y ha cancelado todos los productos y servicios. Si el cliente tiene al menos un producto activo no se considera como desertor.

El objetivo en un análisis de deserción de clientes es determinar los clientes con alto potencial de abandono y si es posible analizar los clientes a quienes vale la pena retener. El tipo de sector empresarial y la relación que la misma mantiene con sus clientes, afecta el cómo es

la deserción de los clientes.La deserción de clientes está relacionada estrechamente con la retención y la lealtad del cliente. [2]

El sector bancario es un mercado típico donde el cliente regularmente no se cambia de compañía. Los clientes, por lo general confían sus negocios bancarios a una o dos compañías por un largo periodo de tiempo. Esto hace que la deserción de clientes sea una prioridad para este tipo de negocio. [3]

2.2. Marco Teórico Instrumental

2.2.1. HADOOP

Hadoop es un sistema de almacenamiento y procesamiento de datos. Es escalable, tolerante a fallos y distribuido [4]. Está diseñado para el procesamiento distribuido de grandes conjuntos de datos a través de clúster de ordenadores mediante un modelo de programación simple. Hadoop es capaz de almacenar cualquier tipo de datos en su formato original para llevar a cabo una amplia variedad de análisis y transformaciones en esos datos. En lugar de confiar en el hardware para tener una alta disponibilidad, el sistema detecta y controla los errores en la capa de aplicación, sin perder los datos o interrumpiendo análisis de los mismos.

Hadoop está soportado por dos componentes fundamentales: HDFS (HadoopDistributed File System) Y MapReduce.

2.2.2. HDFS

El HadoopDistributed File System (HDFS) es un sistema de archivos distribuido, escalable y portátil escrito en Java para el frameworkHadoop. Cada nodo en una instancia Hadoop típicamente tiene un único nodo de datos; un clúster de datos forma el clúster HDFS. La situación es típica porque cuando los datos llegan al clúster, el software HDFS lo rompe en piezas y distribuye las piezas entre los diferentes servidores que participan en el clúster. Cada servidor almacena sólo un pequeño fragmento del conjunto completo de datos, y cada parte de los datos se replica en más de un servidor. HDFS fue diseñado para gestionar archivos muy grandes.

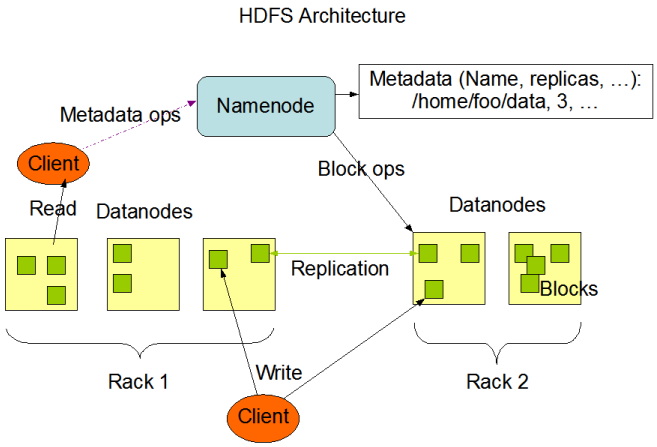

Figura 1.1– Arquitectura HDFS.

2.2.3. MapReduce

La enciclopedia libre Wikipedia [5] define a MapReduce como un framework introducido por Google, el cual soporta la computación en paralelo sobre grandes colecciones de datos en clústeres de computadoras. El framework ha sido inspirado en funciones *map* y *reduce* utilizadas comúnmente en los lenguajes funcionales, aunque con propósitos diferentes.

MapReduce divide el procesamiento en dos fases: Map y Reduce. El trabajo consiste en dividir el conjunto de datos de entrada en bloques independientes que son procesados por la función *map()*, la cual posee la característica de trabajar sobre grandes volúmenes de datos. La función *map()* puede ser invocada de forma distribuida a través de múltiples máquinas, una vez que los datos de entrada han sido divididos. Los bloques de entrada pueden ser procesados por diferentes máquinas de forma paralela. El frameworkordena la salida de la función *map()* y entrega los datos a la función *reduce()* para que ejecute cada uno de los elementos de la lista de valores intermedios que recibe. El resultado final se obtiene mediante la recopilación e interpretación de los resultados de todos los procesos que se ejecutaron.

MapReduce se compone por un nodo maestro JobTracker y uno esclavo TaskTracker. El maestro es responsable de programar las tareas de los esclavos, hacerles seguimiento y re ejecutar las tareas fallidas. Los esclavos ejecutan la tarea según las instrucciones del maestro.

La Figura 1.1 muestra el flujo general de la implementación MapReduce. Cuando el programa de usuario se ejecuta sobre el frameworkMapReduce, se produce las siguientes acciones (los números etiquetados en la figura corresponden a los números de la siguiente lista) [6]:

1. La librería MapReduce divide los archivos de entrada en *M* segmentos. A continuación, se crean copias del programa en clusteres.
2. Una copia de este programa es el Master. El resto son los esclavos, a quienes el Master les ha asginado tareas. Existen un número *M* de tareas map y un número *R* de tareas reduce asignadas.
3. Un esclavo que tiene asignada las tareas *map*, lee el contenido de los bloques de entrada y los convierte en pares de clave/valor para pasar a la función *map*. Los pares clave/valor que son producidos por la función *map* son almacenados en memoria.

4. Periódicamente, los pares almacenados son escritos en el disco local. La ubicación de estos pares almacenados es entregada al máster, quien es responsable de reenviar estas ubicaciones a los esclavos con tareas reduce.
5. Cuando un esclavo reduce recibe estas ubicaciones, usa procedimiento para leer los datos almacenados en el disco local por los esclavos map. Cuando el esclavo ha leído toda la información intermedia, la ordena por las claves.
6. Los esclavos reduce realizan iterativamente el ordenamiento de la información intermedia por cada clave encontrada. La salida de la función *reduce* es añadida a un archivo de salida.

Cuando todas las tareas han sido completadas, el máster se levanta en el programa de usuario.

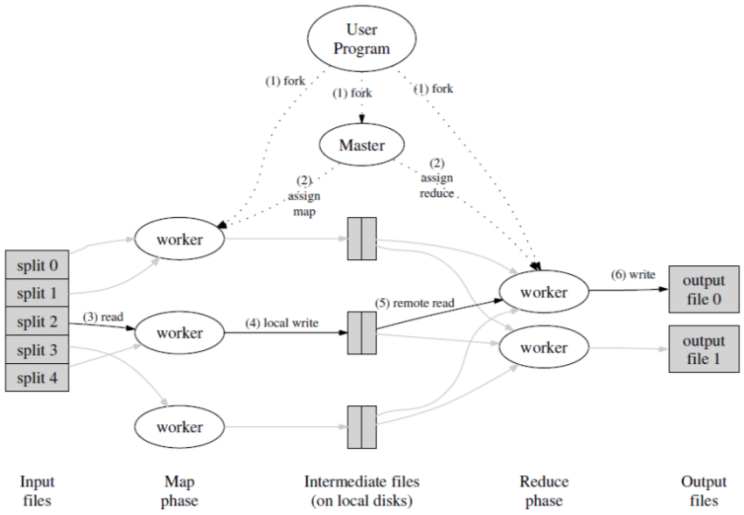

Figura 1.2– Arquitectura MapReduce.

2.2.4. Librería Mahout

Mahout es en teoría una librería escalable y abierta a implementación de todo tipo de técnicas de máquina de aprendizaje. Mahout divide sus algoritmos en tres áreas principales: filtros colaborativos (recomendadores), clustering y clasificación. Actualmente, Mahout se ha implementado y trabaja sobre varias aplicaciones de la vida real: e-commerce, email, videos, fotos, y todo lo que involucre a gran escala.

Los algoritmos declasificación que ofrece Mahoutson el foco de estudio de acuerdo a los objetivos perseguidos en este proyecto, ya que estos usualmente son usados para desarrollar modelos de predicción o detección. El libro Mahout in Action [7], ofrece un ejemplo bastante claro y similar la problemática planteada. La clasificación es válida para predecir el abandono de clientes en industrias de telecomunicaciones. Una forma para diseñar sistemas de clasificación para esta aplicación, es usar los datos de años anteriores de los casos identificados de deserción.Reconocer las características de los clientes incluye el tratamiento del conjunto de datos disponible, la selección de las características que pueden predecir mejor que un cliente dejará el servicio. Una vez probado el modelo, el sistema de clasificación puede ser usado para predecir el comportamiento futuro de otros clientes basado en el conocimiento, especificando las

características más relevantes e incluyendo comportamientos recurrentes.Es importante resaltar, que el términopredicción es usado para describir lo que los sistemas de clasificación realizan. No se refiere en sí, a la habilidad de pronosticar futuros eventos.

En la práctica, el desarrollo de un sistema de clasificación se divide en tres partes: el entrenamiento del modelo, la evaluación del modelo y la implementación del mismo. El entrenamiento del modelo es usado para producir el modelo a través de la información disponible.La evaluación del modelo valora los datos a través del modelo resultante, permitiendo comparar la salida del modelo con la salida deseada, en el caso que se conozca.

Una vez que el modelo se comporta como se desea, normalmente se pone en producción. Los resultados de producción no 100% precisos, pero la calidad de la salida de datos podría mantenerse consistente con los datos que se usaron para el entrenamiento.

Para efecto de este trabajo, se seguirá la metodología sugerida por autor, la cual se describe en detalle en el Capítulo 3.

CAPÍTULO 3

3. ANÁLISIS DE LA SOLUCIÓN.

Como se menciona en el capítulo anterior, en la solución de este proyecto se adoptará el flujo de trabajo propuesto por el autor [7]. La Tabla Idescribe cada una de las etapas y sus pasos correspondientes. Este capítulo abordará únicamente la etapa de entrenamiento del modelo. Las siguientes etapas serán expuestas en elCapítulo 4.

Etapa	Pasos
1. Entrenamiento del modelo	Definición de la variable objetivo.
	Recolección de datos históricos.
	Definición de variables predictivas.
	Selección del algoritmo de aprendizaje.
	Usar el algoritmo seleccionado para entrenar el modelo.
2. Evaluación del modelo	Ejecutar los datos de prueba.
	Ajustar los datos de entrada (usar diferentes variables predictivas o diferentes algoritmos)
3. Usar el modelo en producción	Usar nuevos datos para estimar las valores de la variable objetivo.

Tabla I– Flujo de trabajo para un sistema de clasificación.

3.1. Entrenamiento del modelo

En la primera etapa del proyecto de clasificación, es necesario definir una variable objetivo, esto ayudara a seleccionar de forma apropiada la información histórica para el proceso de entrenamiento.

3.2. Definición de la variable objetivo.

La variable objetivo del presente estudio es la deserción. Al definir la variable objetivo, también se debe precisar los valores o categorías que tomará dicha variable. En este caso, el interés es conocer si un cliente es desertor o no lo es.

3.3. Recolección de datos históricos.

La fuente de datos elegida es dirigida en parte por la necesidad de recogerdatos históricos con valores conocidos que describan la variable objetivo.

Bajo la premisa de deserción de clientes, existen cuatro tipos de conjunto de datos: el comportamiento, la percepción y la información demográfica del cliente, así comolas variables macro económicas.

- El comportamiento del cliente identifica cuales partes de los servicios del banco son usados y cuan a menudo, es decir el comportamiento transaccional.
- La percepción del cliente es definida por la forma en que el cliente aprecia el servicio. Esto puede ser medido a través de encuestas e incluye índices de satisfacción, calidad de servicio, quejas, etc.
- Información demográfica del cliente son las variables más usadas, como edad, nivel de educación, estatus social, información geográfica.
- Las variables macro económicas identifica cambios económicos en la sociedad, las diferencias en las experiencias del cliente y como afecta en el servicio.

A causa de las limitaciones antes expuestas, el conjunto de datos recolectado está constituido únicamente por variables que describen el comportamiento del clientey variables de información demográfica.

3.4. Definición de las variables predictivas

Luego de haber seleccionado la variableobjetivo y definido el conjunto de datos, es necesario definirlas variables predictivas. Estas variables sonla codificación concreta de las características que describen la variable objetivo.

En el siguiente estudio se considerará una base de 4,750 clientes inactivos, cuya definición corresponde a todo aquel cliente que durante el periodo de 12 meses ha cancelado voluntariamente sus productos y servicios con el banco. El concepto de cliente inactivo atañe de manera particular al banco que nos proporcionó los datos. En otras instituciones financieras, la descripción puede variar.

El conjunto de datos a trabajar, registra a todos los clientes cuya fecha de inactivación fue en los meses de julio, agosto y septiembre del 2011. De acuerdo a lo expuesto anteriormente, prevalecen dos tipos de datos: la información demográfica y el comportamiento del cliente. En el primer conjunto se incluye información de la edad, el género, la ciudad, la dependencia salarial, y el tiempo de vida en el banco. Para efecto de análisis de comportamiento del cliente, se considera la información transaccional del periodo de Enero a Julio del 2010, cuando el cliente aún mantenía un estado activo.

El conjunto de variables transaccionales se considera en dos tiempos, en t=1 y t=6 (representa a los meses de Enero y Junio, respectivamente). La información transaccional contempla los productos principales y genéricos respecto a otros bancos. Estos productos son: cuentas de ahorros, cuentas corrientes, pólizas de

acumulación, créditos, créditos para autos y créditos para casas. De estos productos se excluye tarjetas de crédito, puesto que intervienen otros tipos de factores. La Tabla II muestra todas las variables con su respectiva descripción.

Variable	Descripción
AGE	Edad del cliente
BANK_AGE	Tiempo de vida del cliente en el banco
SEX	Género del cliente
DEPENDENCY	Dependencia salarial del cliente
NOPROD_t	Número de productos activos en t
TRXAUT_t	Transacciones correspondiente a créditos de auto en t
TRXCAS_t	Transacciones correspondiente a créditos de casa en t
TRXCC_t	Transacciones correspondiente a cuentas corrientes en t
TRXCA_t	Transacciones correspondiente a cuentas ahorros en t
TRXCRE_t	Transacciones correspondiente a creditos financieros en t
TRXPOL_t	Transacciones correspondiente a polizas en t
NOTRX_IN_t	Número de depositos en t
NOTRX_OUT_t	Número de retiros en t
NOTRX_PRE_t	Número de transacciones presenciales en t
NOTRX_VIR_t	Número de transacciones virtuales en t

Tabla III– Descripción de las variables de entrada.

En un análisis descriptivo de los datos, presenta que el 57% de clientes pertenece al género masculino, y el 43% al género femenino. En la Figura 3.1 se muestra la distribución de las edades, donde

fácilmente se puede observar que un gran porcentaje de clientes fluctúa entre el rango de 25 y 35 años de edad.

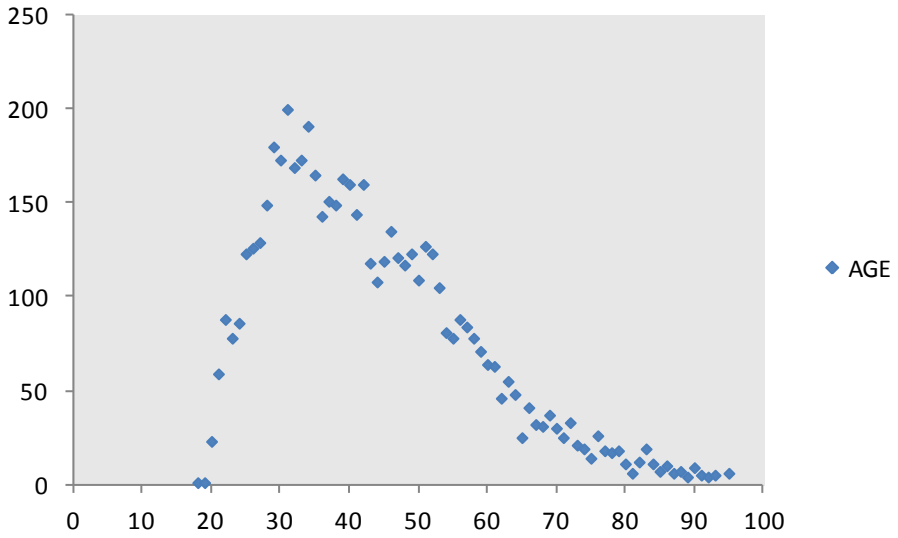

Figura 3.1– Distribución de la edad de los clientes.

De igual manera, se ha calculado la distribución del tiempo de vida del cliente en el banco, el cual revela que el 23% de los clientes desertores tienen un tiempo de actividad promedio de 2 años. La Figura 3.2 muestra el diagrama de dispersión correspondiente al tiempo de vida en el banco.

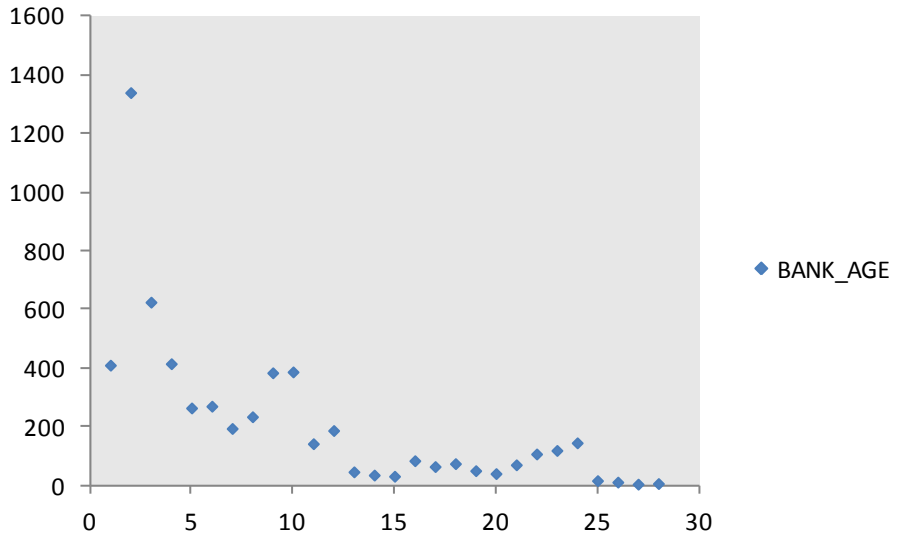

Figura 3.2– Distribución del tiempo de vida del cliente.

La Figura 3.3 muestra la cantidad de transacciones por producto en los tiempos t = 1 y t = 6. Como se aprecia en el gráfico, el número de transacciones disminuye en el tiempo. El producto cuenta de de ahorros tiene una participación significativa en el número de productos activos, y se mantiene al final del tiempo de evaluación. A partir de estos gráficos se puede deducir que el comportamiento transaccional de un cliente desertor es de forma decreciente.

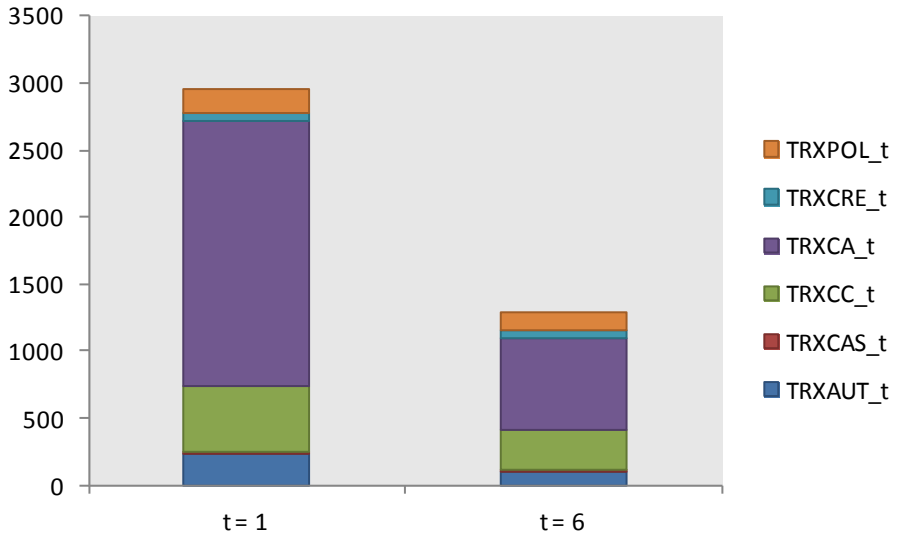

Figura 3.3– Distribución de las transacciones por producto en los tiempos t=1 y t=6.

3.5. Selección del algoritmo de aprendizaje.

Mahout ofrece una serie de algoritmos de clasificación, pero muchos están diseñados para manejar grandes conjuntos de datos y como resultado puede ser un poco incómodo de usar. Algunos, en cambio, son fáciles de empezar a trabajar porque, aunque son escalables, tienen bajos costos para los conjuntos de datos pequeños.

Uno de estos métodos bajo costo operativo es el descenso de gradienteestocástico (SGD) para el algoritmo de regresión logística. Este algoritmo es un algoritmo secuencial (no paralelo), pero es rápido. Lo más importante a considerar para trabajar

con grandes volúmenes de datos, es que el algoritmo SGD utiliza una cantidad constante de la memoriaindependientemente del tamaño de la entrada.

La regresión logística es una forma de regresión la cual es usada en una situación cuando la variable objetivo no es continua, es decir, un estado puede o no puede suceder, o representa una categoría en una clasificación especifica. [8]

Por sus características, los modelos de regresión logística permiten dos finalidades:

- Cuantificar la importancia de la relación existente entre cada una de las variables predictivas y la variableobjetivo, lo que lleva implícito también clarificar la existencia de interacción y confusión entreambos tipos de variables.
- Clasificar individuos dentro de las categorías (presente/ausente) de la variable objetivo, según laprobabilidad que tenga de pertenecer a una de ellas dada la presencia de determinadas variables predictivas.

El objetivo primordial que resuelve esta técnica es modelar la forma cómo influye en la probabilidad de aparición de un suceso, la

presencia o no de diversos factores y el valor o nivel de los mismos. También puede ser usada para estimar la probabilidad de aparición de cada una de la posibilidad de un suceso con más de dos categorías.

CAPÍTULO 4

4. DISEÑO E IMPLEMENTACIÓN

Este capítulo está divido en dos secciones. La primera parte muestra la ejecucióndel algoritmo de aprendizaje `trainlogistic` usando regresión logística, cuya función es entrenar el modelo. En la segunda sección, se ejecuta el algoritmo `runlogistic` para evaluar cuan bueno es el modelo de prueba. Los algoritmos que se van a utilizar están escritos en java y su código fuente se adjuntan en los Anexos A y B. Los resultados de cada algoritmo se presentarán en el siguiente capítulo.

4.1. Entrenamiento del modelo

Después de la identificación de la variable objetivo, la selección adecuada de las variables de predicción y un algoritmo de aprendizaje, el siguiente paso en la formación de la solución es la ejecución del algoritmo de entrenamiento para producir un modelo. La Figura 4.1 captura la esencia de lo que el algoritmo de aprendizaje es capaz de discernir sobre la relación entre las variables predictivas y la variable objetivo.

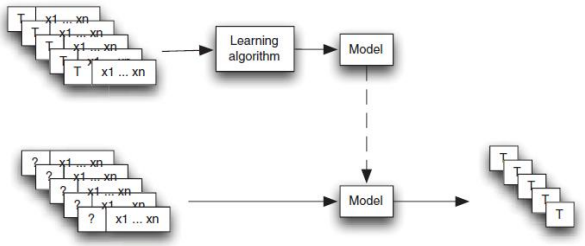

Figura 4.1– Arquitectura del algoritmo de aprendizaje.

Una vez que se ha descargado el paquete de Mahout de mahout.apache.org [4] es conveniente especificar el directorio de la librería en la variable MAHOUT_HOME, esto se realiza a través del siguiente comando:

```
$ export MAHOUT_HOME=/home/hadoop/mahout-distribution-
0.5
```

Es recomendable descargar la versión 0.5 de Mahout, para el correcto funcionamiento de los algoritmos.

Antes de ejecutar el algoritmo `trainlogistic`, es necesario convertir la base de datos en un archivo csv, el cual será recibido como el conjunto de entrada para entrenar el modelo. El archivo está compuesto de la primera que especifica los nombres de cada campo, seguido de los datos separados por coma. Para listar el contenido del archivo, se usa el comando `cat`:

```
$ bin/mahout cat /home/hadoop/baseClientes.csv
```

Este comando especifica que los datos de entrada se encuentran en el archivo baseClientes.csv y que el modelo resultante se almacenará en el archivo model. A través de la opción `--target` se especifica la variable objetivo, que en este caso es *churn*y el número de categorías que puede tomar. También se establece la variables predictivas a evaluar y el tipo de datos al que corresponde. El siguiente ejemplo solo se muestra algunas de las variables predictivas, para efecto de pruebas. Las demás opciones especifican parámetros internos del algoritmo de regresión logística.

```
$ bin/mahouttrainlogistic
--input /home/hadoop/baseClientes.csv
--output ./model
--target churn --categories 2
--predictorsAGE BANK_AGE SEX DEPENDENCY--types numeric
--features 16 --passes 100 --rate 20
```

El listado de todas las opciones para el algoritmo `trainlogistic` se detallan en la Tabla III.

Opción	Descripción
--input <archivo>	Usa un archivo específico como entrada.
--output <archivo>	Deposita el modelo resultante en un archivo específico.
--target <variable>	Usa una específica variable como objetivo.
--categories<n>	Especifica cuantas categorías tiene la variable objetivo.
--predictors<v1> …<vn>	Especifica los nombres de las variables predictivas.
--types<t1> …<tm>	Especifica el tipo de dato de las variables. Estas pueden ser numeric, word o text, para los tipos numéricos, categóricos o de texto, respectivamente.
--pases	Especifica el número de veces que el conjunto de entrada debe ser examinado durante el entrenamiento.
--rate	Establece el tamaño del vector de funciones internas para el uso en la construcción del modelo.

Tabla IIIII–Opciones para el algoritmo trainlogistic.

La salida del algoritmo es el modelo de entrenamiento, el cual consta de la variable objetivo *churn*y las variables predictivas, especificadas anteriormente, con sus respectivos coeficientes.

```
churn ~ 5.701*Intercept Term + -1.513*AGE +
7.885*BANK_AGE + 1.852*SEX + 5.701*DEPENDENCY
          Intercept Term  5.70054
          AGE             -1.51264
          BANK_AGE        7.88537
          SEX             1.85234
          DEPENDENCY      5.70054
```

4.2. Evaluación del modelo

Ahora que se ha entrenado un modelo, es esencial saber que tan bueno es antes de implementarlo. Para ellose debeevaluar la exactitud del modelo y hacer los ajustes que seannecesarios antes de comenzar la clasificación. Se procederá a correr el algoritmo `runlogistic` usando la misma base de datos para realizar la evaluación:

```
$ bin/mahoutrunlogistic
--input /home/hadoop/baseClientes.csv
--model ./model
--auc
```

La salida es el valor de AUC (un acrónimo de área bajo la curva), una medida ampliamente utilizada para determinar la calidad del modelo. El valor de AUC puede variar desde 0 para un modelo no muy

bueno, desde 0,5 para un modelo que es mejor que un aleatorio y a 1,0 para un modelo que es perfecto.
El valor para este ejemplo es de 0,56 e indica que modelo es apenas mejor que el aleatorio.

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS.

5.1. Ejecución de las pruebas

En este trabajo una colección de 4 diferentes modelos de regresión logística fueron entrenados yevaluados. El primer modelo corresponde únicamente a las variables predictivas de tipo demográficas, el cual se muestra como ejemplo en el capítulo anterior. El segundo modelo solo consideró las variables del comportamiento transaccional del cliente, en los tiempos t = 1 y t = 6. Los dos modelos siguientes consideran ambos tipos de variables.

Luego de ser entrenados los modelos, se procedió con la respectiva evaluación. La Tabla IV muestra los valores de AUC para los cuatro modelos resultantes:

	Modelo 1	**Modelo 2**	**Modelo 3**	**Modelo 4**
AUC	0.56	0.37	0.62	0.68

Tabla IV–Valores de AUC para cada modelo entrenado.

Como se puede apreciar en la tabla, los resultados de la evaluación de los dos primeros modelos, los cuales consideraban únicamente un

solo tipo de variables, no son muy significativos. De hecho, se puede concluir que las variables de tipo transaccional, por si solas, no describen correctamente el comportamiento de un cliente desertor. Para el problema planteado ambos modelos han sido descartados.

Los modelos 3 y 4 fueron los modelos que más alto AUC obtuvieron.Ambos modelos serán validados en la siguiente sección.

5.2. Análisis de los resultados

Los modelos entrenados de regresión logística, generarán un valor entre 0 y 1 como se mencionó en el capítulo 3. En la Tabla V el número de predicciones correctas se presenta para cada modelo. En la validación se utilizó una muestra de 1,000 clientes.

Modelo	Número correcto de predicción de clientes desertores	Porcentaje de clientes desertores del conjunto	Porcentaje predicho de clientes desertores
Modelo 3	794	83%	94%
Modelo 4	813	83%	97%

Tabla V–Validación de los modelos entrenados 3 y 4.

De acuerdo a los resultados de las validaciones, el modelo 4 es el que mayor porcentaje de aciertos obtuvo en la predicción. El modelo entrenado y validado es el siguiente:

```
churn ~ 3.284*Intercept Term + 0.369*NOPROD_01 + -
0.084*NOTRX_OUT_06 + 0.247*TRXCA_01 + -0.238*TRXCA_06 +
0.900*TRXCC_06 + 0.020*AGE + -0.285*SEX + -
0.727*DEPENDENCY
      Intercept Term  3.284
      NOPROD_01       0.369
      NOTRX_OUT_06    -0.084
      TRXCA_01        0.247
      TRXCA_06        -0.238
      TRXCC_06        0.900
      AGE             0.020
      SEX             -0.285
      DEPENDENCY      -0.727
```

CONCLUSIONES

Las conclusiones son:

1. Este trabajo presentó un análisis de deserción de clientes para el sector bancario. El análisis se centró en el entrenamiento de un modelo de predicción basado en regresión logística, proporcionado por los algoritmos de clasificación de la librería Mahout. Pese a que Hadoop está diseñado para el procesamiento masivo de datos, una de las limitantes del presente estudio fue la recolección de datos. El modelo fue entrenado con una pequeña muestra de datos y aunquelos dos modelos que fueron sometidos a validación predijeron relativamente bien a los clientes desertores sobre el conjunto de prueba. El valor de evaluación AUC, en ambos caso no fue de 1 para describir un modelo perfecto.

2. La definición de deserción de clientes en esta tesis se basó en el comportamiento transaccional de los clientes desertores. Para ello, se adoptó el terminó de cliente inactivo definido por el banco; el cual se sustenta en la inactividad de los productos y servicios que el cliente mantiene con la empresa. Pero, si la definición de deserción se desea basar en otros factores como: los reclamos o la rentabilidad del cliente, entonces el modelo se debería redefinir.

3. Una vez que el resultado del modelo ha llegado a unnivel aceptable de precisión, la clasificación denuevos datos puede comenzar. El rendimiento del sistema de clasificación en la producciónde dependerá de varios factores, uno de los más importantes es la calidad de los datos de entrada.Si los datos a que se analiza tiene errores en los valores de las variables de predicción, o si los nuevos datos no es un partido correspondiente alos datos de entrenamiento, o si las condiciones externas cambian con el tiempo, la calidad de la producción del modelo de clasificación se degradará. Con el fin de evitar este problema,pruebas periódicamente de la modelo es útil, y el reciclaje puede ser necesario.

RECOMENDACIONES

Las recomendaciones son:

1. Se recomienda tomar más muestras a través del tiempo con el fin de hacer más datos de entrenamiento, de modo que las versiones actualizadas del modelo se puedan producir. Estas muestras son importante si las condiciones externas cambian, ya que la calidad de las decisiones tomadas por el modelo se pueden degradar con el tiempo.En este caso, un nuevo modelo puede ser producido y la calidad de las decisiones puede ser mejorado.

2. Para implementar el modelo de predicción dado, se sugiere tomar los datos históricos de hasta seir meses de anterioridad, puesto que el análisis considera el comportamiento transaccional en un tiempo t = 1 (Inicio del periodo de evaluación) y t = 6 (Fin del periodo de evaluación). El objetivo es mantener la consistencia con los datos de entrada y prototipo del modelo predictivo.

3. El perfil de los clientes desertores no se entrega en el presente análisis. Sin embargo, se deja abierta una brecha para que se pueda estudiar y perfilar a los clientes que abandonan sus servicios y productos con el banco. Con ello, la empresa conocería que tipo de

clientes vale la pena o no retener. Y desde la perspectiva de marketing, que tipo de estrategias se podrían utilizar para retener a dichos clientes de acuerdo al perfil dado.

ANEXOS

ANEXO A

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * ALGORITMO: TRAINLOGISTIC
 *
 */

packageorg.apache.mahout.classifier.sgd;

importcom.google.common.collect.Lists;
importcom.google.common.io.Resources;
import org.apache.commons.cli2.CommandLine;
import org.apache.commons.cli2.Group;
import org.apache.commons.cli2.Option;
import org.apache.commons.cli2.builder.ArgumentBuilder;
import org.apache.commons.cli2.builder.DefaultOptionBuilder;
import org.apache.commons.cli2.builder.GroupBuilder;
import org.apache.commons.cli2.commandline.Parser;
import org.apache.commons.cli2.util.HelpFormatter;
importorg.apache.mahout.math.RandomAccessSparseVector;
importorg.apache.mahout.math.Vector;

importjava.io.BufferedReader;
importjava.io.FileReader;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.InputStreamReader;
importjava.io.OutputStreamWriter;
import java.net.URL;
importjava.util.List;


/**
 * Train a logistic regression for the examples from Chapter 13 of Mahout
in Action
```

```
 */
public final class TrainLogistic {

private static String inputFile;
private static String outputFile;
private static LogisticModelParameterslmp;

private static int passes;
private static boolean scores;
private static OnlineLogisticRegression model;

privateTrainLogistic() {
  }

public static void main(String[] args) throws IOException {
if (parseArgs(args)) {
doublelogPEstimate = 0;
int samples = 0;

CsvRecordFactorycsv = lmp.getCsvRecordFactory();
OnlineLogisticRegressionlr = lmp.createRegression();
for (int pass = 0; pass < passes; pass++) {
BufferedReader in = open(inputFile);

        // read variable names
csv.firstLine(in.readLine());

        String line = in.readLine();
while (line != null) {
          // for each new line, get target and predictors
          Vector input = new
RandomAccessSparseVector(lmp.getNumFeatures());
inttargetValue = csv.processLine(line, input);

          // check performance while this is still news
doublelogP = lr.logLikelihood(targetValue, input);
if (!Double.isInfinite(logP)) {
if (samples < 20) {
logPEstimate = (samples * logPEstimate + logP) / (samples + 1);
            } else {
logPEstimate = 0.95 * logPEstimate + 0.05 * logP;
            }
samples++;
          }
double p = lr.classifyScalar(input);
if (scores) {
System.out.printf("%10d %2d %10.2f %2.4f %10.4f %10.4f\n",
samples, targetValue, lr.currentLearningRate(), p, logP, logPEstimate);
          }

          // now update model
lr.train(targetValue, input);
```

```
line = in.readLine();
        }
in.close();
      }

OutputStreamWritermodelOutput = new FileWriter(outputFile);
try {
lmp.saveTo(modelOutput);
      } finally {
modelOutput.close();
      }

System.out.printf("%d\n", lmp.getNumFeatures());
System.out.printf("%s ~ ", lmp.getTargetVariable());
      String sep = "";
for (String v : csv.getPredictors()) {
double weight = predictorWeight(lr, 0, csv, v);
if (weight != 0) {
System.out.printf("%s%.3f*%s", sep, weight, v);
sep = " + ";
        }
      }
System.out.printf("\n");
model = lr;
for (int row = 0; row <lr.getBeta().numRows(); row++) {
for (String key : csv.getTraceDictionary().keySet()) {
double weight = predictorWeight(lr, row, csv, key);
if (weight != 0) {
System.out.printf("%20s %.5f\n", key, weight);
          }
        }
for (int column = 0; column <lr.getBeta().numCols(); column++) {
System.out.printf("%15.9f ", lr.getBeta().get(row, column));
        }
System.out.println();
      }
    }
  }

private static double predictorWeight(OnlineLogisticRegressionlr, int row,
RecordFactorycsv, String predictor) {
double weight = 0;
for (Integer column : csv.getTraceDictionary().get(predictor)) {
weight += lr.getBeta().get(row, column);
    }
return weight;
  }

private static booleanparseArgs(String[] args) {
DefaultOptionBuilder builder = new DefaultOptionBuilder();
```

```
    Option help = builder.withLongName("help").withDescription("print this
list").create();

    Option quiet = builder.withLongName("quiet").withDescription("be extra
quiet").create();
    Option scores = builder.withLongName("scores").withDescription("output
score diagnostics during training").create();

ArgumentBuilderargumentBuilder = new ArgumentBuilder();
    Option inputFile = builder.withLongName("input")
            .withRequired(true)

.withArgument(argumentBuilder.withName("input").withMaximum(1).create())
            .withDescription("where to get training data")
            .create();

    Option outputFile = builder.withLongName("output")
            .withRequired(true)

.withArgument(argumentBuilder.withName("output").withMaximum(1).create())
            .withDescription("where to get training data")
            .create();

    Option predictors = builder.withLongName("predictors")
            .withRequired(true)
            .withArgument(argumentBuilder.withName("p").create())
            .withDescription("a list of predictor variables")
            .create();

    Option types = builder.withLongName("types")
            .withRequired(true)
            .withArgument(argumentBuilder.withName("t").create())
            .withDescription("a list of predictor variable types (numeric,
word, or text)")
            .create();

    Option target = builder.withLongName("target")
            .withRequired(true)

.withArgument(argumentBuilder.withName("target").withMaximum(1).create())
            .withDescription("the name of the target variable")
            .create();

    Option features = builder.withLongName("features")
            .withArgument(
argumentBuilder.withName("numFeatures")
                            .withDefault("1000")
                            .withMaximum(1).create())
            .withDescription("the number of internal hashed features to
use")
            .create();
```

```
    Option passes = builder.withLongName("passes")
            .withArgument(
argumentBuilder.withName("passes")
                            .withDefault("2")
                            .withMaximum(1).create())
            .withDescription("the number of times to pass over the input
data")
            .create();

    Option lambda = builder.withLongName("lambda")

.withArgument(argumentBuilder.withName("lambda").withDefault("1e-
4").withMaximum(1).create())
            .withDescription("the amount of coefficient decay to use")
            .create();

    Option rate = builder.withLongName("rate")

.withArgument(argumentBuilder.withName("learningRate").withDefault("1e-
3").withMaximum(1).create())
            .withDescription("the learning rate")
            .create();

    Option noBias = builder.withLongName("noBias")
            .withDescription("don't include a bias term")
            .create();

    Option targetCategories = builder.withLongName("categories")
            .withRequired(true)

.withArgument(argumentBuilder.withName("number").withMaximum(1).create())
            .withDescription("the number of target categories to be
considered")
            .create();

    Group normalArgs = new GroupBuilder()
            .withOption(help)
            .withOption(quiet)
            .withOption(inputFile)
            .withOption(outputFile)
            .withOption(target)
            .withOption(targetCategories)
            .withOption(predictors)
            .withOption(types)
            .withOption(passes)
            .withOption(lambda)
            .withOption(rate)
            .withOption(noBias)
            .withOption(features)
            .create();

    Parser parser = new Parser();
```

```
parser.setHelpOption(help);
parser.setHelpTrigger("--help");
parser.setGroup(normalArgs);
parser.setHelpFormatter(new HelpFormatter(" ", "", " ", 130));
CommandLinecmdLine = parser.parseAndHelp(args);

if (cmdLine == null) {
return false;
    }

TrainLogistic.inputFile = getStringArgument(cmdLine, inputFile);
TrainLogistic.outputFile = getStringArgument(cmdLine, outputFile);

    List<String>typeList = Lists.newArrayList();
for (Object x : cmdLine.getValues(types)) {
typeList.add(x.toString());
    }

    List<String>predictorList = Lists.newArrayList();
for (Object x : cmdLine.getValues(predictors)) {
predictorList.add(x.toString());
    }

lmp = new LogisticModelParameters();
lmp.setTargetVariable(getStringArgument(cmdLine, target));
lmp.setMaxTargetCategories(getIntegerArgument(cmdLine, targetCategories));
lmp.setNumFeatures(getIntegerArgument(cmdLine, features));
lmp.setUseBias(!getBooleanArgument(cmdLine, noBias));
lmp.setTypeMap(predictorList, typeList);

lmp.setLambda(getDoubleArgument(cmdLine, lambda));
lmp.setLearningRate(getDoubleArgument(cmdLine, rate));

TrainLogistic.scores = getBooleanArgument(cmdLine, scores);
TrainLogistic.passes = getIntegerArgument(cmdLine, passes);

return true;
  }

private static String getStringArgument(CommandLinecmdLine, Option
inputFile) {
return (String) cmdLine.getValue(inputFile);
  }

private static booleangetBooleanArgument(CommandLinecmdLine, Option option)
{
returncmdLine.hasOption(option);
  }

private static intgetIntegerArgument(CommandLinecmdLine, Option features) {
returnInteger.parseInt((String) cmdLine.getValue(features));
  }
```

```
private static double getDoubleArgument(CommandLinecmdLine, Option op) {
returnDouble.parseDouble((String) cmdLine.getValue(op));
  }

public static OnlineLogisticRegressiongetModel() {
return model;
  }

public static LogisticModelParametersgetParameters() {
returnlmp;
  }

staticBufferedReader open(String inputFile) throws IOException {
InputStreamReader s;
try {
      URL resource = Resources.getResource(inputFile);
      s = new InputStreamReader(resource.openStream());
    } catch (IllegalArgumentException e) {
      s = new FileReader(inputFile);
    }
return new BufferedReader(s);
  }
}
```

ANEXO B

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * ALGORITMO: RUNLOGISTIC
 *
*/

packageorg.apache.mahout.classifier.sgd;

import org.apache.commons.cli2.CommandLine;
import org.apache.commons.cli2.Group;
import org.apache.commons.cli2.Option;
import org.apache.commons.cli2.builder.ArgumentBuilder;
import org.apache.commons.cli2.builder.DefaultOptionBuilder;
import org.apache.commons.cli2.builder.GroupBuilder;
import org.apache.commons.cli2.commandline.Parser;
import org.apache.commons.cli2.util.HelpFormatter;
importorg.apache.mahout.math.Matrix;
importorg.apache.mahout.math.SequentialAccessSparseVector;
importorg.apache.mahout.math.Vector;
importorg.apache.mahout.classifier.evaluation.Auc;

importjava.io.BufferedReader;
importjava.io.File;
importjava.io.IOException;

public final class RunLogistic {

private static String inputFile;
private static String modelFile;
private static booleanshowAuc;
private static booleanshowScores;
private static booleanshowConfusion;

privateRunLogistic() {
```

```
  }

public static void main(String[] args) throws IOException {
if (parseArgs(args)) {
if (!showAuc&& !showConfusion&& !showScores) {
showAuc = true;
showConfusion = true;
      }

Auc collector = new Auc();
LogisticModelParameterslmp = LogisticModelParameters.loadFrom(new
File(modelFile));

CsvRecordFactorycsv = lmp.getCsvRecordFactory();
OnlineLogisticRegressionlr = lmp.createRegression();
BufferedReader in = TrainLogistic.open(inputFile);
      String line = in.readLine();
csv.firstLine(line);
line = in.readLine();
if (showScores) {
System.out.printf("\"%s\",\"%s\",\"%s\"\n", "target", "model-output", "log-
likelihood");
      }
while (line != null) {
        Vector v = new SequentialAccessSparseVector(lmp.getNumFeatures());
int target = csv.processLine(line, v);
double score = lr.classifyScalar(v);
if (showScores) {
System.out.printf("%d,%.3f,%.6f\n", target, score, lr.logLikelihood(target,
v));
        }
collector.add(target, score);
line = in.readLine();
      }

if (showAuc) {
System.out.printf("AUC = %.2f\n", collector.auc());
      }
if (showConfusion) {
        Matrix m = collector.confusion();
System.out.printf("confusion: [[%.1f, %.1f], [%.1f, %.1f]]\n",
m.get(0, 0), m.get(1, 0), m.get(0, 1), m.get(1, 1));
        m = collector.entropy();
System.out.printf("entropy: [[%.1f, %.1f], [%.1f, %.1f]]\n",
m.get(0, 0), m.get(1, 0), m.get(0, 1), m.get(1, 1));
      }
    }
  }

private static booleanparseArgs(String[] args) {
DefaultOptionBuilder builder = new DefaultOptionBuilder();
```

```
    Option help = builder.withLongName("help").withDescription("print this
list").create();

    Option quiet = builder.withLongName("quiet").withDescription("be extra
quiet").create();

    Option auc = builder.withLongName("auc").withDescription("print
AUC").create();
    Option confusion =
builder.withLongName("confusion").withDescription("print confusion
matrix").create();

    Option scores = builder.withLongName("scores").withDescription("print
scores").create();

ArgumentBuilderargumentBuilder = new ArgumentBuilder();
    Option inputFileOption = builder.withLongName("input")
            .withRequired(true)

.withArgument(argumentBuilder.withName("input").withMaximum(1).create())
            .withDescription("where to get training data")
            .create();

    Option modelFileOption = builder.withLongName("model")
            .withRequired(true)

.withArgument(argumentBuilder.withName("model").withMaximum(1).create())
            .withDescription("where to get a model")
            .create();

    Group normalArgs = new GroupBuilder()
            .withOption(help)
            .withOption(quiet)
            .withOption(auc)
            .withOption(scores)
            .withOption(confusion)
            .withOption(inputFileOption)
            .withOption(modelFileOption)
            .create();

    Parser parser = new Parser();
parser.setHelpOption(help);
parser.setHelpTrigger("--help");
parser.setGroup(normalArgs);
parser.setHelpFormatter(new HelpFormatter(" ", "", " ", 130));
CommandLinecmdLine = parser.parseAndHelp(args);

if (cmdLine == null) {
return false;
    }

inputFile = getStringArgument(cmdLine, inputFileOption);
```

```
modelFile = getStringArgument(cmdLine, modelFileOption);
showAuc = getBooleanArgument(cmdLine, auc);
showScores = getBooleanArgument(cmdLine, scores);
showConfusion = getBooleanArgument(cmdLine, confusion);

return true;
  }

private static booleangetBooleanArgument(CommandLinecmdLine, Option option)
{
returncmdLine.hasOption(option);
  }

private static String getStringArgument(CommandLinecmdLine, Option
inputFile) {
return (String) cmdLine.getValue(inputFile);
  }

                                        }
```

BIBLIOGRAFÍA

[1] TeemuMutanen, Customer Churn Analysis – a case study. Research report VTT-R-01184-0, VTT Information Technology (2006).

[2] K. Chitra, B. Subashini, Customer Retention in Banking Sector using Predictive Data Mining Technique. ICIT 2011 The 5th International Conference on Information Technology (2011)

[3] J. Ma. Santiago Merino, Factores de protección y riesgo de infidelidad en la banca comercial. Universidad Complutense Madrid (2008)

[4] Apache Hadoop, Sistema Distribuido Hadoop, http://hadoop.apache.org,[Citado el: 8 de Diciembre de 2011.]

[5]Wikipedia, MapReduce, http://en.wikipedia.org/wiki/MapReduce,[Citado el: 5 de Diciembre de 2011.]

[6] Jeffrey Dean, Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters. (2004)

[7] Owen Sean, Robin Anil. Mahout in Action.(2005)

[8] Cramer J.S. The Logit Model: An Introduction. Edward Arnold (1991)