Análisis de datos de accidentes de tráfico mediante soluciones BigData y Business Intelligence

Marc Alvarez Brotons
Ingeniería Informática

David Isern Alarcón

27/12/2014

Tabla de contenidos:

1. Introducción 4
1.1 Contexto 4
1.2 Objetivos 4
1.3 Productos 4
1.4 Planificación 5
1.5 Contenido de este documento 5
2. Arquitectura de la solución global 6
3. Solución BigData 7
3.1 Logistic Regresion 9
4. Solución Business Intelligence 16
4.1 Análisis de datos de tráfico 18
4.2 Análisis de datos climatológicos 21
4.3 Análisis de datos cruzados 21
4.4 Análisis de datos predictivos 23
5. Análisis de resultados y conclusiones 24
6. Bibliografía 25
7. Anexos 26
7.1 Código Solución BigData 26
7.1.1 Instrucciones ejecución algoritmos 26
7.1.2 LoadFilesIA.java 27
7.1.3 LoadFilesDWH.java 39
7.2 Código Solución Business Intelligence 52

Tabla de figuras:

Figura 1. Tabla tecnologías Big Data 4
Figura 2. Tabla tecnologías Business Intelligence 4
Figura 3. Planificación del PFC....5
Figura 4. Arquitectura de la solución global 6
Figura 5. Big data 7
Figura 6. Procesos de incorporación de datos para los procesos de predicción....8
Figura 7. Funcionamiento del algoritmo de clasificación 10
Figura 8. Accidentes 2012 11
Figura 9. Accidentes 2012 vías interurbanas 12
Figura 10. Proceso de incorporación de los datos al repositorio HDFS 16
Figura 11. Diagrama modelo de datos lógico QlikView....17
Figura 12. Relación accidentes, muertos y heridos por provincia....19
Figura 13. Accidentes por tipo de vía 19
Figura 14. Información accidentes Barcelona....19
Figura 15. Información accidentes Madrid 20
Figura 16. Relación accidentes, muertos y heridos por provincia descartando Barcelona y Madrid 20
Figure 17. Accidentes por tipo de vía excluyendo Barcelona y Madrid 20
Figura 18. Ranking de accidentes por provincias 21
Figura 19. Análisis de información climatológica excluyendo a Barcelona y Madrid....21
Figura 20. Humedad media, temperatura mínima y accidentes 22
Figura 21. Precipitación total, temperatura mínima y accidentes 22
Figura 22. Precipitación total, horas con el cielo despejado y accidentes 23
Figura 23. Resultados categorización....23

1. Introducción

1.1 Contexto

Este proyecto de final de carrera (a partir de ahora PFC) corresponde al área de Inteligencia Artificial y representa un caso de uso que pretende utilizar datos reales referentes a accidentes de tráfico (datos de accidentes, muertos, heridos, etc.) y analizarlas conjuntamente con datos que puedan tener una posible relación con los accidentes como el parque de vehículos, las temperaturas de la zona de los accidentes, etc. con la finalidad de poder obtener las posibles relaciones causa - efecto.

1.2 Objetivos

El actual PFC tiene los siguientes objetivos:

- Definir de una solución tecnológica que disponga de la arquitectura que dé cobertura a todos los ámbitos del PFC
- Construir una solución tecnológica que permita analizar la información y posibilitar una futura toma de decisiones.

1.3 Productos

Durante el PFC se utilizar distintos productos relacionados con los dos principales ámbitos tecnológicos:

BigData

cloudera hadoop	Distribución del framework Apache Hadoop, el cual soporta aplicaciones distribuidas bajo una licencia Open Source y que permite a las aplicaciones trabajar con miles de nodos y altas volumetrías de datos.
mahout	Proyecto de Apache Software Fundation que permite la implementación de algoritmos de máquinas de aprendizaje destinados al filtrado, agrupación o clasificación de datos de forma distribuida, combinado con las capacidades de Hadoop.

Figura 1. Tabla tecnologías Big Data

Business Intelligence

Qlik	**QlikView Personal Edition** (http://www.qlik.com). Herramienta de Business Intelligence destinada al Business Discovery y que está utiliza la lógica asociativa para el realizar la explotación de los datos.

Figura 2. Tabla tecnologías Business Intelligence

1.4 Planificación

La planificación del proyecte se constituye por 5 fases diferenciadas por su naturaleza:

- Fase 0. Desarrollo de las tareas iniciales para la definición del PFC
- Fase 1. Definición de la solución a desarrollar en el PFC
- Fase 2. Construcción de la solución
- Fase 3. Evaluación de los datos para determinar posibles patrones
- Fase 4. Redacción de la memoria

PFC – Análisis de datos de tránsito	**9/17/14 8:00 AM**	**1/2/15 5:00 PM**
Fase 0: tareas previas	**9/17/14 8:00 AM**	**10/1/14 8:00 AM**
Investigación casos de uso	9/17/14 8:00 AM	9/24/14 5:00 PM
Definición alcance alto nivel	9/25/14 8:00 AM	9/29/14 5:00 PM
Planificación	9/30/14 8:00 AM	9/30/14 5:00 PM
PAC 1	10/1/14 8:00 AM	10/1/14 8:00 AM
Fase 1: Definición solución tecnológica	**10/1/14 8:00 AM**	**10/28/14 5:00 PM**
Componentes tecnológicos	10/1/14 8:00 AM	10/3/14 5:00 PM
Arquitectura de la solución	10/6/14 8:00 AM	10/7/14 5:00 PM
Definición fuentes de datos	10/8/14 8:00 AM	10/14/14 5:00 PM
Definición proceso de datos	10/15/14 8:00 AM	10/21/14 5:00 PM
Definición explotación de datos	10/22/14 8:00 AM	10/28/14 5:00 PM
Fase 2: Construcción	**10/29/14 8:00 AM**	**12/16/14 5:00 PM**
Entorno	**10/29/14 8:00 AM**	**10/31/14 5:00 PM**
Configuración entorno de desarrollo	10/29/14 8:00 AM	10/29/14 5:00 PM
Configuración arquitectura	10/30/14 8:00 AM	10/31/14 5:00 PM
Desarrollo	**11/3/14 8:00 AM**	**12/4/14 5:00 PM**
Carga de datos en Hadoop	11/3/14 8:00 AM	11/10/14 5:00 PM
Clasificación de datos Mahaut	11/11/14 8:00 AM	11/24/14 5:00 PM
Carga de datos BI	11/25/14 8:00 AM	11/27/14 5:00 PM
Construcción cuadros de mando	11/28/14 8:00 AM	12/4/14 5:00 PM
Pruebas	**12/5/14 8:00 AM**	**12/16/14 5:00 PM**
Carga	12/5/14 8:00 AM	12/5/14 5:00 PM
Clasificación	12/8/14 8:00 AM	12/9/14 5:00 PM
Explotación	12/10/14 8:00 AM	12/16/14 5:00 PM
Fase 3: Evaluación datos	**12/17/14 8:00 AM**	**1/2/15 5:00 PM**
Test de la solución	12/17/14 8:00 AM	12/30/14 5:00 PM
Redacción conclusiones	12/31/14 8:00 AM	1/2/15 5:00 PM
Fase 4: Memoria	**10/1/14 8:00 AM**	**1/2/15 5:00 PM**
Elaboración memoria	10/1/14 8:00 AM	1/2/15 5:00 PM
Entrega final	1/9/15 8:00 AM	1/9/15 8:00 AM

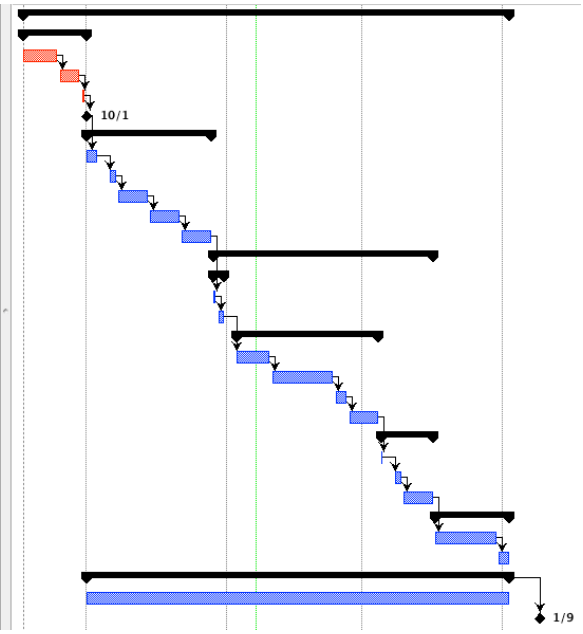

Figura 3. Planificación del PFC

1.5 Contenido de este documento

El actual documento está dividido en cuatro apartados donde se detalla los distintos ámbitos que conforman la solución del PFC:

- Arquitectura de la solución. Se realiza una descripción general de la solución definida e implementada así como de la utilización e integración de las distintas tecnologías Big data y Business Intelligence.
- Solución Big Data. Se describe la configuración de la plataforma Hadoop Cloudera, para el almacenamiento de los datos, así como su procesado y clasificación mediante algoritmos de máquinas de aprendizaje.
- Solución Business Intelligence. Se describe la configuración de la herramienta QlikView y desarrollo de cuadros de mando para el análisis de los datos.
- Análisis de resultados y conclusiones. Se realiza un análisis de los datos almacenados y procesados con el objetivo de determinar patrones que identifique la relación causa efecto.

2. Arquitectura de la solución global

El objetivo de la arquitectura propuesta es el de ofrecer un entorno integrado para los mundos de las tecnologías BigData y Business Intelligence. La primera de ellas ofrecerá a la solución los servicios de almacenamiento masivo de información así como los mecanismos de predicción mediante algoritmos de clasificación. La segunda, permitirá analizar la información tratada así como los resultados predictivos, permitiendo un toma de decisión con ellos, así como la optimización de las futuras predicciones.

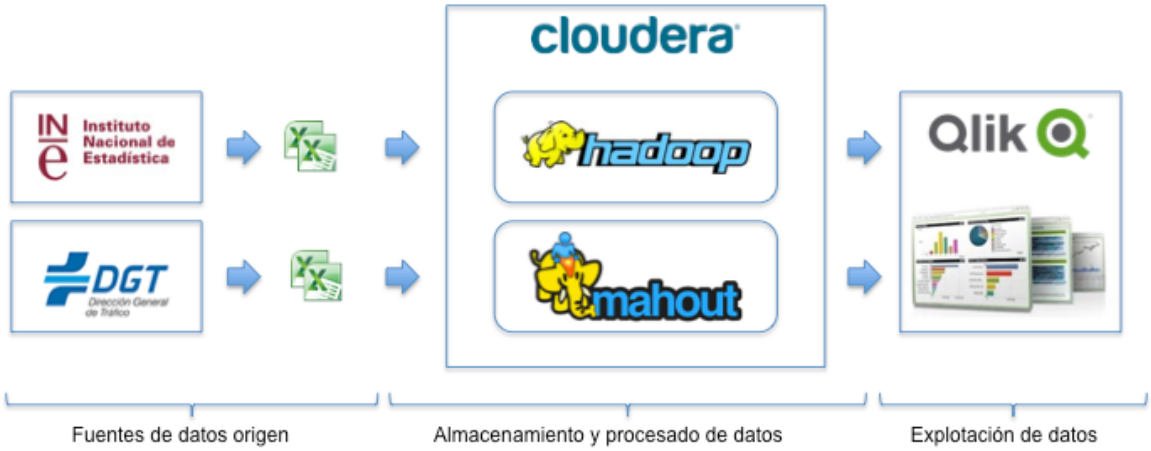

Figura 4. Arquitectura de la solución global

La solución propuesta se basa en tres grandes pilares, tal y como se muestra el diagrama de la Figura 4:

- Fuentes origen. Datos procedentes de dos fuentes de datos estadísticos referentes a accidentes de tráfico (Dirección General de Tráfico) así como de climatología (Instituto Nacional de Estadística) de las diferentes regiones del territorio español.
- Componentes Big Data. Elementos que permitirán almacenar y procesar los datos:
  - Mediante HDFS (Hadoop Distributed File System) se almacenarán todos los datos necesarios para el análisis.
  - Mediante los algoritmos de Mahout se procesarán los datos para poder detectar posibles patrones mediante la clasificación de estos.
- Componentes Business Intelligence. Mediante la herramienta QlikView se analizarán los datos.

3. Solución BigData

BigData es un conjunto de tecnologías que permiten a las organizaciones almacenar y analizar ingentes volúmenes de información en tiempos de respuesta bajos.

Figura 5. Big data

En base a los tres ejes que giran alrededor del concepto Big data, se plantea la posibilidad de poder gestionar las necesidades existentes en el mundo actual:

- gran variedad de datos a almacenar y analizar, procedentes en gran medida de sistemas ajenos a las organizaciones y con formatos heterogéneos y no estructurados, como por ejemplo posts de redes sociales o foros, información multimedia como imágenes, vídeos o locuciones.
- una evolución exponencial de los datos generados cada año.
- la información se genera con más velocidad y los análisis que se requieren por las organizaciones se han de realizar en el menor tiempo posible o en tiempo real, para tomar decisiones que aporten una ventaja competitiva

El software utilizado para la construcción del piloto es la distribución de Cloudera para Apache Hadoop ya instalada y configurada en una máquina virtual distribuida por Cloudera (Máquina virtual cloudera-quickstart-vm-5.1.0-1). Dicha distribución está soportada baso un sistema operativo Linux versión CentOS 6.2.

Componentes:

- CDH (Cloudera Distribution Including Apache Hadoop) versión 5.1.3
- Hadoop Cliente versión 2.0
- Hadoop HDFS versión 2.3
- Mahout versión 0.9

En base al software seleccionado, se procederá a la incorporación de los datos estadísticos procedentes del Instituto Nacional de Estadística (en adelante INE) así como de la Dirección General de Tráfico (en adelante DGT) al repositorio HDFS.

Instituto Nacional de Estadística:
http://www.ine.es/inebmenu/mnu_entornofis.htm

Dirección General de Tráfico:
http://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/

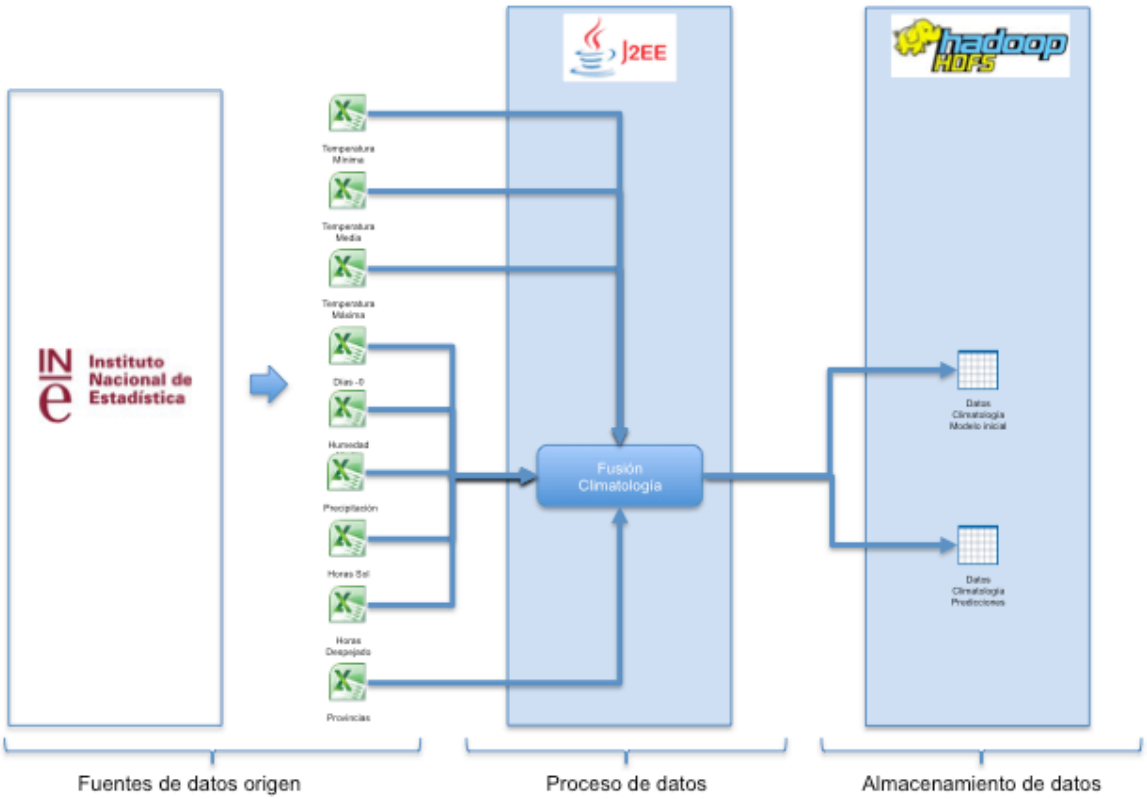

Figura 6. Procesos de incorporación de datos para los procesos de predicción

El diagrama de la figura 6 describe el flujo de datos en la incorporación de los datos procedentes del INE referentes a la climatología con la finalidad de ser incorporados en el repositorio HDFS con la finalidad de ser utilizados por los algoritmos de predicción. De igual modo que los datos con fines analíticos, la incorporación se realiza mediante un proceso desarrollado con el lenguaje de programación JAVA que consolida la información en un único registro.

El objetivo del proceso es el de fusionar la información procedente de datos climatológicos hasta conseguir una estructura de datos que represente las futuras variables predictivas y la clase base para la clasificación. La estructura de datos consta de las siguientes variables:

- Identificador de provincia
- Nombre de la provincia
- Año correspondiente a los datos del fichero

- Temperatura máxima
- Temperatura media
- Temperatura mínima
- Número de días con temperatura igual o menor a cero grados
- Precipitación total
- Humedad media
- Número de días con el cielo despejado
- Horas de sol
- Clase que clasifica en riesgo alto o bajo de accidente a causa del clima

Es importante en primer lugar carga la tabla maestra de provincias ya que los datos climatológicos no están normalizados. Una vez cargadas las provincias, se cargarán fichero a fichero los datos climatológicos y utilizando un método de búsqueda, se reemplazará la provincia descrita por el código correspondiente y se almacenará en la estructura de datos por provincia.
Una vez cargados los datos, se generará un fichero de salida almacenado en el repositorio HDFS.
Hay que distinguir que el mismo proceso se utiliza para dos fines distintos:

- una ejecución inicial con datos representativos (subset de los datos totales) para generar un pequeño fichero que servirá como información inicial para la creación del modelo de aprendizaje. Dicha selección, descrita en el siguiente sección, es muy importante ya que de ella depende el éxito del modelo generado. En este caso se ha determinado utilizar 45 registros con información de algunas provincias del año 2012.
- siguientes iteraciones para incorporar tanta información actual o histórica como queramos para procesar por el algoritmo de clasificación. En este caso se ha utilizado información histórica de todas las provincias desde el año 2007 al año 2012, lo que representan 342 registros.

El mismo proceso está preparado para procesar una mayor cantidad de registros cambiando el periodo a nivel mes o incluso día, lo que refinaría el detalle climatológico y como consecuencia el correspondiente modelo de aprendizaje generado.

Una vez ya disponibles los datos en el repositorio distribuido de Hadoop (HDFS), se procesan mediante el algoritmo de clasificación que nos ofrece el proyecto Mahout: Logistic Regresion.

3.1 Logistic Regresion

Logistic Regresion es un algoritmo destinado a la predicción de la probabilidad de la ocurrencia de un evento concreto, mediante la utilización de variables predictivas que pueden ser tanto numéricas como categorías alfanuméricas. La distribución ofrecida de Mahout, utiliza el método Stochastic Gradient Descent para el entrenamiento de modelos en casos con alta volumetría de datos. Trabaja de forma secuencial, no en paralelo, pero que permite procesar rápidamente y con bajo coste de cálculo.

Dadas sus características, el algoritmo ofrece dos posibilidades:

- Determinar numéricamente la importancia de la relación existente entre las variables predictivas y la variable a predecir (o variable objetivo), así como el nivel de confusión entre dichas variables.
- Clasificar datos según la probabilidad de pertenecer a la variable objetivo en base a sus propias variable predictivas.

Al tratarse de un algoritmo de clasificación, su funcionamiento está basado en un aprendizaje previo mediante una muestra de los datos a analizar que se utilizará de ejemplo para que el algoritmo pueda aprender y generar consecuentemente un modelo de aprendizaje. En base a este modelo construido, se podrán procesar nuevos datos para obtener las predicciones deseadas.

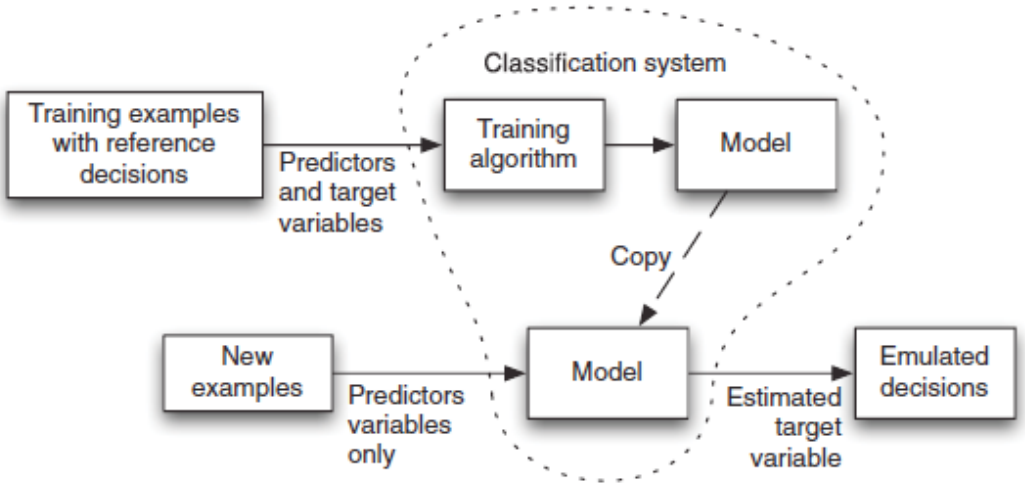

Figura 7. Funcionamiento del algoritmo de clasificación

En base a la descripción anterior del funcionamiento de los algoritmos de clasificación, el algoritmo Logistic Regresion consta de dos fases: una para llevar a cabo el entreno y otra para la validación del modelo. Para el entreno del modelo se dispone del ejecutable *tranlogistic*, y para la validación de este en base al modelo construido, se dispone del ejecutable *runlogistic*.

El programa *trainlogistic* dispone de las siguientes opciones de ejecución:

-- help	Muestra la ayuda
-- quiet	Reduce la información de salida en cada ejecución
-- input <input>	Fichero con la información de entrada para construir el modelo
-- output <output>	Fichero donde se almacenará el modelo
-- target <target>	Variable marcada como objetivo
-- categories <numbers>	Número de categorías utilizadas por la variable objetivo
-- predictors <p1> [<p2> …]	Lista de los nombres dela variables predictivas
-- types <t1> [<t2> …]	Lista de las tipologías de variables predictivas (numérica, alfanumérica o texto)
-- passes <passes>	Número de veces que los datos introducidos serán revisados durante el entreno
-- lambda <lambda>	Parámetro que controla la eliminación de variables del modelo final en caso de ser necesario. En caso de ser cero no realiza esfuerzo
-- rate <learningRate>	Tasa de aprendizaje inicial. Va disminuyendo progresivamente a medida que va examinando los datos.
-- noBias	Elimina la constante del modelo
-- features <numFeatures>	Tamaño del vector de características utilizado para construir el modelo

El programa *runlogistic* dispone de las siguientes opciones de ejecución:

-- help	Muestra la ayuda
-- quiet	Reduce la información de salida en cada ejecución
-- input <input>	Fichero con los datos a procesar
-- model <model>	Lee el modelo desde el fichero específico
-- auc	Retorna el resultado del AUC (área debajo de la curva) del modelo respecto de

	los datos procesados. (0: totalmente aleatorio, 1:alto nivel de acierto en la predicción)
--threshold <t>	Ajusta el umbral para el cálculo para la matriz de confusión en t (0,5 por defecto)
-- confusión	Retorna la matriz de confusión de los datos procesados

Los datos a utilizar para el aprendizaje y la predicción será información procedente del Instituto Nacional de Estadística referente a la climatología de las provincias de España:

- Tmedia: Temperatura media
- Tmin: Temperatura mínima
- Tmax: Temperatura máxima
- Hmedia: Humedad media
- Ptotal: Precipitación total
- Dias-0: Número de días con temperaturas inferiores a cero grados
- Ddes: Días con el cielo destapado
- Hsol: Horas de sol

Una vez se dispone de la variables predictivas, el siguiente paso es seleccionar la variable marcada como objetivo (target) y asignarle un valor coherente para los datos iniciales de aprendizaje. En este caso la variable objetivo será el nivel de posibles accidentes y tendrá dos posibles valores: alto y bajo.
Para poder asignar estos valores a los datos climatológicos a nivel de provincia, se deben analizar los datos referentes a los accidentes.

En un primer análisis podemos determinar como se observa en la figura 8 que Barcelona y Madrid tienen un tasa muy alta de accidentes dada la alta volumetría de vehículos existentes y el consecuente tráfico ocasionado. De igual modo podemos observar que Melilla también tiene una tasa muy alta de accidentes, pero en este caso respecto al volumen de vehículos que tiene. Este alto nivel de accidentes es debido a los vehículos de paso entre la península y África, que no deben ser considerados como afectación climatológica, sino como al aumento de tránsito.

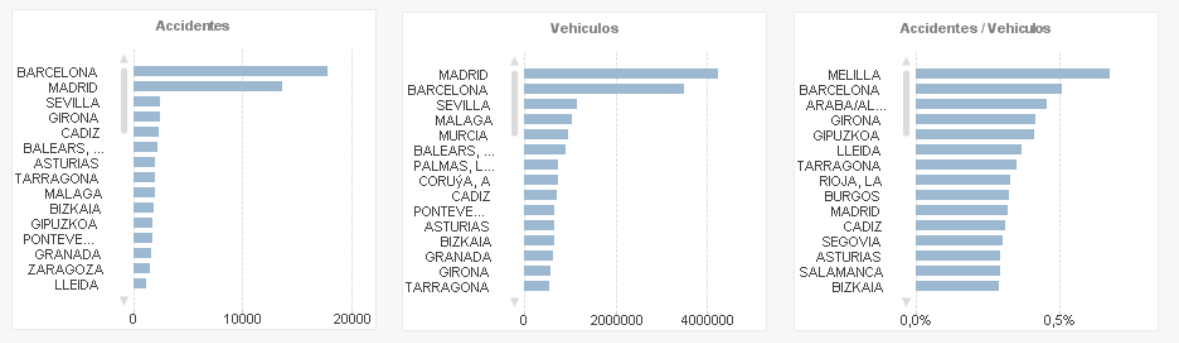

Figura 8. Accidentes 2012

Una vez excluidas las dos provincias del análisis, realizaremos un nuevo filtro para poder disponer de aquellos accidentes que se den lugar en vías interurbanas, que es donde más afectación climatológica puede haber. Esto nos llevará a la lista de provincias mostrada en la figura 9, con unas posibles candidatas para categorizar como provincias con alto riesgo de accidente.

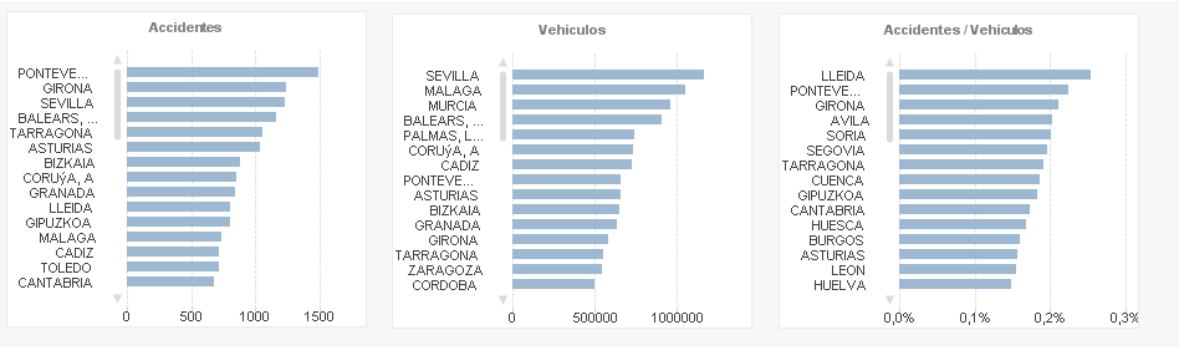

Figura 9. Accidentes 2012 vías interurbanas

Para finalizar y refinar un poco la categorización se pueden eliminar de la lista de las más afectadas, provincias que por el alto grado de afluencia turística, el nivel de accidentes se vea afectado y distorsione la probabilidad de accidentes por causas del clima (véase Baleares y Girona)

Una vez definidas las variables predictivas y la objetivo, y para la ejecución del primer paso del algoritmo requeriremos del fichero clima_train donde reside la información climatológica por provincia y año 2012, y que utilizaremos para entrenar el modelo.

Para poder ejecutar el algoritmo debemos acceder al sistema con el usuario cloudera que por defecto tiene configurada la variable PATH para poder ejecutar cualquier programa desde la ubicación que se desee.

```
$mahout trainlogistic --input /home/cloudera/ia/clima_train --output /home/cloudera/ia/model --target Clase --categories 2 --predictors Provincia Tmedia Tmin Hmedia Hdes Ptotal Dias-0 Hsol --types numeric --features 114 --passes 1000 --rate 80
```

Una vez construido y entrenado el modelo lo validaremos con las misma información para determinar el nivel de madurez de este, tanto por el valor de evaluación auc que determinará mediante el rango de 0 a 1 el nivel de madurez siendo 0 el nivel de aleatoriedad máximo, y 1 el máximo nivel de certeza en la predicción, así como la matriz de confusión que nos ayudará a determinar el grado de acierto por cada una de las dos posibilidades de predicción:

```
$mahout runlogistic --input /home/cloudera/ia/clima_train --model /home/cloudera/ia/model --auc --confusion
```

Seguidamente, se evaluará el modelo realizando la predicción con nueva información y de igual modo, obteniendo el nivel de predicción. Esta nueva información climatológica será de los años 2007 al 2012.

```
$mahout runlogistic --input /home/cloudera/ia/clima_run --model /home/cloudera/ia/model --auc --confusion
```

Por último, se obtendrán la predicciones para su futura incorporación al datawarehouse y explotación con QlikView.

```
$mahout runlogistic --input /home/cloudera/ia/clima_run --model /home/cloudera/ia/model --scores
```

Dado que el modelo a utilizar no es obvio y requiere de una selección de las variables predictivas y comprobación en base a los resultados obtenidos, detallaremos algunos de los resultados obtenidos por cada uno de los modelos seleccionados.

Modelo 1
Variables predictivas utilizadas:

- Tmedia
- Tmin
- Tmax
- Hmedia
- Ptotal
- Dias-0
- Ddes
- Hsol

```
$mahout trainlogistic --input /home/cloudera/ia/clima --output /home/cloudera/model --target Clase --categories 2 --predictors Provincia Tmedia Tmin Tmax Hmedia Ptotal Dias-0 Ddes Hsol --types numeric --features 200 --passes 1000 --rate 80 --lambda 0.00001
```

Resultados trainlogistic:

```
$mahout runlogistic --input /home/cloudera/ia/clima --model /home/cloudera/model --auc –confusion
```

- AUC: 0,61
- Matriz de confusión: [[29.0, 6.0], [6.0, 4.0]]

Resultados runlogistic:

```
$mahout runlogistic --input /home/cloudera/ia/clima_run --model /home/cloudera/model --auc –confusion
```

- AUC: 0,50
- Matriz de confusión: [[225.0, 49.0], [39.0, 29.0]]

Valoración:
En esta primera ejecución, se han utilizado todas las variables disponibles, siendo el resultado de entrenamiento del 0,61 y aportando un grado bajo de predicción sobre los datos iniciales.
En cambio, al ejecutar el algoritmo con el modelo generado, con el resto de datos, los resultados ya no son tan positivos ya que son aleatorios y todas las predicciones quedan a un nivel muy bajo.

Modelo 2
Variables predictivas utilizadas:

- Tmedia
- Tmin
- Tmax

```
$mahout trainlogistic --input /home/cloudera/ia/clima --output /home/cloudera/model --target Clase --categories 2 --predictors Tmedia Tmin Tmax  --types numeric --features 200 --passes 1000 --rate 80 --lambda 0.00001
```

Resultados trainlogistic:

$mahout runlogistic --input /home/cloudera/ia/clima --model /home/cloudera/model --auc –confusion

- AUC: 0,70
- Matriz de confusión: [[30.0, 5.0], [5.0, 5.0]]

Resultados runlogistic:

$mahout runlogistic --input /home/cloudera/ia/clima_run --model /home/cloudera/model --auc –confusion

- AUC: 0,63
- Matriz de confusión: [[141.0, 24.0], [123.0, 54.0]]

Valoración:
En esta segunda ejecución, se han utilizado tres de las variables disponibles, todas ellas relacionadas con la temperatura, siendo el resultado de entrenamiento del 0,70 y aportando un mejor grado de predicción sobre los datos iniciales. Esto ha sido dado al eliminar del modelo variables que distorsionaban el aprendizaje y que aportaban poco a la predicción.
Al ejecutar el algoritmo con el modelo generado, los resultados han mejorado respecto al modelo anterior, pero siguen siendo bajos, todo y que se alejan de las predicciones aleatorias.

Modelo 3
Variables predictivas utilizadas:

- Tmedia
- Tmin
- Tmax
- Hmedia
- Ptotal

$mahout trainlogistic --input /home/cloudera/ia/clima --output /home/cloudera/model --target Clase --categories 2 --predictors Tmedia Tmin Tmax Hmedia Ptotal --types numeric --features 200 --passes 1000 --rate 80 --lambda 0.00001

Resultados trainlogistic:

$mahout runlogistic --input /home/cloudera/ia/clima --model /home/cloudera/model --auc –confusion

- AUC: 0,82
- Matriz de confusión: [[32.0, 3.0], [3.0, 7.0]]

Resultados runlogistic:

$mahout runlogistic --input /home/cloudera/ia/clima_run --model /home/cloudera/model --auc –confusion

- AUC: 0,65
- Matriz de confusión: [[248.0, 57.0], [16.0, 21.0]]

Valoración:
En esta tercera ejecución, se han utilizado cinco de las variables disponibles, relacionadas con la temperatura y la humedad o precipitación, siendo el resultado de entrenamiento del 0,82 y

aportando un mejor grado de predicción sobre los datos iniciales. En este caso, se han utilizado las variables que tienen un alto grado de influencia en el modelo de aprendizaje. Al ejecutar el algoritmo con el modelo generado, los resultados han mejorado respecto al modelo anterior, todo y que son mejorables para obtener una predicción más certera.

Incorporación nuevas variables al modelo

En el caso que se desee incorporar nuevas fuentes de datos para enriquecer el modelo estadístico, se deberá tener en cuenta un conjunto de consideraciones y seguir un conjunto de pasos.

Consideraciones:

- Los datos deben tener el mismo nivel de dimensión temporal que los datos climatológicos actuales: nivel anual
- Los datos deben tener el mismo nivel de dimensión temporal que los datos climatológicos actuales: nivel provincia

Pasos a seguir aprovisionamiento:

- Disposición de los datos en un fichero plano acordes a la temporalidad y geografía que los ya incorporados en el sistema
- Adaptación del proceso de carga para que se realice la normalización de los datos a nivel geográfico.
- Adaptación del proceso de carga para que se incorpore la información al fichero global.

Pasos a seguir modelo estadístico:

- Adaptación del modelo de aprendizaje seleccionando una muestra representativa de lo datos con la nueva variable añadida, con la finalidad de volver a generar un nuevo modelo contemplando esta nueva variable predictiva.
- Reproceso de los datos en base el nuevo modelo y valoración de los resultados

4. Solución Business Intelligence

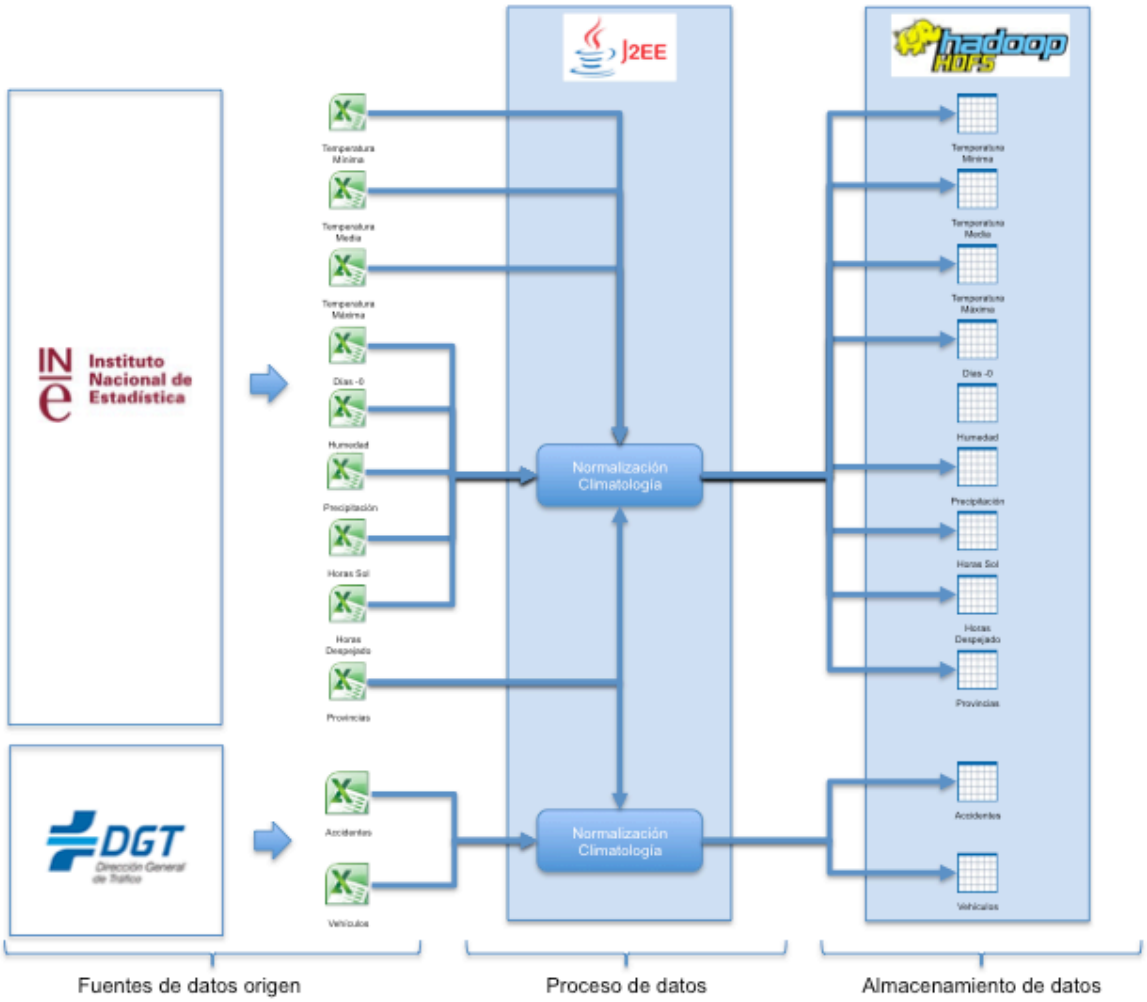

Figura 10. Proceso de incorporación de los datos al repositorio HDFS

El diagrama de la figura 10 describe el flujo de datos en la incorporación de los datos procedentes del INE y de la DGT con la finalidad de ser incorporados en el repositorio HDFS y ser explotados mediante la herramienta de Business Intelligence QlikView.
Para la incorporación de los datos se utiliza un procesos desarrollado mediante el lenguaje de programación JAVA que por un lado permite la normalización de los datos a nivel de provincia, y por otro, permite la incorporación de los datos al HDFS de una forma sencilla y ágil.

El modelo de datos utilizado para analizar la información procedente del repositorio Bigdata HDFS estará constituido por las siguiente modelo de datos:

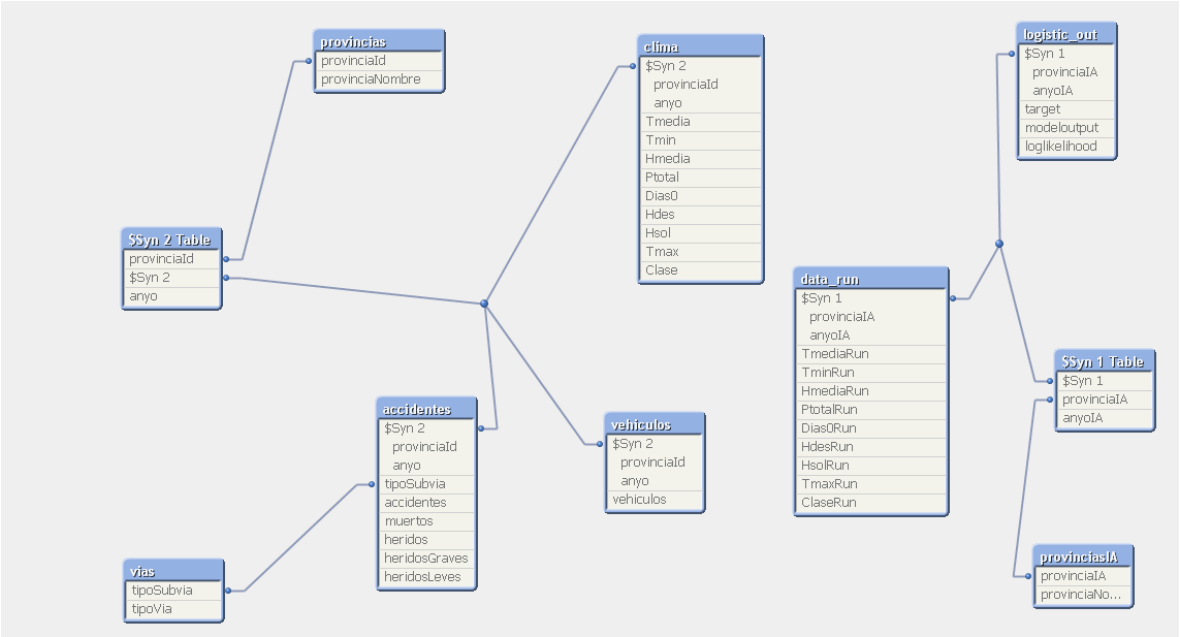

Figura 11. Diagrama modelo de datos lógico QlikView

El modelo está constituido por la siguientes dimensiones:

- Dimensión temporal. Reflejará los distintos periodos de tiempo en los que se darán lugar los hechos climatológicos y de tráfico Dicha dimensión estará constituida por los siguientes atributos:
  - Año
- Dimensión territorial. Reflejará los distintos espacios geográficos en los que se dará lugar los hechos climatológicos y de tráfico. Dicha dimensión estará constituida por los siguientes atributos:
  - Provincia
- Dimensión vía. Reflejará los distintos tipos de vía en los que se dará lugar los hechos de tráfico. Dicha dimensión estará constituida por los siguientes atributos:
  - Tipo vía
  - Tipo subvía

El modelo está constituido por la siguientes hechos:

- Hechos de tráfico. Datos disponibles a nivel anual, por provincia y por tipo de subvía.
  - Hecho Accidentes
  - Hecho Muertos
  - Hecho Heridos
  - Hecho Heridos graves
  - Hecho Heridos leves
  - Hecho Parque de vehículos
- Hechos climatológicos. Datos disponibles a nivel anual y por provincia.
  - Hecho Temperatura media
  - Hecho Temperatura mínima
  - Hecho Temperatura máxima
  - Hecho Humedad media
  - Hecho Precipitación total
  - Hecho Número de días con temperatura inferior a 0 grados centígrados

Paralelamente a los datos de tráfico y climatología, se dispondrá de los resultados de los dos algoritmos de clasificación para determinar las posibles predicciones.

Los datos de los que se dispondrá para realizar el análisis del resultado de los dos algoritmos de clasificación son:

- Registros utilizados en el muestreo base y utilizados para construir el modelo inicial. Dichos datos vendrán separados en dos subset de datos.
- Un primer subset con los datos de accidentes donde lleva asignado manualmente una clasificación de provincias, mediante el atributo clase, según su volumetría de accidentes de forma proporcional al parque de vehículos. Dicha clasificación determina las provincias con mayor riesgo, pero sin saber las causas de este.
- Un segundo subset con los datos del climas por provincia donde se le asignará el correspondiente atributo clase determinado en el primer subset. Este segundo subset de datos es la base para generar el modelo estadístico.
    - Provincia
    - Año
    - Temperatura media
    - Temperatura mínima
    - Humedad media
    - Precipitación total
    - Número de días con temperatura inferior a 0 grados centígrados
    - Días despejados
    - Horas de sol
    - Clase utilizada para la clasificación
- Registros resultantes de la clasificación en base al anterior modelo estadístico:
    - Provincia
    - Año
    - Temperatura media
    - Temperatura mínima
    - Humedad media
    - Precipitación total
    - Número de días con temperatura inferior a 0 grados centígrados
    - Días despejados
    - Horas de sol
    - Clase asignada por al algoritmo

A parte de las tablas ya nombradas, existen dos tablas sintéticas autogeneradas “Syn 1 Table” y “Syn 2 Table”, que la propia herramienta las genera para disponer de una tabla de relación para las claves compuestas. Este comportamiento es propio de la herramienta al no poder disponer de claves compuestas que relacionen directamente a las tablas de hechos con las de dimensiones.

4.1 Análisis de datos de tráfico

Mediante la herramienta QlikView dispondremos de la información de accidentes de tráfico, lo cuales podremos analizar, tanto los propios accidentes a nivel de año y provincias como de la cantidad de muertos y heridos resultantes de los accidentes.

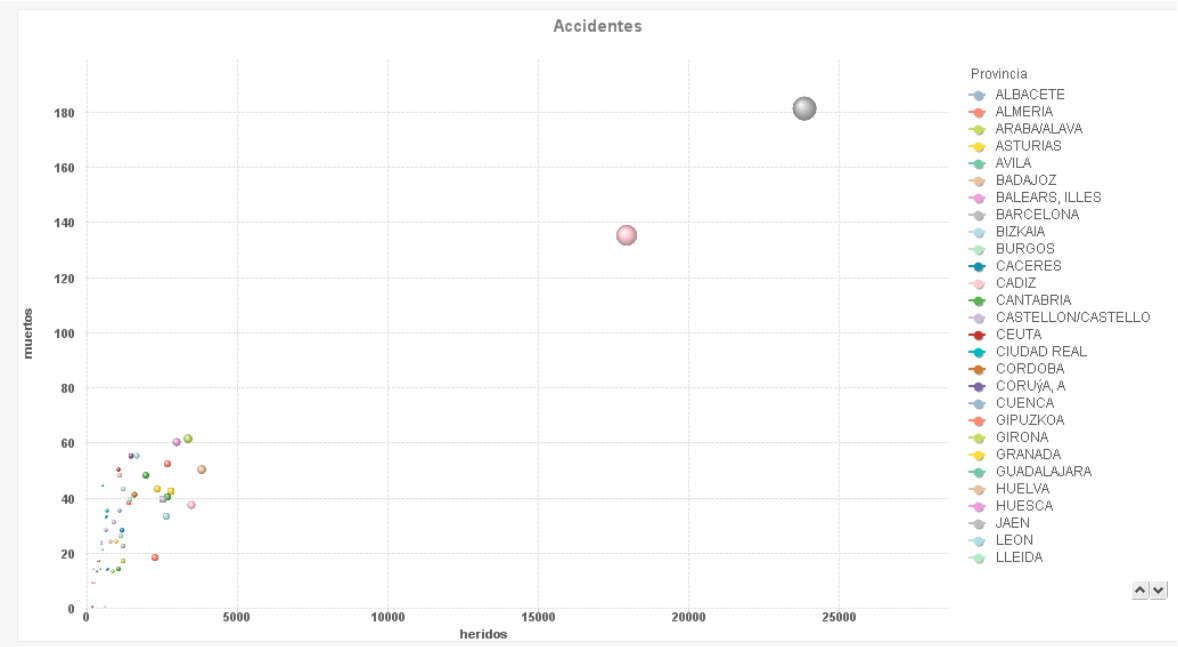

Figura 12. Relación accidentes, muertos y heridos por provincia

Como se puede observar en la figura 12, las provincias de Barcelona y Madrid, dada su volumetría de vehículos, no permiten poder realizar un análisis en detalle del resto de las provincias. Todo y con ello, es importante hacer foco a las dos provincias, ya que dado su alto nivel de tráfico, la volumetría de accidentes, heridos y muertos.

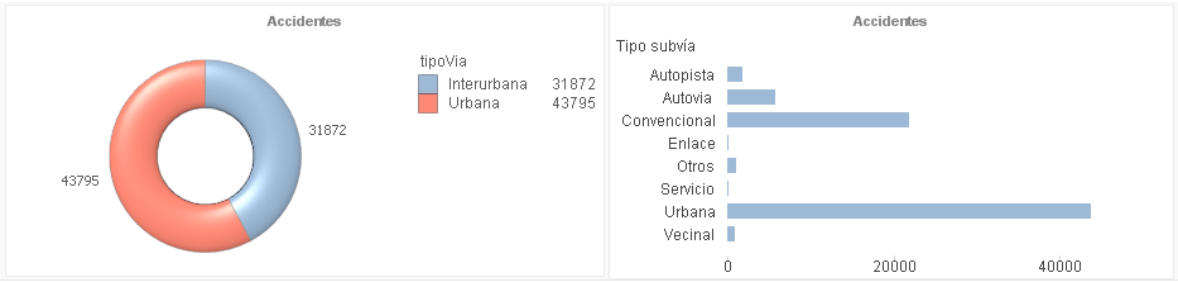

Figura 13. Accidentes por tipo de vía

Figura 14. Información accidentes Barcelona

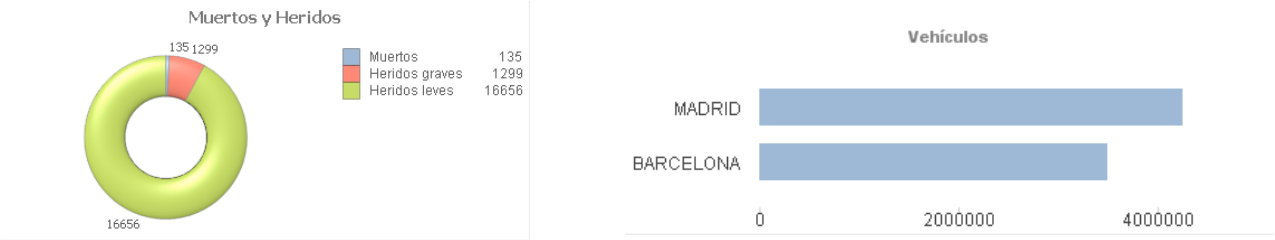

Figura 15. Información accidentes Madrid

Una vez excluidas dichas provincias, en la figura 13 ya se puede observar la distribución de las provincias por gravedad de sus accidentes por la volumetría de muertos resultantes, y determinando las zonas con menor volumen de accidentes pero con mayor gravedad.

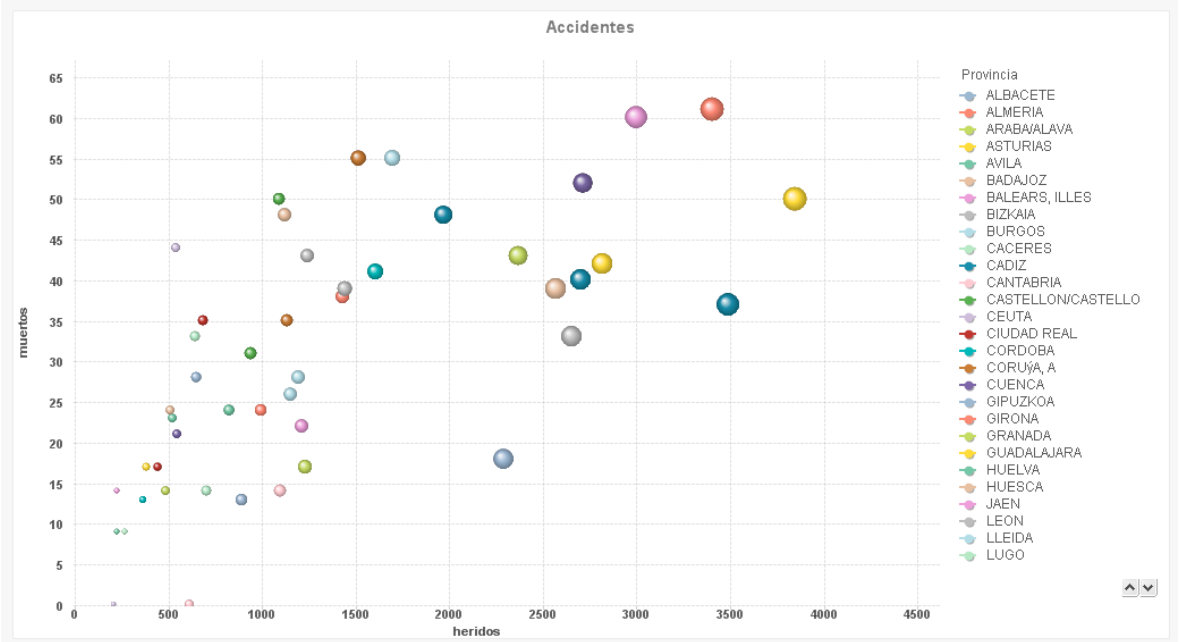

Figura 16. Relación accidentes, muertos y heridos por provincia descartando Barcelona y Madrid

Figure 17. Accidentes por tipo de vía excluyendo Barcelona y Madrid

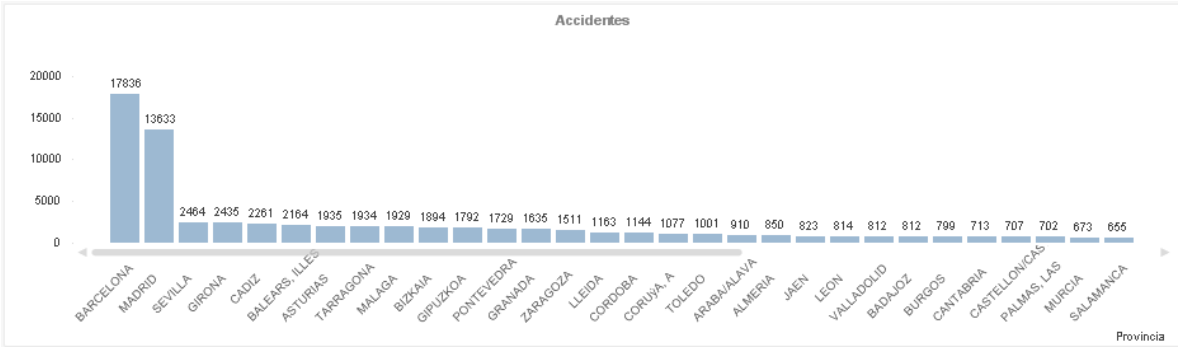

Figura 18. Ranking de accidentes por provincias

4.2 Análisis de datos climatológicos

A continuación se muestran los rankings de los diferentes indicadores climatológicos que permitirán determinar aquellas provincias que pertenecen a los primeros puestos de algunos de los indicadores.

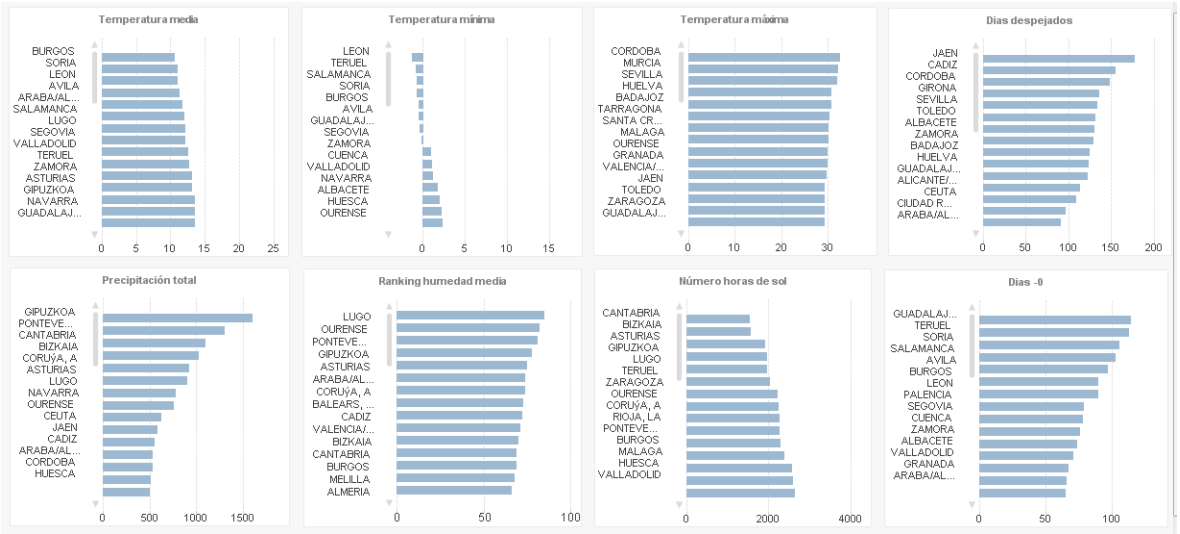

Figura 19. Análisis de información climatológica excluyendo a Barcelona y Madrid

4.3 Análisis de datos cruzados

En los siguientes gráficos se realiza un análisis cruzado de los distintos indicadores mediante los gráficos de burbujas donde el número de accidentes siempre será el volumen de las burbujas, mientras que los otros indicadores estarán en los ejes.

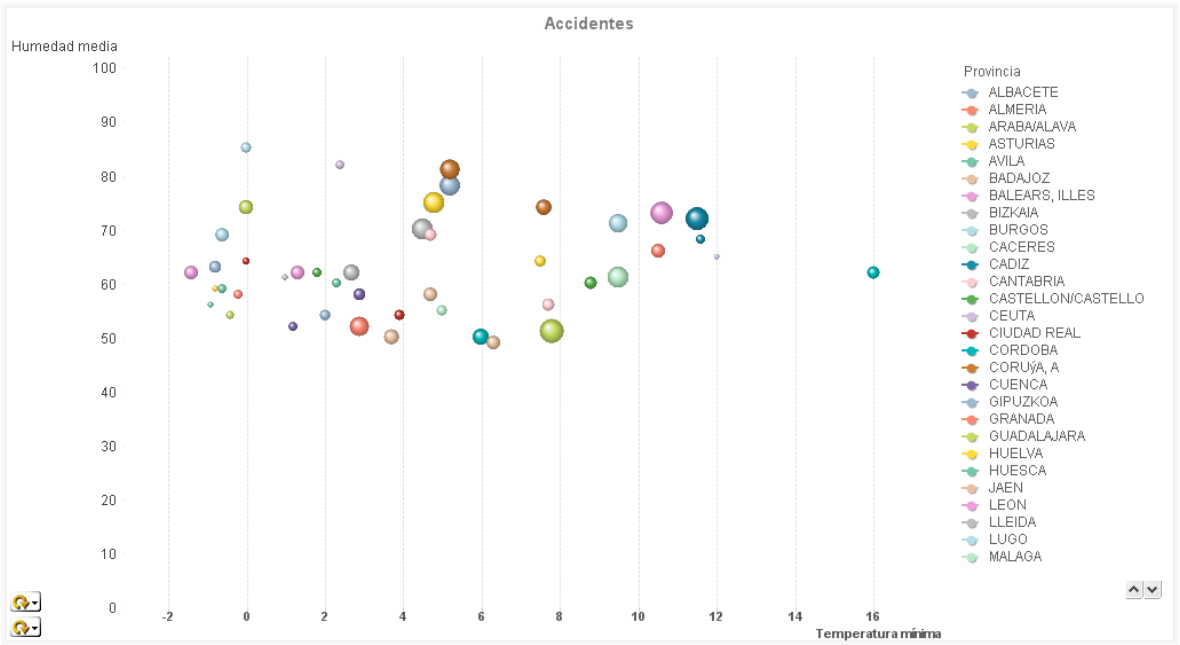

Figura 20. Humedad media, temperatura mínima y accidentes

En la figura 20 vemos que el indicador de humedad media no es relevante ya que su distribución está muy concentrada por la mayor parte de las provincias y no tiene una relación clara con los accidentes, no siendo así con la temperatura mínima donde se aprecia que hay una distribución en sus distintos valores y la gran parte de los accidentes están entre los 2 y los 12 grados.

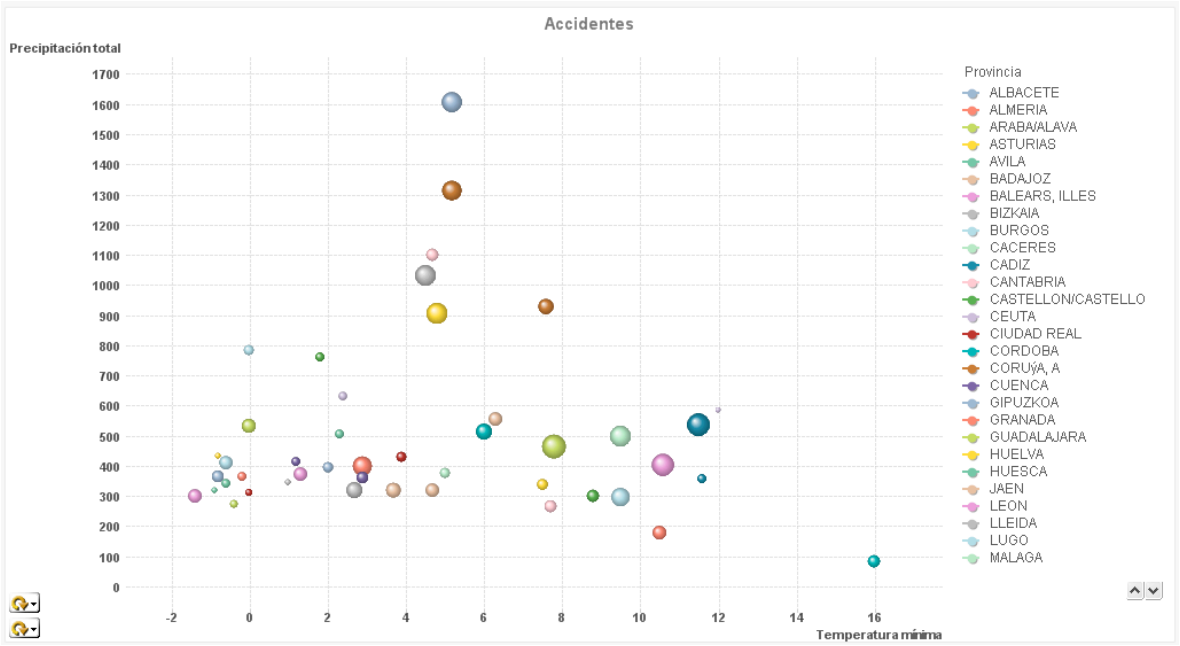

Figura 21. Precipitación total, temperatura mínima y accidentes

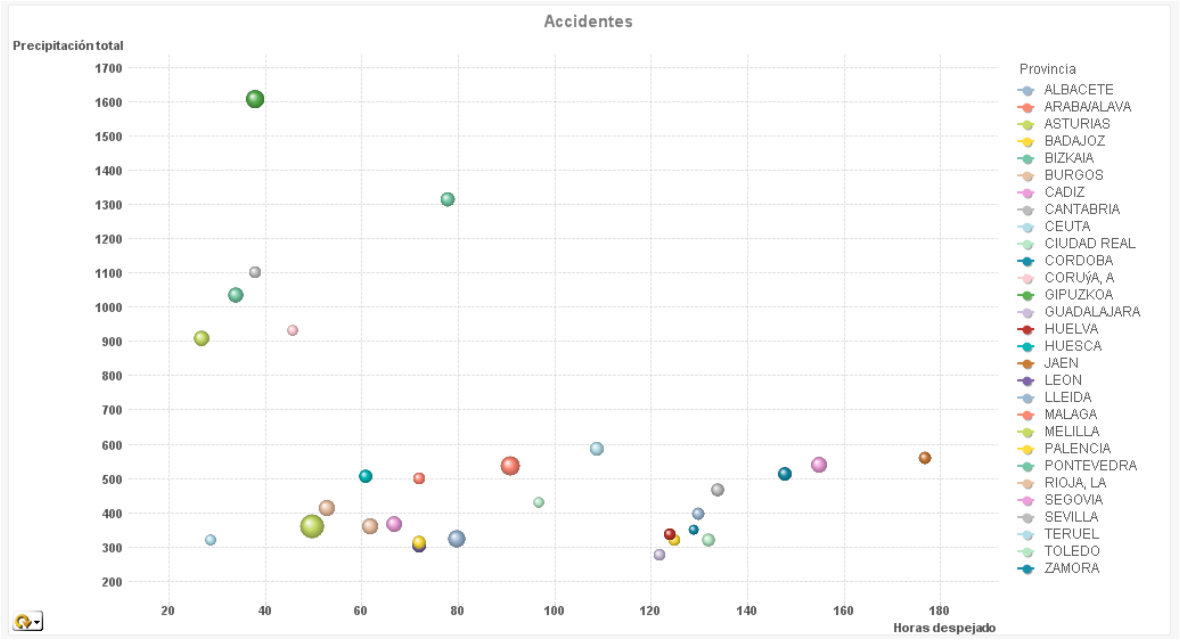

Figura 22. Precipitación total, horas con el cielo despejado y accidentes

4.4 Análisis de datos predictivos

A parte de los resultados obtenidos por las ejecuciones del algoritmo predictivo, se dispone en la herramienta de Business Intelligence los datos utilizados en la predicción así como el resultado una vez procesados.
Para cada registro se podrá consultar la categorización de los registros así como su tasa de error.

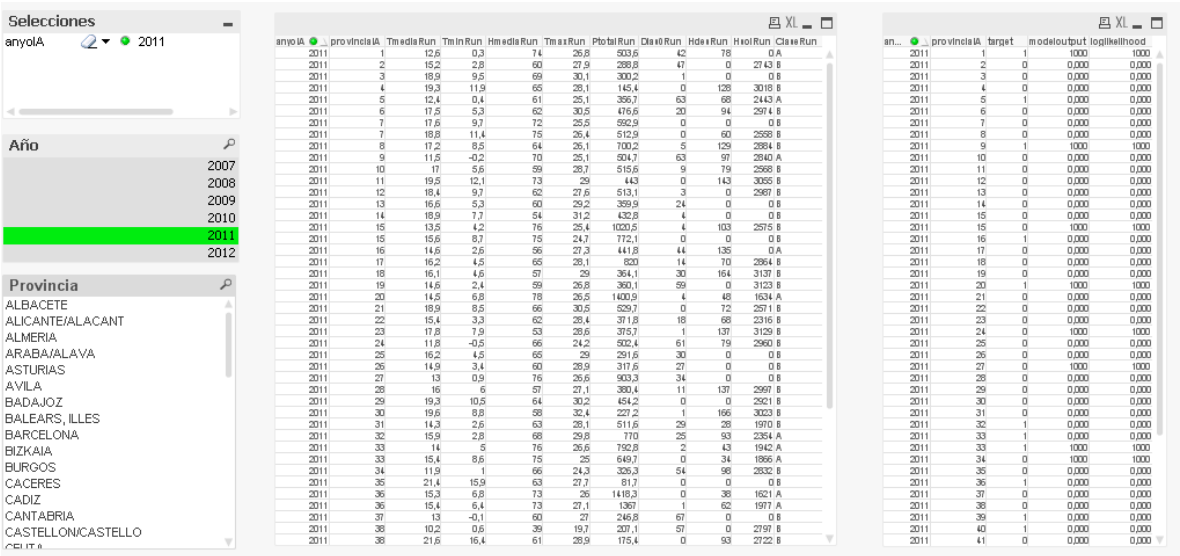

Figura 23. Resultados categorización

5. Análisis de resultados y conclusiones

El actual modelo predictivo permite realizar un ejercicio de relación causa efecto mediante dos tecnologías punteras a día de hoy, y aprovechar sinergias de ambas:

- por un lado tenemos los algoritmos predictivos que en base a información histórica, se crea un modelo estadístico que permite procesar nueva información y predecir en base a la clasificación qué posible afectación tendrá. En el caso actual, la afectación será el nivel de accidentes que podría tener una provincia en base a la climatología.
- por otro lado tenemos una herramienta de análisis que nos permite explorar los datos de accidentes como climatológicos, tanto por separado como de forma conjunta, con la finalidad de poder detectar patrones de comportamiento mediante tablas de datos y gráficas, así como analizar los resultados de los algoritmos predictivos.

Como resultado del piloto y estudio realizado se ha llegado a las siguientes conclusiones:

- En el caso actual, se han analizado los datos, y una vez descartados aquellos que podrían distorsionar lo resultados (zonas con alto nivel de tráfico ya sea por volumen de vehículos locales, o por se una zona de alta afluencia turística, o zona de paso), se ha llegado a un modelo predictivo que permite realizar predicciones con un nivel de fiabilidad mayor a la aleatoriedad.
- Para llegar a un modelo predictivo con un nivel de fiabilidad alto, en nuestro caso, un modelo de relación causa-efecto, no todas la variables predictivas pueden ayudar a tal fin, sino que algunas de ellas pueden llevar a resultados erróneos y distorsionarlos de forma sutil. Para ello, se debe trabajar en su elección, conjuntamente con la combinatoria de las variables así como con la herramienta de Business Intelligence.
- El actual modelo predictivo, al disponer de los datos de accidentes y climatológicos a nivel de anual y provincia, sólo permite realizar predicciones muy genéricas a nivel de provincia en base a las medias anuales de sus datos climatológicos. En caso de disponer de una información más detallada, como por ejemplo los accidentes y el clima a nivel diario, las predicciones podrían ser mucho más ajustadas.
- Este modelo predictivo abre las puertas a la incorporación de nuevas variables predictivas que permitan analizar y predecir la probabilidad de accidentes en base a ratios económicos, antigüedad del parque de vehículos, niveles de alcoholemia, así como el estado de las vías, etc.

6. Bibliografía

cloudera. QuickStart VMs for CDH 5.2.x. Máquina virtual con el software preconfigurado, obtenido de http://www.cloudera.com/content/cloudera/en/downloads/quickstart_vms/cdh-5-2-x.html

cloudera. Documentación sobre configuración y uso de la distribución del framework de Apache Hadoop, obtenido de http://www.cloudera.com/content/cloudera/en/documentation.html#ClouderaDocumentation

Apache Hadoop. Documentación sobre los distintos componentes que constituyen el framework, obtenido de http://hadoop.apache.org/docs/current/

Apache Mahout. Documentación sobre el proyecto de librerías de máquinas de aprendizaje, obtenido de http://mahout.apache.org

Wikipedia. Stochastic gradient descent, obtenido de http://en.wikipedia.org/wiki/Stochastic_gradient_descent

Logistic Regresion (SGD) del proyecto Apache Mahout. Descripción y ejemplos, obtenido de http://mahout.apache.org/users/classification/logistic-regression.html

Paul Komarek. Logistic Regression for Data Mining and High-Dimensional Classification. Obtenido de http://www.autonlab.org/autonweb/14709/version/4/part/5/data/komarek:lr_thesis.pdf?branch=main&language=en

Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman (2012). Mahout in Action: Manning.

QlikView. Tutorial de desarrollo, obtenido de http://www.qlik.com/~/media/Files/training/tutorials-qv11/Spanish.ashx

7. Anexos

7.1 Código Solución BigData

7.1.1 Instrucciones ejecución algoritmos

Productos necesarios:

- Descargar la herramienta Oracle VirtualBox Manager la cual nos permitirá poder ejecutar la maquina virtual de cloudera donde estará instalado el software Big Data: https://www.virtualbox.org/wiki/Downloads
- Descargar la máquina virtual de cloudera: http://www.cloudera.com/content/support/en/downloads/quickstart_vms.html

Pasos a seguir:

- Acceder a la máquina virtual con el usuario "cloudera" el cual dispone de acceso a todos los ejecutables y tiene preconfigurado las variables PATH y CLASSPATH necesarios para la ejecución de los programas.

```
$/home/cloudera
```

- Dejar en el directorio */home/cloudera* el fichero "data_train" con los datos representativos para que el algoritmo genere un modelo de aprendizaje. Lo dejaremos en un directorio local para facilitar la ejecución.
- Ejecutar el algoritmo trainlogistic mediante el programa mahout desde el directorio raíz del usuario cloudera, indicando el fichero origen de datos */home/cloudera/data_train*, así como la ubicación del modelo a generar */home/cloudera/model*, y los parámetros correspondientes (variables, target y parámetros de ejecución)

```
mahout trainlogistic --input /home/cloudera/data_train --output /home/cloudera/model --target Clase --categories 2 --predictors Tmedia Tmin Tmax --types numeric --features 104 --passes 5000 --rate 80
```

- Validar y evaluar el modelo construido con los datos iniciales. Para ello tendremos que indicar la ubicación del modelo */home/cloudera/model* y de los datos utilizados para su creación */home/cloudera/data_train*. En caso que el resultado no se adapte a lo esperado se deberá replantear el punto anterior y volverlo a ejecutar.

```
mahout runlogistic --input /home/cloudera/data_train --model /home/cloudera/model --auc --confusion
```

- Dejar en el directorio */home/cloudera* el fichero "data_run" con el resto de los datos para que el algoritmo los procese en base al modelo de aprendizaje generado en el anterior paso.
- Validar y evaluar el modelo construido con nuevos datos ejecutando el algoritmo runlogistic mediante el programa mahout. Para ello tendremos que indicar la ubicación del modelo */home/cloudera/model* y de los nuevos datos a procesar */home/cloudera/data_run*. En caso que el resultado no se adapte a lo esperado se deberá replantear el punto inicial de la creación del modelo.

```
mahout runlogistic --input /home/cloudera/data_run --model /home/cloudera/model --auc -confusion
```

- En caso que el resultado sea satisfactorio generaremos un fichero de salida con la información de los datos calcificados donde se nos informará registro a registro la clasificación así como el ratio de error de esta.

```
mahout runlogistic --input /home/cloudera/data_run --model /home/cloudera/model --scores
```

7.1.2 LoadFilesIA.java

Para poder ejecutar el siguiente código se deben configurar los paths de ubicación de los ficheros a utilizar tanto en disco como en HDFS (marcados en amarillo).

Disco local:

- Rutas de los ficheros con datos origen

HDFS:

- Directorio dentro del HDFS donde se ubican los ficheros

```
package uoc.edu;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class LoadFilesIA {

		private HashMap<String, String> provincias;
		private HashMap<String, ProvinciaDatosClima> provinciasData;


		//Paths en HDFS
		private String iaPath = "/user/cloudera/accidentes/ia/";

		/**
		 *
		 */
		public LoadFilesIA() {
			provincias = new HashMap<String,String>();
			provinciasData = new HashMap<String, ProvinciaDatosClima>();
		}
```

```
public void cargarProvincias(String fichero){

    File file = null;
    FileReader fr = null;
    BufferedReader br = null;

    try{
        //Apertura del fichero y creación del buffer para poder leerlo
        file = new File(fichero);
        fr = new FileReader(file);
        br = new BufferedReader(fr);

        //Lectura del fichero
        String linea;
        StringTokenizer str = null;
        while((linea = br.readLine())!=null){
            //Leemos los campos separados por ;
            str = new StringTokenizer(linea,";");

            //id
            String id = null;
            //Provincia
            String provincia = null;

            if(str.hasMoreTokens())     id = str.nextToken();
            if(str.hasMoreTokens())     provincia = str.nextToken();

            provincias.put(id, provincia);
        }
    }
    catch(IOException e){
        System.out.println(e.getMessage());
    }
    finally{
        try{
            if(fr!=null)            fr.close();
        }
        catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
}

public void cargarTemperaturaMedia(String fichero){

    File accidentesFile = null;
    FileReader accidentesFR = null;
    BufferedReader accidentesBR = null;

    try{
        //Apertura del fichero y creación del buffer para poder leerlo
        accidentesFile = new File(fichero);
        accidentesFR = new FileReader(accidentesFile);
        accidentesBR = new BufferedReader(accidentesFR);

        //Lectura del fichero
        String linea;
        ProvinciaDatosClima data = null;
        StringTokenizer str = null;

        //Provincia
        String provincia = null;
```

```
                String anyo = null;
                while((linea = accidentesBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String provinciaId = this.buscarProvincia(provincia);
                        data = new ProvinciaDatosClima(provinciaId);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        data.setAnyo(Integer.parseInt(anyo));

                        if(str.hasMoreTokens())
    data.setTemperaturaMedia((Double.parseDouble(str.nextToken().replace(",",".")));
                        provinciasData.put(provinciaId+anyo, data);
                }
    }
    catch(IOException e){
                System.out.println(e.getMessage());
    }
    finally{
                try{
                        if(accidentesFR!=null)       accidentesFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
    }
}

public void cargarTemperaturaMaxima(String fichero){

    File climaFile = null;
    FileReader climaFR = null;
    BufferedReader climaBR = null;

    try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
```

```
                    key += anyo;
                    data = this.provinciasData.get(key);

                    if(str.hasMoreTokens())
        data.setTemperaturaMaxima((Double.parseDouble(str.nextToken().replace(",","."))));

                }
            }
            catch(IOException e){
                System.out.println(e.getMessage());
            }
            finally{
                try{
                    if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                    System.out.println(e.getMessage());
                }
            }
        }

        public void cargarTemperaturaMinima(String fichero){

            File climaFile = null;
            FileReader climaFR = null;
            BufferedReader climaBR = null;

            try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                    //Leemos los campos separados por tabulador
                    str = new StringTokenizer(linea,";");

                    provincia = null;
                    anyo = null;

                    if(str.hasMoreTokens())     provincia = str.nextToken();
                    String key = this.buscarProvincia(provincia);
                    if(str.hasMoreTokens())     anyo = str.nextToken();
                    key += anyo;
                    data = this.provinciasData.get(key);

                    if(str.hasMoreTokens())
        data.setTemperaturaMinima((Double.parseDouble(str.nextToken().replace(",","."))));

                }
            }
            catch(IOException e){
                System.out.println(e.getMessage());
            }
```

```
        finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}

public void cargarPrecipitiacion(String fichero){

        File climaFile = null;
        FileReader climaFR = null;
        BufferedReader climaBR = null;

        try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())
        data.setPrecipitacionTotal((Double.parseDouble(str.nextToken().replace(",","."))));

                }
        }
        catch(IOException e){
                System.out.println(e.getMessage());
        }
        finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}

public void cargarHumedadMedia(String fichero){
```

```
        File climaFile = null;
        FileReader climaFR = null;
        BufferedReader climaBR = null;

        try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())
        data.setHumedadMedia((Double.parseDouble(str.nextToken().replace(",","."))));
                }
        }
        catch(IOException e){
                System.out.println(e.getMessage());
        }
        finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}

public void cargarHorasSol(String fichero){

        File climaFile = null;
        FileReader climaFR = null;
        BufferedReader climaBR = null;

        try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
```

```
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())
    data.setHorasSol((Double.parseDouble(str.nextToken().replace(",","."))));
                }
    }
    catch(IOException e){
                System.out.println(e.getMessage());
    }
    finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
    }
}

public void cargarDiasMenos0(String fichero){

    File climaFile = null;
    FileReader climaFR = null;
    BufferedReader climaBR = null;

    try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
```

```
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())
    data.setDiasMenos0((Double.parseDouble(str.nextToken().replace(",",".")));
                }
    }
    catch(IOException e){
                System.out.println(e.getMessage());
    }
    finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
    }
}

public void cargarDiasDespejado(String fichero){

    File climaFile = null;
    FileReader climaFR = null;
    BufferedReader climaBR = null;

    try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())
    data.setDiasDespejado((Double.parseDouble(str.nextToken().replace(",",".")));
                }
    }
```

```
        catch(IOException e){
                System.out.println(e.getMessage());
        }
        finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}

public void cargarClase(String fichero){

        File climaFile = null;
        FileReader climaFR = null;
        BufferedReader climaBR = null;

        try{
                //Apertura del fichero y creación del buffer para poder leerlo
                climaFile = new File(fichero);
                climaFR = new FileReader(climaFile);
                climaBR = new BufferedReader(climaFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosClima data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = climaBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        key += anyo;
                        data = this.provinciasData.get(key);

                        if(str.hasMoreTokens())     data.setClase(str.nextToken());
                }
        }
        catch(IOException e){
                System.out.println(e.getMessage());
        }
        finally{
                try{
                        if(climaFR!=null)  climaFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}
```

```
public String buscarProvincia(String nombre){
    Iterator<String> provinciasIt = provincias.keySet().iterator();
    String provinciaId = null;
    boolean encontrado = false;

    while(!encontrado && provinciasIt.hasNext()){
            provinciaId = (String)provinciasIt.next();
            if(nombre.equals((String)this.provincias.get(provinciaId)))     encontrado = true;
    }
    return provinciaId;
}

public void grabarClima(){
    Configuration conf = new Configuration();
    conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
   conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

    try {
            FileSystem hdfs = FileSystem.get(conf);
            Path path = new Path(iaPath+"clima");
            FSDataOutputStream out = hdfs.create(path);

            StringBuffer sb = new StringBuffer();
            ProvinciaDatosClima datos = null;

            Iterator<ProvinciaDatosClima> iterator = provinciasData.values().iterator();

            if(iterator.hasNext()){
                    sb.append("\"Provincia\",");
                    sb.append("\"Periodo\",");
                    sb.append("\"Tmedia\",");
                    sb.append("\"Tmin\",");
                    sb.append("\"Hmedia\",");
                    sb.append("\"Ptotal\",");
                    sb.append("\"Dias-0\",");
                    sb.append("\"Ddes\",");
                    sb.append("\"Hsol\",");
                    sb.append("\"Tmax\",");
                    sb.append("\"Clase\"");

                    sb.append("\n");
            }

            while(iterator.hasNext()){
                    datos = (ProvinciaDatosClima)iterator.next();

                    sb.append(datos.getId());
                    sb.append(",");
                    sb.append(datos.getAnyo());
                    sb.append(",");
                    sb.append(datos.getTemperaturaMedia());
                    sb.append(",");
                    sb.append(datos.getTemperaturaMinima());
                    sb.append(",");
                    sb.append(datos.getHumedadMedia());
                    sb.append(",");
                    sb.append(datos.getPrecipitacionTotal());
                    sb.append(",");
                    sb.append(datos.getDiasMenos0());
                    sb.append(",");
                    sb.append(datos.getDiasDespejado());
                    sb.append(",");
```

```
                                sb.append(datos.getHorasSol());
                                sb.append(",");
                                sb.append(datos.getTemperaturaMaxima());
                                sb.append(",");
                                sb.append(datos.getClase());

                                sb.append("\n");
                        }

                        out.writeBytes(sb.toString());

                } catch (IOException e) {
                        System.out.println(e.getMessage());
                }
        }

        public void grabarClima12(){
             Configuration conf = new Configuration();
             conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
            conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

             try {
                        FileSystem hdfs = FileSystem.get(conf);
                        Path path = new Path(iaPath+"clima12");
                        FSDataOutputStream out = hdfs.create(path);

                        StringBuffer sb = new StringBuffer();
                        ProvinciaDatosClima datos = null;

                        Iterator<ProvinciaDatosClima> iterator = provinciasData.values().iterator();

                        if(iterator.hasNext()){
                                sb.append("\"Provincia\",");
                                sb.append("\"Periodo\",");
                                sb.append("\"Tmedia\",");
                                sb.append("\"Tmin\",");
                                sb.append("\"Hmedia\",");
                                sb.append("\"Ptotal\",");
                                sb.append("\"Dias-0\",");
                                sb.append("\"Ddes\",");
                                sb.append("\"Hsol\",");
                                sb.append("\"Tmax\",");
                                sb.append("\"Clase\"");

                                sb.append("\n");
                        }

                        while(iterator.hasNext()){
                                datos = (ProvinciaDatosClima)iterator.next();

                                sb.append(datos.getId());
                                sb.append(",");
                                sb.append(datos.getAnyo());
                                sb.append(",");
                                sb.append(datos.getTemperaturaMedia());
                                sb.append(",");
                                sb.append(datos.getTemperaturaMinima());
                                sb.append(",");
                                sb.append(datos.getHumedadMedia());
                                sb.append(",");
                                sb.append(datos.getPrecipitacionTotal()/12);
                                sb.append(",");
```

```
                                        sb.append(datos.getDiasMenos0()/12);
                                        sb.append(",");
                                        sb.append(datos.getDiasDespejado()/12);
                                        sb.append(",");
                                        sb.append(datos.getHorasSol()/12);
                                        sb.append(",");
                                        sb.append(datos.getTemperaturaMaxima());
                                        sb.append(",");
                                        sb.append(datos.getClase());

                                        sb.append("\n");
                                }

                                out.writeBytes(sb.toString());

                        } catch (IOException e) {
                                System.out.println(e.getMessage());
                        }
                }

                /**
                 * @param args
                 */
                public static void main(String[] args) {

                        String ficheroProvincias = "/home/cloudera/shared/inbox/clima/provincias.csv";
                        String ficheroDiasDespejado = "/home/cloudera/shared/inbox/clima/diasDespejado.csv";
                        String ficheroDiasMenos0 = "/home/cloudera/shared/inbox/clima/diasDespejado.csv";
                        String ficheroHorasSol = "/home/cloudera/shared/inbox/clima/horasSol.csv";
                        String ficheroHumedadMedia = "/home/cloudera/shared/inbox/clima/humedadMedia.csv";
                        String ficheroPrecipitacion = "/home/cloudera/shared/inbox/clima/precipitacion.csv";
                        String ficherotemperaturaMaxima =
                "/home/cloudera/shared/inbox/clima/temperaturaMaxima.csv";
                        String ficherotemperaturaMinima =
                "/home/cloudera/shared/inbox/clima/temperaturaMinima.csv";
                        String ficherotemperaturaMedia =
                "/home/cloudera/shared/inbox/clima/temperaturaMedia.csv";
                        String ficheroClase = "/home/cloudera/shared/inbox/clima/clase.csv";

                        LoadFilesIA carga = new LoadFilesIA();
                        carga.cargarProvincias(ficheroProvincias);
                        carga.cargarTemperaturaMedia(ficherotemperaturaMedia);
                        carga.cargarTemperaturaMaxima(ficherotemperaturaMaxima);
                        carga.cargarTemperaturaMinima(ficherotemperaturaMinima);
                        carga.cargarDiasDespejado(ficheroDiasDespejado);
                        carga.cargarDiasMenos0(ficheroDiasMenos0);
                        carga.cargarHorasSol(ficheroHorasSol);
                        carga.cargarHumedadMedia(ficheroHumedadMedia);
                        carga.cargarPrecipitiacion(ficheroPrecipitacion);
                        carga.cargarClase(ficheroClase);

                        carga.grabarClima();
                        carga.grabarClima12();
                }

}
```

7.1.3 LoadFilesDWH.java

Para poder ejecutar el siguiente código se deben configurar los paths de ubicación de los ficheros a utilizar tanto en disco como en HDFS (marcados en amarillo).

Disco local:
- Rutas de los ficheros con datos origen

HDFS:
- Directorio dentro del HDFS donde se ubican los ficheros

```
package uoc.edu;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class LoadFilesDWH {

		private HashMap<String, String> provincias;
		private HashMap<String, ProvinciaDatosTrafico> provinciasData;


		//Paths en HDFS
		private String dwhPath = "/user/cloudera/accidentes/dwh/";

		/**
		 *
		 */
		public LoadFilesDWH() {
			provincias = new HashMap<String,String>();
			provinciasData = new HashMap<String, ProvinciaDatosTrafico>();
		}

		public void cargarProvincias(String fichero){

			File file = null;
			FileReader fr = null;
			BufferedReader br = null;

			try{
				//Apertura del fichero y creación del buffer para poder leerlo
				file = new File(fichero);
				fr = new FileReader(file);
				br = new BufferedReader(fr);

				//Lectura del fichero
				String linea;
				StringTokenizer str = null;
				while((linea = br.readLine())!=null){
```

```
                        //Leemos los campos separados por ;
                        str = new StringTokenizer(linea,";");

                        //id
                        String id = null;
                        //Provincia
                        String provincia = null;

                        if(str.hasMoreTokens())     id = str.nextToken();
                        if(str.hasMoreTokens())     provincia = str.nextToken();

                        provincias.put(id, provincia);
                }
        }
        catch(IOException e){
                System.out.println(e.getMessage());
        }
        finally{
                try{
                        if(fr!=null)            fr.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
        }
}

public void cargarAccidentes(String fichero){

        File accidentesFile = null;
        FileReader accidentesFR = null;
        BufferedReader accidentesBR = null;

        try{
                //Apertura del fichero y creación del buffer para poder leerlo
                accidentesFile = new File(fichero);
                accidentesFR = new FileReader(accidentesFile);
                accidentesBR = new BufferedReader(accidentesFR);

                //Lectura del fichero
                String linea;
                ProvinciaDatosTrafico data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = accidentesBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;
                        if(str.hasMoreTokens())     anyo = str.nextToken();



                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String provinciaId = this.buscarProvincia(provincia);
                        data = new ProvinciaDatosTrafico(provinciaId);
                        data.setAnyo(Integer.parseInt(anyo));
```

```
                        /* Autopistas */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Autopista(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Autopista(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Autopista(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Autopista(Integer.parseInt(str.nextToken().replace(".","")));
                                //Heridos leves

    data.setHeridosLeves_VIU_Autopista(data.getHeridos_VIU_Autopista()-
data.getHeridosGraves_VIU_Autopista());
                        }

                        /* Autovías */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Autovia(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Autovia(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Autovia(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Autovia(Integer.parseInt(str.nextToken().replace(".","")));
                                //Heridos leves

    data.setHeridosLeves_VIU_Autovia(data.getHeridos_VIU_Autovia()-
data.getHeridosGraves_VIU_Autovia());
                        }

                        /* Convencional */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Convencional(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Convencional(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Convencional(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Convencional(Integer.parseInt(str.nextToken().replace(".","")));

                                //Heridos leves

    data.setHeridosLeves_VIU_Convencional(data.getHeridos_VIU_Convencional()-
data.getHeridosGraves_VIU_Convencional());
                        }
```

```
                        /* Vecinal */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Vecinal(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Vecinal(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Vecinal(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Vecinal(Integer.parseInt(str.nextToken().replace(".","")));
                            //Heridos leves

    data.setHeridosLeves_VIU_Vecinal(data.getHeridos_VIU_Vecinal()-
data.getHeridosGraves_VIU_Vecinal());
                        }

                        /* Servicio */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Servicio(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Servicio(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Servicio(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Servicio(Integer.parseInt(str.nextToken().replace(".","")));
                            //Heridos leves

    data.setHeridosLeves_VIU_Servicio(data.getHeridos_VIU_Servicio()-
data.getHeridosGraves_VIU_Servicio());
                        }

                        /* Enlace */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Enlace(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Enlace(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Enlace(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Enlace(Integer.parseInt(str.nextToken().replace(".","")));
                            //Heridos leves

    data.setHeridosLeves_VIU_Enlace(data.getHeridos_VIU_Enlace()-
data.getHeridosGraves_VIU_Enlace());
                        }

                        /* Otros */
                        //Accidentes
```

```
                        if(str.hasMoreTokens())
    data.setAccidentes_VIU_Otros(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VIU_Otros(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VIU_Otros(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VIU_Otros(Integer.parseInt(str.nextToken().replace(".","")));
                            //Heridos leves
                            data.setHeridosLeves_VIU_Otros(data.getHeridos_VIU_Otros()-
data.getHeridosGraves_VIU_Otros());
                        }

                        /* Vías Urbanas */
                        //Accidentes
                        if(str.hasMoreTokens())
    data.setAccidentes_VU(Integer.parseInt(str.nextToken().replace(".","")));
                        //Muertos
                        if(str.hasMoreTokens())
    data.setMuertos_VU(Integer.parseInt(str.nextToken().replace(".","")));
                        //Total heridos
                        if(str.hasMoreTokens())
    data.setHeridos_VU(Integer.parseInt(str.nextToken().replace(".","")));
                        //Heridos graves
                        if(str.hasMoreTokens()){

    data.setHeridosGraves_VU(Integer.parseInt(str.nextToken().replace(".","")));
                            //Heridos leves
                            data.setHeridosLeves_VU(data.getHeridos_VU()-
data.getHeridosGraves_VU());
                        }
                        provinciasData.put(provinciaId+anyo, data);
                }
    }
    catch(IOException e){
            System.out.println(e.getMessage());
    }
    finally{
            try{
                    if(accidentesFR!=null)      accidentesFR.close();
            }
            catch(IOException e){
                    System.out.println(e.getMessage());
            }
    }
}

public void cargarParque(String fichero){

    File parqueFile = null;
    FileReader parqueFR = null;
    BufferedReader parqueBR = null;

    try{
            //Apertura del fichero y creación del buffer para poder leerlo
            parqueFile = new File(fichero);
            parqueFR = new FileReader(parqueFile);
            parqueBR = new BufferedReader(parqueFR);
```

```
                //Lectura del fichero
                String linea;
                ProvinciaDatosTrafico data = null;
                StringTokenizer str = null;

                //Provincia
                String provincia = null;
                String anyo = null;
                while((linea = parqueBR.readLine())!=null){
                        //Leemos los campos separados por tabulador
                        str = new StringTokenizer(linea,";");

                        provincia = null;
                        anyo = null;

                        if(str.hasMoreTokens())     anyo = str.nextToken();
                        if(str.hasMoreTokens())     provincia = str.nextToken();
                        String key = this.buscarProvincia(provincia);
                        key += anyo;
                        data = this.provinciasData.get(key);
                        //Parque
                        if(str.hasMoreTokens())
    data.setParque((Integer.parseInt(str.nextToken().replace(".",""))));

                }
    }
    catch(IOException e){
                System.out.println(e.getMessage());
    }
    finally{
                try{
                        if(parqueFR!=null)              parqueFR.close();
                }
                catch(IOException e){
                        System.out.println(e.getMessage());
                }
    }
}


public String buscarProvincia(String nombre){
    Iterator<String> provinciasIt = provincias.keySet().iterator();
    String provinciaId = null;
    boolean encontrado = false;

    while(!encontrado && provinciasIt.hasNext()){
                provinciaId = (String)provinciasIt.next();
                if(nombre.equals((String)this.provincias.get(provinciaId)))        encontrado = true;
    }
    return provinciaId;
}

public void grabarProvincias(){
    Configuration conf = new Configuration();
    conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
   conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

    try {
                FileSystem hdfs = FileSystem.get(conf);

                Path path = new Path(dwhPath+"provincias");
```

```
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();

                Iterator<String> iterator = provincias.keySet().iterator();
                String key = null;

                while(iterator.hasNext()){
                        key = (String)iterator.next();

                        sb.append(key);
                        sb.append("\t");
                        sb.append((String)provincias.get(key));
                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}

public void grabarVias(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

     try {
                FileSystem hdfs = FileSystem.get(conf);

                Path path = new Path(dwhPath+"vias");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();

                sb.append("Urbana");       sb.append("\t");   sb.append("Urbana");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Autopista");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Autovia");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Convencional");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Servicio");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Enlace");
     sb.append("\n");
                sb.append("Interurbana"); sb.append("\t");   sb.append("Otros");
     sb.append("\n");

                out.writeBytes(sb.toString());

     } catch (IOException e) {
                System.out.println(e.getMessage());
     }
}

public void grabarAccidentesUrbana(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));
```

```
        try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesUrbana");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();

                        sb.append(datos.getAnyo());
                        sb.append("\t");
                        sb.append(datos.getId());
                        sb.append("\t");
                        sb.append("Urbana");
                        sb.append("\t");
                        // Vía Urbana
                        sb.append(datos.getAccidentes_VU());
                        sb.append("\t");
                        sb.append(datos.getMuertos_VU());
                        sb.append("\t");
                        sb.append(datos.getHeridos_VU());
                        sb.append("\t");
                        sb.append(datos.getHeridosGraves_VU());
                        sb.append("\t");
                        sb.append(datos.getHeridosLeves_VU());
                        sb.append("\t");

                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}



public void grabarAccidentesAutopista(){
    Configuration conf = new Configuration();
    conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
   conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

    try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesAutopista");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();
```

```
                                sb.append(datos.getAnyo());
                                sb.append("\t");
                                sb.append(datos.getId());
                                sb.append("\t");
                                sb.append("Autopista");
                                sb.append("\t");
                                // Autopistas
                                sb.append(datos.getAccidentes_VIU_Autopista());
                                sb.append("\t");
                                sb.append(datos.getMuertos_VIU_Autopista());
                                sb.append("\t");
                                sb.append(datos.getHeridos_VIU_Autopista());
                                sb.append("\t");
                                sb.append(datos.getHeridosGraves_VIU_Autopista());
                                sb.append("\t");
                                sb.append(datos.getHeridosLeves_VIU_Autopista());
                                sb.append("\t");

                                sb.append("\n");
                        }

                        out.writeBytes(sb.toString());

                } catch (IOException e) {
                        System.out.println(e.getMessage());
                }
        }

        public void grabarAccidentesAutovia(){
             Configuration conf = new Configuration();
             conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
            conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

             try {
                        FileSystem hdfs = FileSystem.get(conf);
                        Path path = new Path(dwhPath+"accidentesAutovia");
                        FSDataOutputStream out = hdfs.create(path);

                        StringBuffer sb = new StringBuffer();
                        ProvinciaDatosTrafico datos = null;

                        Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                        while(iterator.hasNext()){
                                datos = (ProvinciaDatosTrafico)iterator.next();

                                sb.append(datos.getAnyo());
                                sb.append("\t");
                                sb.append(datos.getId());
                                sb.append("\t");
                                sb.append("Autovia");
                                sb.append("\t");
                                // Autovías
                                sb.append(datos.getAccidentes_VIU_Autovia());
                                sb.append("\t");
                                sb.append(datos.getMuertos_VIU_Autovia());
                                sb.append("\t");
                                sb.append(datos.getHeridos_VIU_Autovia());
                                sb.append("\t");
                                sb.append(datos.getHeridosGraves_VIU_Autovia());
                                sb.append("\t");
                                sb.append(datos.getHeridosLeves_VIU_Autovia());
```

```
                        sb.append("\t");

                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}

public void grabarAccidentesConvencional(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

     try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesConvencional");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();

                        sb.append(datos.getAnyo());
                        sb.append("\t");
                        sb.append(datos.getId());
                        sb.append("\t");
                        sb.append("Convencional");
                        sb.append("\t");
                        // Convencional
                        sb.append(datos.getAccidentes_VIU_Convencional());
                        sb.append("\t");
                        sb.append(datos.getMuertos_VIU_Convencional());
                        sb.append("\t");
                        sb.append(datos.getHeridos_VIU_Convencional());
                        sb.append("\t");
                        sb.append(datos.getHeridosGraves_VIU_Convencional());
                        sb.append("\t");
                        sb.append(datos.getHeridosLeves_VIU_Convencional());
                        sb.append("\t");

                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}

public void grabarAccidentesVecinal(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));
```

```
        try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesVecinal");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();

                        sb.append(datos.getAnyo());
                        sb.append("\t");
                        sb.append(datos.getId());
                        sb.append("\t");
                        sb.append("Vecinal");
                        sb.append("\t");
                        // Vecinal
                        sb.append(datos.getAccidentes_VIU_Vecinal());
                        sb.append("\t");
                        sb.append(datos.getMuertos_VIU_Vecinal());
                        sb.append("\t");
                        sb.append(datos.getHeridos_VIU_Vecinal());
                        sb.append("\t");
                        sb.append(datos.getHeridosGraves_VIU_Vecinal());
                        sb.append("\t");
                        sb.append(datos.getHeridosLeves_VIU_Vecinal());
                        sb.append("\t");

                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
    }

    public void grabarAccidentesServicio(){
        Configuration conf = new Configuration();
        conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
       conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

        try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesServicio");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();

                        sb.append(datos.getAnyo());
                        sb.append("\t");
```

```
                                sb.append(datos.getId());
                                sb.append("\t");
                                sb.append("Servicio");
                                sb.append("\t");
                                // Servicio
                                sb.append(datos.getAccidentes_VIU_Servicio());
                                sb.append("\t");
                                sb.append(datos.getMuertos_VIU_Servicio());
                                sb.append("\t");
                                sb.append(datos.getHeridos_VIU_Servicio());
                                sb.append("\t");
                                sb.append(datos.getHeridosGraves_VIU_Servicio());
                                sb.append("\t");
                                sb.append(datos.getHeridosLeves_VIU_Servicio());
                                sb.append("\t");

                                sb.append("\n");
                        }

                        out.writeBytes(sb.toString());

                } catch (IOException e) {
                        System.out.println(e.getMessage());
                }
        }

        public void grabarAccidentesEnlace(){
             Configuration conf = new Configuration();
             conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
            conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

             try {
                        FileSystem hdfs = FileSystem.get(conf);
                        Path path = new Path(dwhPath+"accidentesEnlace");
                        FSDataOutputStream out = hdfs.create(path);

                        StringBuffer sb = new StringBuffer();
                        ProvinciaDatosTrafico datos = null;

                        Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                        while(iterator.hasNext()){
                                datos = (ProvinciaDatosTrafico)iterator.next();

                                sb.append(datos.getAnyo());
                                sb.append("\t");
                                sb.append(datos.getId());
                                sb.append("\t");
                                sb.append("Enlace");
                                sb.append("\t");
                                // Enlace
                                sb.append(datos.getAccidentes_VIU_Enlace());
                                sb.append("\t");
                                sb.append(datos.getMuertos_VIU_Enlace());
                                sb.append("\t");
                                sb.append(datos.getHeridos_VIU_Enlace());
                                sb.append("\t");
                                sb.append(datos.getHeridosGraves_VIU_Enlace());
                                sb.append("\t");
                                sb.append(datos.getHeridosLeves_VIU_Enlace());
                                sb.append("\t");
```

```
                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}

public void grabarAccidentesOtros(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

     try {
                FileSystem hdfs = FileSystem.get(conf);
                Path path = new Path(dwhPath+"accidentesOtros");
                FSDataOutputStream out = hdfs.create(path);

                StringBuffer sb = new StringBuffer();
                ProvinciaDatosTrafico datos = null;

                Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                while(iterator.hasNext()){
                        datos = (ProvinciaDatosTrafico)iterator.next();

                        sb.append(datos.getAnyo());
                        sb.append("\t");
                        sb.append(datos.getId());
                        sb.append("\t");
                        sb.append("Otros");
                        sb.append("\t");
                        // Otros
                        sb.append(datos.getAccidentes_VIU_Otros());
                        sb.append("\t");
                        sb.append(datos.getMuertos_VIU_Otros());
                        sb.append("\t");
                        sb.append(datos.getHeridos_VIU_Otros());
                        sb.append("\t");
                        sb.append(datos.getHeridosGraves_VIU_Otros());
                        sb.append("\t");
                        sb.append(datos.getHeridosLeves_VIU_Otros());

                        sb.append("\n");
                }

                out.writeBytes(sb.toString());

        } catch (IOException e) {
                System.out.println(e.getMessage());
        }
}

public void grabarParque(){
     Configuration conf = new Configuration();
     conf.addResource(new Path("/etc/hadoop/conf/core-site.xml"));
    conf.addResource(new Path("/etc/hadoop/conf/hdfs-site.xml"));

     try {
                FileSystem hdfs = FileSystem.get(conf);
```

```
                        Path path = new Path(dwhPath+"parque");
                        FSDataOutputStream out = hdfs.create(path);

                        StringBuffer sb = new StringBuffer();
                        ProvinciaDatosTrafico datos = null;

                        Iterator<ProvinciaDatosTrafico> iterator = provinciasData.values().iterator();

                        while(iterator.hasNext()){
                                datos = (ProvinciaDatosTrafico)iterator.next();

                                sb.append(datos.getAnyo());
                                sb.append("\t");
                                sb.append(datos.getId());
                                sb.append("\t");
                                sb.append(datos.getParque());

                                sb.append("\n");
                        }

                        out.writeBytes(sb.toString());

                    } catch (IOException e) {
                            System.out.println(e.getMessage());
                    }
                }

                /**
                 * @param args
                 */
                public static void main(String[] args) {

                    String ficheroProvincias = "/home/cloudera/shared/inbox/accidentes/provincias.csv";
                    String ficheroAccidentes = "/home/cloudera/shared/inbox/accidentes/accidentes.csv";
                    String ficheroParque = "/home/cloudera/shared/inbox/accidentes/parque.csv";


                    LoadFilesDWH carga = new LoadFilesDWH();
                    carga.cargarProvincias(ficheroProvincias);
                    carga.cargarAccidentes(ficheroAccidentes);
                    carga.cargarParque(ficheroParque);
                    carga.grabarProvincias();
                    carga.grabarVias();

                    carga.grabarAccidentesAutopista();
                    carga.grabarAccidentesAutovia();
                    carga.grabarAccidentesConvencional();
                    carga.grabarAccidentesEnlace();
                    carga.grabarAccidentesOtros();
                    carga.grabarAccidentesServicio();
                    carga.grabarAccidentesUrbana();
                    carga.grabarAccidentesVecinal();

                    carga.grabarParque();

                }
}
```

7.2 Código Solución Business Intelligence

Para poder ejecutar el siguiente código se deben configurar los paths de ubicación de los ficheros a utilizar en disco (marcados en amarillo).

```
SET ThousandSep='.';
SET DecimalSep=',';
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='#.##0,00 €;-#.##0,00 €';
SET TimeFormat='h:mm:ss';
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff]';
SET MonthNames='ene;feb;mar;abr;may;jun;jul;ago;sep;oct;nov;dic';
SET DayNames='lun;mar;mié;jue;vie;sáb;dom';

        [provincias]:
        LOAD @1 as provinciaId,
            @2 as provinciaNombre
        FROM
        [Z:\data\dwh\provincias]
        (txt, codepage is 1252, no labels, delimiter is '\t', msq);

        [accidentes]:
        LOAD @1 as anyo,
            @2 as provinciaId,
            @3 as tipoSubvia,
            @4 as accidentes,
            @5 as muertos,
            @6 as heridos,
            @7 as heridosGraves,
            @8 as heridosLeves
        FROM
        [Z:\data\dwh\accidentesAutopista]
        (txt, codepage is 1252, no labels, delimiter is '\t', msq);


        OUTER JOIN(accidentes)
        LOAD @1 as anyo,
            @2 as provinciaId,
            @3 as tipoSubvia,
            @4 as accidentes,
            @5 as muertos,
            @6 as heridos,
            @7 as heridosGraves,
            @8 as heridosLeves
        FROM
        [Z:\data\dwh\accidentesAutovia]
        (txt, codepage is 1252, no labels, delimiter is '\t', msq);

        OUTER JOIN(accidentes)
        LOAD @1 as anyo,
            @2 as provinciaId,
            @3 as tipoSubvia,
            @4 as accidentes,
            @5 as muertos,
            @6 as heridos,
            @7 as heridosGraves,
            @8 as heridosLeves
        FROM
        [Z:\data\dwh\accidentesConvencional]
        (txt, codepage is 1252, no labels, delimiter is '\t', msq);
```

```
OUTER JOIN(accidentes)
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as tipoSubvia,
    @4 as accidentes,
    @5 as muertos,
    @6 as heridos,
    @7 as heridosGraves,
    @8 as heridosLeves
FROM
[Z:\data\dwh\accidentesEnlace]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

OUTER JOIN(accidentes)
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as tipoSubvia,
    @4 as accidentes,
    @5 as muertos,
    @6 as heridos,
    @7 as heridosGraves,
    @8 as heridosLeves
FROM
[Z:\data\dwh\accidentesOtros]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

OUTER JOIN(accidentes)
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as tipoSubvia,
    @4 as accidentes,
    @5 as muertos,
    @6 as heridos,
    @7 as heridosGraves,
    @8 as heridosLeves
FROM
[Z:\data\dwh\accidentesServicio]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

OUTER JOIN(accidentes)
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as tipoSubvia,
    @4 as accidentes,
    @5 as muertos,
    @6 as heridos,
    @7 as heridosGraves,
    @8 as heridosLeves
FROM
[Z:\data\dwh\accidentesUrbana]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

OUTER JOIN(accidentes)
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as tipoSubvia,
    @4 as accidentes,
    @5 as muertos,
    @6 as heridos,
    @7 as heridosGraves,
    @8 as heridosLeves
```

```
FROM
[Z:\data\dwh\accidentesVecinal]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);



[vias]:
LOAD @1 as tipoVia,
    @2 as tipoSubvia
FROM
[Z:\data\dwh\vias]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

[vehiculos]:
LOAD @1 as anyo,
    @2 as provinciaId,
    @3 as vehiculos
FROM
[Z:\data\dwh\parque]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);

[clima]:
LOAD Provincia as provinciaId,
    Periodo as anyo,
    Replace(Tmedia,'.',',') as Tmedia,
    Replace(Tmin,'.',',') as Tmin,
    Replace(Hmedia,'.',',') as Hmedia,
    Replace(Ptotal,'.',',') as Ptotal,
    Replace(Dias0,'.',',') as Dias0,
    Replace(Ddes,'.',',') as Hdes,
    Replace(Hsol,'.',',') as Hsol,
    Replace(Tmax,'.',',') as Tmax,
    Clase as Clase
FROM
[Z:\data\dwh\clima]
(txt, utf8, embedded labels, delimiter is ',', msq);

[data_run]:
LOAD Provincia as provinciaIA,
    Periodo as anyoIA,
    Replace(Tmedia,'.',',') as TmediaRun,
    Replace(Tmin,'.',',') as TminRun,
    Replace(Hmedia,'.',',') as HmediaRun,
    Replace(Ptotal,'.',',') as PtotalRun,
    Replace(Dias0,'.',',') as Dias0Run,
    Replace(Ddes,'.',',') as HdesRun,
    Replace(Hsol,'.',',') as HsolRun,
    Replace(Tmax,'.',',') as TmaxRun,
    Clase as ClaseRun
FROM
[Z:\data\dwh\clima_run]
(txt, utf8, embedded labels, delimiter is ',', msq);


LOAD @1 as provinciaIA,
    @2 as anyoIA,
    Replace(@3,'.',',') as target,
    Replace(@4,'.',',') as modeloutput,
    Replace(@4,'.',',') as loglikelihood
FROM
[Z:\data\dwh\logistic_out]
(txt, codepage is 1252, no labels, delimiter is ',', msq);
```

```
 [provinciasIA]:
LOAD @1 as provinciaIA,
    @2 as provinciaNombreIA
FROM
[Z:\data\dwh\provincias]
(txt, codepage is 1252, no labels, delimiter is '\t', msq);
```