Máster Big Data & Analytics

TRABAJO FIN DE MÁSTER

MONETIZACIÓN DEL DATO EMPRESA DE TELECOMUNICACIONES

Autor: Andrés Cárdenas Parra

Tutor: Jorge López-Malla Matute

Las Rozas, Septiembre de 2018

A Merchi.

Resumen

En la actualidad las empresas pueden guardar una mayor cantidad datos, como pueden ser datos de localización y compras de sus clientes, estado de funcionamientos de sus sistemas de producción o recopilación de datos desde diferentes fuentes (móviles, sensores, etc.). Además las nuevas tecnología he permitido realizar el análisis a los datos, conocido como Big Data & Analytics.

Con este incesante incremento de los datos, se hace necesaria una correcta gestión de estos utilizando tecnologías y herramientas enfocadas a manejar cantidades muy grandes de datos.

Teniendo en cuenta lo anterior, he considerado necesario realizar estudios que me permitan conocer este tipo de tecnologías y así poder incrementar mi valor como profesional en el mercado laboral.

Con el fin de poder consolidar los conocimientos adquiridos durante el curso y usar varias de estas tecnologías de manera conjunta, he decidido realizar el trabajo de fin de máster relativo al análisis de eventos de una empresa de telecomunicaciones con el fin de llegar a obtener un valor monetario de estos.

En este trabajo se incluyen las fases principales de un proyecto de Big Data & Analytics, desde la ingesta a la visualización del análisis realizado, pasando por la utilización de modelos de Machine Learning.

Palabras clave: Big Data, Análisis de datos, visualización, Machine Learning.

Abstract

Currently, each day companies can store huge amounts of data, such as location data and purchases of their customers, status of the operations of their production systems or data collection from different sources (mobile, sensors, etc.) and also analyze the data, known as Big Data & Analytics.

With the increasing data, it is necessary to implement a new way to manage data using new technologies and tools focused on handling very large amounts of data.

Taking into account the above, I have considered to conduct studies that allow me to learn this kind of technologies and thus be able to increase my value as a professional in the market.

In order to be able to consolidate the knowledge acquired during the course and to use several of these technologies together, I decided to carry out the master's final project related to the analysis of events of a telecommunications company in order to obtain monetary value from them.

This work includes the main phases of a Big Data & Analytics project, including data ingestion, visualization of the related analysis and using Machine Learning models.

Keywords: Big Data, Data analysis, data visualization, Machine Learning.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción y Objetivos

1.1 Introducción

Con el fin de consolidar los conocimientos adquiridos durante el curso, he decidido realizar el proyecto de fin de máster propuesto por el profesor Jorge López-Malla Matute llamado "Monetización del dato, empresa de telecomunicaciones".

Durante el trabajo se abarcarán las principales fases de un proyecto de Big Data & Analytics, lo que me permitirá utilizar conjuntamente varias de las tecnologías vistas y terminar de comprender el tratamiento completo del dato.

Las distintas fases genéricas de un proyecto con "Machine Learning" son:

- Limpieza y recolección de datos
- Generación de los datos que alimentarán el modelo
- Entrenar el modelo
- Interpretar los datos del modelo
- Predecir con nuevos datos con el modelo anteriormente obtenido
- Representación visual de los datos
- Obtener valor de los datos generados por el modelo

En la fase de recolección y limpieza de datos se guardarán los datos recibidos en un sistema de almacenamiento de larga duración, guardando siempre datos limpios y homogéneamente formateados.

Una vez limpios, se mezclarán y formatearán los datos para alimentar el modelo de aprendizaje automático con el cual se podrá realizar un análisis de las relaciones entre los datos.

Después de obtener los datos como los requiere el modelo, se entrenará un modelo. En concreto y como específica el enunciado, el modelo a usar es K-means, un modelo de agrupación (Clustering). También se guardará el modelo entrenado para su posterior uso.

Luego se pasará a la fase de análisis e interpretación de las agrupaciones obtenidas con el fin de poder entender los resultados y así conseguir el valor necesario.

En un siguiente paso, se procederá a la predicción con nuevos datos correctamente formateados, usando el modelo entrenado.

Para finalizar, se realizarán diferentes visualizaciones de los datos, para llegar a la obtención del valor monetario.

1.2 Enunciado Caso de uso

La empresa de telecomunicaciones *phone.inc* requiere de la realización de un proyecto con el que persiguen monetizar datos de sus antenas. Suministrarán un número indeterminado de fuentes heterogéneas de eventos telefónicos pero que siempre tendrán los siguientes datos:

- Identificador único de cliente
- Fecha (hasta el mili segundo) del evento telefónico.
- Identificador de la antena asociada a ese evento.

El número de eventos diarios puede variar pero se calcula aproximadamente un millón de eventos por día entre todos los eventos telefónicos. Estos eventos se suministraran de manera semanal y contendrán toda la información del día anterior.

A continuación se detallan los formatos de los ficheros que serán recibidos:

- TelephoneEvents:
  CSV con campos separados por ";"
  Se suministran sólo una vez por semana

  - Identificador único de cliente -> String
  - Fecha (hasta el mili segundo) del evento telefónico. -> String formato: HH/MM/YYYY-HH;MM:SS.mmm
  - Identificador de la antena asociada a ese evento. -> String

- Clientes:
  CSV con campos separados por ";"
  Se suministran sólo una vez por mes

  - ClientId -> String

  - Age -> Int
  - Gender -> String (F -> Female, M -> Male)
  - Nationality -> String
  - Civil Status -> String (Married/Single)
  - Socioeconomic Level -> String (High/Medium/Low)

- Antenas:
  CSV con campos separados por ";"
  Se suministran una única vez en el proyecto

  - AtennaID.-> String
  - Intensity -> Int (en metros)
  - X (Coordenada X) -> Float
  - Y (Coordenada Y) -> Float

- Cities:
  CSV con campos separados por ";"
  Se suministran una única vez en el proyecto

  - CityName.-> String
  - Population -> Int
  - X1 (Coordenada1) -> Float,Float
  - X2 (Coordenada2) -> Float,Float
  - X3 (Coordenada3) -> Float,Float
  - X4 (Coordenada4) -> Float,Float
  - X5 (Coordenada5) -> Float,Float

Estas 5 coordenadas hacen un "Boundary Box" con el que es posible comprobar que una antena pertenece a una localidad o a otra.

Para la obtención de patrones, los expertos de la empresa han decidido que se debe usar el algoritmo **K-means** alimentado con las **horas** en las que cada antena tiene actividad en una semana.

La empresa desea una herramienta que le permita, de la manera más usable posible, la extracción de valor de esa información.

Para ello se propone la clasificación de las distintas localizaciones de las antenas en **lugares de trabajo y hogares** basándose en la actividad de los eventos telefónicos suministrados.

Se debe decidir en dónde almacenar la información para obtener el mejor resultado a la hora de consultarlos mediante una herramienta de visualización.

Cualquier idea e iniciativa sobre cómo poder sacar más rendimiento de la información será bien recibida

Es requisito indispensable justificar cualquier elección o descarte de tecnología y de desarrollo, siendo esto lo más importante.

1.3 Objetivos

El objetivo de este proyecto es la consolidación de los conocimientos adquiridos durante el curso y sobre todo la aplicación de estos conocimientos en un trabajo práctico que permita el uso de las diferentes tecnologías aprendidas.

Además de la consolidación del conocimiento, el trabajo de fin de máster tiene como objetivo que, como alumnos, hagamos uso de manera conjunta de varias de las tecnologías aprendidas y consigamos su integración y sobre todo que seamos capaces de identificar la tecnología a usar en cada caso de uso que se nos presente en nuestra vida laboral como profesionales del dato.

1.4 Método K-means

A continuación se explica brevemente el funcionamiento del método de agrupación K-means, con el fin de entender el objetivo de su uso.

K-means es un método de agrupamiento, cuyo objetivo es la clasificación/partición de un conjunto de *n* observaciones en *k* grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano a un punto (llamado centroide). Es un método utilizado en minería de datos.

Descripción:

> *Dado un conjunto de observaciones (x_1, x_2, …, x_n), donde cada observación es un vector real de d dimensiones, k-means construye una partición de las observaciones en k conjuntos ($k \leq n$) a fin de minimizar la suma de los cuadrados dentro de cada grupo (WCSS en inglés): $S = \{S_1, S_2, \ldots, S_k\}$*
>
> $$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$
>
> *Donde μ_i es la media de puntos en S_i. [1]*

Un ejemplo gráfico donde se pueden observar los centroides:

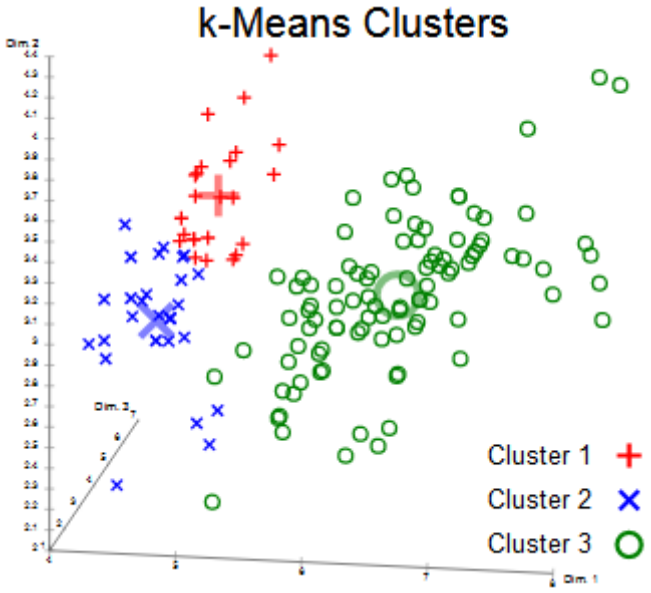


Figura 1. Ejemplo centroides K-means [2]

1.5 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

En el Capítulo 1 se muestra el enunciado del trabajo, se introduce y explica el trabajo realizado, los objetivos a conseguir, explicación del método K-Means y esta breve explicación de los diferentes capítulos.

En el Capítulo 2 se explican la parte más técnica como las herramientas Big Data usadas y descartadas para la realización de este trabajo y el entorno de desarrollo.

En el siguiente capítulo se detalla el trabajo realizado en cada una de sus fases.

Posteriormente, en el capítulo 4, se dan detalles acerca de la gestión del proyecto y los recursos usados.

Para finalizar, en el capítulo 5 se exponen las conclusiones del trabajo realizado.

Capítulo 2

Tecnología

En este capítulo serán expuestos aquellos detalles referentes a las tecnologías usadas y descartadas para la realización de este trabajo.

2.1 Herramientas Big Data seleccionadas

A continuación se detallan las tecnologías que han sido seleccionadas, indicando una descripción de las mismas y el motivo de su utilización.

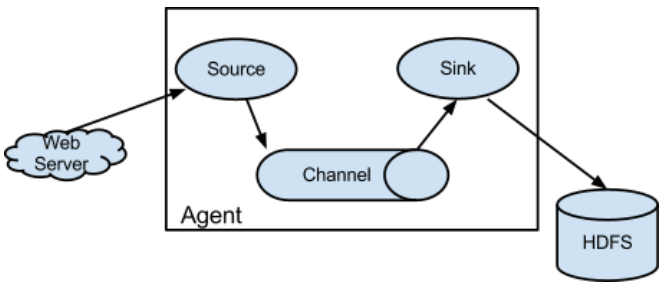
2.1.1 Apache Flume

Figura 2. Logo Apache Flume

Descripción:

Apache Flume es una herramienta que hace parte del ecosistema que rodea a Hadoop. Está construida en el lenguaje de programación Java y funciona de manera distribuida, lo que permite una alta disponibilidad.

Su funcionalidad es la de recolectar, agregar y mover grandes cantidades de datos de log. Su arquitectura es simple y flexible basada en streams de datos. [1]



Motivación:

Esta herramienta fue seleccionada principalmente debido a que su funcionalidad cubre perfectamente las necesidades de ingesta que se plantean en el proyecto.

Otro de los principales motivos el cual Apache Flume ha sido seleccionado, es por su integración nativa con HDFS (Hadoop Distributed File System), la tecnología que se usará para el almacenamiento de los datos, lo que garantiza un correcto funcionamiento en un entorno productivo.

2.1.2 Apache Hadoop (HDFS)

Figura 3. Logo Apache Hadoop

Descripción:

Hadoop es un framework para computación distribuida, fiable y escalable. Está hecho en el lenguaje de programación Java e incluye los siguientes módulos:

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop YARN
- Hadoop MapReduce

En concreto en este trabajo se han usado los módulos HDFS y Hadoop Common (por dependencias de HDFS).

HDFS es un sistema de ficheros distribuido que genera una capa de abstracción, así el programador no se encarga de la gestión de la distribución de los ficheros ni de su procesamiento, facilitando la programación y mantenimiento. Se considera un sistema robusto, según como sea configurado, ya que permite la réplica de los datos según se desee, logrando que los datos sigan siendo accesibles en caso de caída de alguno de los nodos, siendo esta gestión de manera automática.[4]

Motivación:

Principalmente se ha elegido esta tecnología debido a la gran cantidad de datos que espera recibir durante la "vida útil" del software, además de que dispone de integración nativa con Apache Spark, lo que ayudará y facilitará el procesamiento de los datos recibidos.

Teniendo en cuenta lo anterior, se usará HDFS como repositorio final de los datos agregados. Se ha tomado esta decisión debido a su integración nativa con Spark y que este a su vez permite disponer de conectores SQL para la conexión con los datos, lo que será aprovechado por la herramienta de visualización.

2.1.3 Apache Spark + Scala

Figura 4. Logo Apache Spark

Figura 5. Logo Scala

Descripción:

Apache Spark es un framework de computación de propósito general de alta velocidad que proporciona APIs en Java, Scala, Python y R. Soporta un gran conjunto de herramientas de alto nivel como puede ser Spark SQL para procesar datos estructurados y MLlib para la implementación de modelo de aprendizaje automático (Machine Learning).

Como lenguaje de programación para el uso del framework de computación Spark, se ha elegido Scala. [7][13]

Motivación:

El principal motivo de uso de esta tecnología es debido a su integración nativa con Apache Hadoop, así como también su gran cantidad de herramientas para Machine Learning que tiene disponibles.

Esto último ayudará con la creación y entrenamiento del modelo de Machine Learning K-means.

Otro de los motivos que me ha llevado a decantarme por esta tecnología, es que Spark en su distribución dispone de un servidor SQL llamado Spark Thrift Server, el cual facilitará la conexión de la herramienta de visualización con los datos.

La motivación principal para la elección de Scala como lenguaje de programación, ha sido que actualmente es uno de los lenguajes más demandado para Big Data & Analytics del mercado en este momento.

2.1.4 Apache Parquet

Figura 6. Logo Apache Parquet

Descripción:

Apache Parquet es un formato para el almacenamiento de datos basado en columnas.

Dicho formato es compatible con la mayoría de las herramientas existentes en el ecosistema Hadoop.

Sus principales características son:

- Compresión de los datos basándose en las columnas
- Compresión de los datos basándose en su tipo, aprovechando que en la mayoría de los casos los datos de la columna son del mismo tipo
- Las consultas a los datos que haga referencia a columnas específicas, solo leen datos de esas columnas, lo que aumenta el rendimiento[8]

Motivación:

Se ha seleccionado este formato de almacenamiento debido a su integración con el ecosistema Hadoop, pero sobre todo la mayor motivación se debe a su gran capacidad de compresión como se indica a continuación:

Tipos de compresión:

- Codificación por diccionario: automáticamente crea un diccionario que permite una compresión significativa y mejora los tiempos de procesamiento
- Bit Packing: en caso de números enteros pequeños, es capaz de almacenar múltiples valores en un mismo espacio reservado.
- Run-length encoding (RLE): para optimizar el espacio cuando hay múltiples ocurrencias de un mismo valor. Este se almacena una única vez con el total de apariciones.

2.1.5 Tableau Desktop

Figura 7. Logo Tableau

Descripción:

Tableau Desktop es uno de los productos desarrollados por la empresa Tableau. En concreto este producto es para el análisis de datos mediante representaciones de manera visual. [9]

Motivación:

Se ha seleccionado esta herramienta debido a su gran cantidad de conectores para el acceso a datos. Entre ellos se encuentra un conector a Spark SQL (Mediante el uso de un controlador JDBC/ODBC Simba), que permite visualizar los datos almacenados en HDFS mediante el acceso usando el metastore de Hive sin necesidad de hacer uso de Hive como tal. Esto proporciona la ventaja de que es posible realizar pre procesamientos de los datos usando Spark, lo que evita procesamiento de los datos durante la fase de visualización, en caso de ser necesario.

Otro de los motivos por lo que Tableau Desktop es debido a su gran aceptación por el mercado, lo que ocasiona que sea necesario tener un conocimiento de esta herramienta para que mi perfil profesional sea más atractivo en el mercado laboral.

2.2 Herramientas Big Data descartadas

2.2.1 Apache Flink

Figura 8. Logo Flink

Descripción:

Apache Flink es una plataforma para el procesamiento de datos de manera distribuida, tanto Streaming como Batch. Siendo el procesamiento en Streaming su punto más fuerte y el Procesamiento Batch un caso de uso especial del procesamiento en Streaming. Su motor tiene capacidades que permiten la distribución de los datos, comunicación y tolerancia a fallos en sistemas distribuidos. También dispone de librerías para la aplicación de modelo de Machine Learning en el procesamiento de datos. [14] [15]

Motivo Descarte:

El motivo de descarte de uso de esta tecnología se debe al limitado número de modelos de Machine Learning disponibles. Se ha revisado detalladamente la documentación disponible de Apache Flink y se ha detectado que no dispone del modelo requerido en el enunciado del caso de uso, K-means, lo que ha hecho que se descarte directamente.

También ha sido descartado por su naturaleza enfocada al procesamiento en Streaming y que su procesamiento en Batch está basado en micro Batching, cuando el procesamiento que se realiza es puramente Batch.

2.2.2 Apache Nifi

Figura 9. Logo Nifi

Descripción:

Apache Nifi es otro de los proyectos de software de la Apache Software Foundation. El objetivo de este software es la automatización de flujos de datos entre diferentes sistemas.

Apache Nifi soporta grafos dirigidos para el enrutamiento de datos de una manera escalable, así como también la transformación de los datos y dispone de lógicas para la mediación de sistemas. [6]

Motivo Descarte:

El motivo principal del descarte de este sistema es debido a que en el alcance de este trabajo los datos solo tienen un flujo posible, por tanto no justifica la necesidad de uso de esta herramienta. También ha sido motivo para su descarte el hecho de que las transformaciones a los datos se harán mediante Spark.

2.2.3 Apache Hive

Figura 10. Logo hive

Descripción:

Apache Hive es una infraestructura de almacenamiento (Data Warehouse) que ayuda a hacer más fácil la lectura, escritura y gestión de conjuntos de datos muy grandes que residen en un almacenamiento distribuido.

Hive es otra de las herramientas que hace parte de ecosistema Hadoop que usa como almacenamiento el sistema de ficheros HDFS. Ofrece un lenguaje de consultas basado en SQL llamado HiveQL. Las consultas realizadas en HiveQL finalmente son transformadas en tareas MapReduce o trabajos Spark. [10]

Motivo Descarte:

Hive se ha descartado debido a que la cantidad de ficheros y datos que se guardarían durante el desarrollo de este trabajo de fin de máster no requerían un sistema tan robusto como Hive, además de que Spark

dispone SparkThriftServer que provee las funcionalidades básicas necesarias SQL para la correcta y completa ejecución de este proyecto.

2.2.4 Apache Kafka

Figura 11. Logo Apache Kafka

Descripción:

Apache Kafka es una herramienta para el tratamiento de datos en tiempo real que usa el concepto de publicación/suscripción.

Es una herramienta distribuida altamente escalable y su objetivo es procesar los eventos o datos según ocurren. [11]

Motivo Descarte:

Apache Kafka se ha descartado ya que este trabajo de fin de máster no es un caso de uso aplicable a esta tecnología, ya que Kafka es una tecnología orientada al procesamiento de eventos según ocurren y en este caso, es necesario disponer de una gran cantidad de eventos para poder tomar una decisión. También es necesario tener en cuenta que la máxima frecuencia a la que se esperan datos es una vez por semana, lo que hace que no se justifique el uso de esta herramienta.

2.2.5 D3.js

Figura 12. Logo D3.js

Descripción:

D3.js (Data Driven Documents) es una librería JavaScript para la visualización Web interactiva de datos. D3.js hace uso de otras herramientas / tecnologías como SVG, HTML5 y CSS. [12]

Motivo Descarte:

Esta herramienta ha sido descartada ya que la visualización de los datos no se realizaría vía web y además se buscaba una herramienta que permita de una manera más sencilla tener diferentes formas de representación de los datos, cosa que se dificulta con D3.js.

2.2.6 Magellan

Descripción:

Magellan es un motor para analítica de datos geoespaciales. Está implementado sobre Apache Spark. Según comentan en su documentación, es la primera librería que extiende a Spark SQL para proveer de una abstracción para el análisis de datos geoespaciales. [16]

Motivo Descarte:

Está tecnología ha sido descartada por dos motivos principales: el primero de ellos es que dentro del alcance del proyecto no hay analítica de datos geoespaciales como tal, solo era necesario determinar la ciudad a la que pertenece cada una de las antenas, no siendo esto un motivo suficiente para justificar su uso. El segundo motivo es que durante el proceso de análisis se detectó una carencia de documentación, lo que hace difícil la integración y uso de esta tecnología.

2.2.7 GeoSpark

Figura 13. Logo GeoSpark

Descripción:

GeoSpark es un sistema de computación distribuida (en clúster) para procesar datos espaciales a gran escala. GeoSpark amplía Apache Spark / SparkSQL con un conjunto (RDD Espaciales) / SpatialSQL para la carga, procesamiento y análisis datos espaciales de gran escala de manera eficiente en todas las máquinas. [17]

Motivo Descarte:

Esta tecnología ha sido descartada ya que dentro del alcance del proyecto no hay analítica de datos geoespaciales como tal, unicamente se usaría esta tecnología para determinar la ciudad a la que pertenece cada una de las antenas, no siendo esto un motivo suficiente para justificar su uso.

2.3 Esquema arquitectura general del sistema

A continuación se muestra un esquema en donde se puede observar la arquitectura técnica del sistema que se ha decidido con el fin de conseguir los objetivos establecidos.

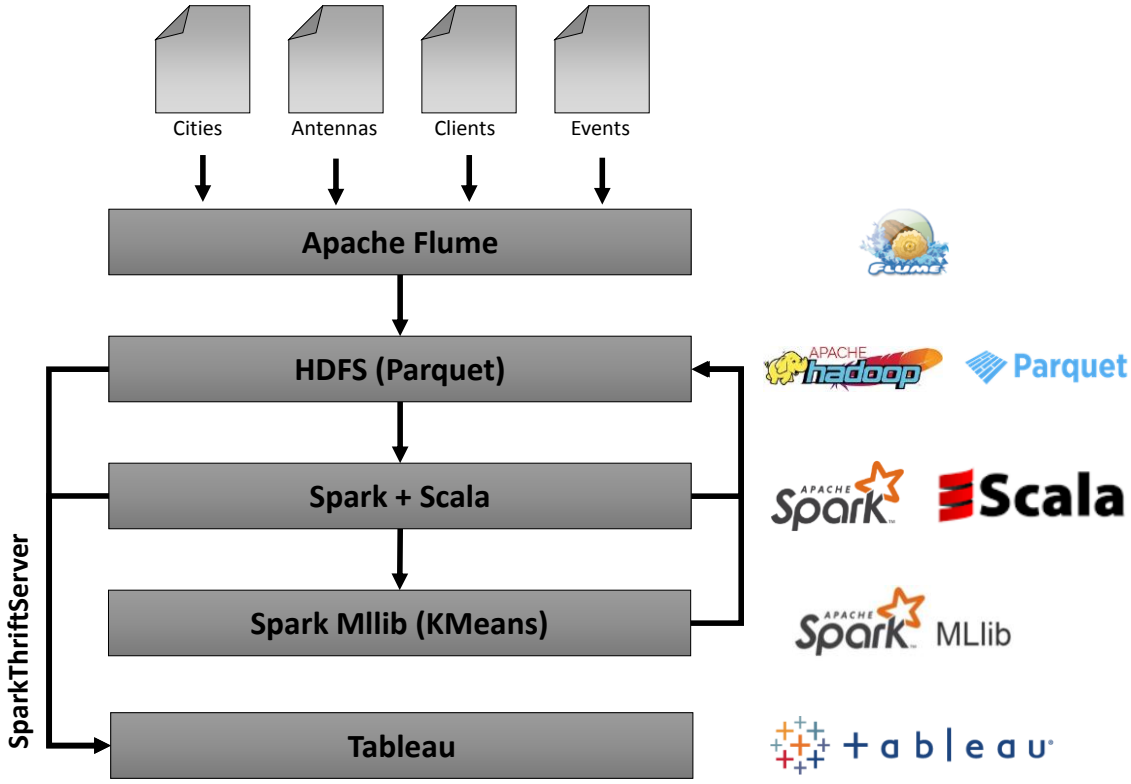


Figura 14. Arquitectura del sistema

2.4 Entorno de Desarrollo

Para la realización del desarrollo del código de este trabajo se ha usado el IDE de la empresa JetBrains llamado IntelliJ.

Para la compilación del código y ensamblado con dependencias del software desarrollado se ha usado la herramienta SBT.

El desarrollo se trata de un proyecto en Scala con diferentes módulos, que se detallan a continuación:

- Código Scala (src/main)

    - Carga & limpieza de los datos (/src/main/scala/dataLoading/*)
    - Generación y entrenamiento del modelo (/src/main/scala/ml/modelTrainWeekHours.scala)
    - Uso del modelo para realizar predicciones con nuevos datos (/src/main/scala/ml/modelPredictWeekHours.scala)
    - Elementos comunes (/src/main/scala/common/*)
- Ficheros de configuración para Apache Flume (flumeConf)
- Scripts de arranque de los agentes de Apache Flume (flumeStarters)
- Scripts SQL de creación de tablas en Spark Thrift Server (sqlScripts)

Con el fin de dejar trazas de los pasos que va realizando el código desarrollado en Scala durante su ejecución, se ha usado la librería log4j.

Para facilitar la posible modificación de las rutas de localización de los ficheros de entrada y los generados por el software, se ha decidido hacer uso de un fichero de configuración en donde se especifican las diferentes rutas en HDFS. Para el manejo y uso de este fichero de configuración se ha usado la librería com.typesafe.config.

En los anexos A y B se muestran los ficheros de configuración, tanto para log4j, como para la aplicación desarrollada, respectivamente.

Como repositorio de código, se ha usado GitHub, creando un repositorio, el cual se ha definido como privado hasta la finalización del máster.
https://github.com/andrescardenasp/antenas

Capítulo 3

Desarrollo

En este capítulo se expone el trabajo realizado explicando la solución desarrollada en cada una de las diferentes herramientas / tecnologías utilizadas.

3.1 Ingesta

Como se comentó en el capítulo anterior, para la realización de la ingesta se ha usado la herramienta Apache Flume.

Las especificaciones dadas en el enunciado del proyecto indicaban que se recibirían 4 ficheros, como se detalla a continuación:

- TelephoneEvents:
  CSV con campos separados por ";"
  Se suministran sólo una vez por semana
  - Identificador único de cliente -> String
  - Fecha (hasta el mili segundo) del evento telefónico. -> String formato: HH/MM/YYYY-HH;MM:SS.mmm
  - Identificador de la antena asociada a ese evento. -> String

- Clientes:
  CSV con campos separados por ";"
  Se suministran sólo una vez por mes
  - ClientId -> String
  - Age -> Int
  - Gender -> String (F -> Female, M -> Male)
  - Nationality -> String
  - Civil Status -> String (Married/Single)
  - Socioeconomic Level -> String (High/Medium/Low)

- Antenas:
  CSV con campos separados por ";"
  Se suministran una única vez en el proyecto
  - AtennaID.-> String
  - Intensity -> Int (en metros)
  - X (Coordenada X) -> Float
  - Y (Coordenada Y) -> Float

- Cities:
  CSV con campos separados por ";"
  Se suministran una única vez en el proyecto

  - CityName.-> String
  - Population -> Int
  - X1 (Coordenada1) -> Float,Float
  - X2 (Coordenada2) -> Float,Float
  - X3 (Coordenada3) -> Float,Float
  - X4 (Coordenada4) -> Float,Float
  - X5 (Coordenada5) -> Float,Float

A continuación se muestra un ejemplo de cada uno de ellos:

- TelephoneEvents

```
ClientId;Date;AntennaId
1665053N;05/06/2017-09:30:00.000;A01
```

- Clientes

```
ClientId;Age;Gender;Nationality;CivilStatus;SocioeconomicLevel
1666517V;50;F;JPN;Single;Low
```

- Antenas

```
AntennaId;Intensity;X;Y
A01;100;-3.710762;40.425788
```

- Ciudades

```
CityName;Population;X1;X2;X3;X4;X5
Madrid;3165541;-3.7906265259,40.3530853269;-3.5769081116,40.3530853269;-
3.5769081116,40.5349377098;-3.7906265259,40.5349377098;-
3.7906265259,40.3530853269
```

Cada uno de los ficheros se recibirá con una máscara diferente en el nombre, como se detalla a continuación:

Fichero	Nombre	Cabecera
TelephoneEvents	telephoneEvents-DD_MM_YYYY.csv	Si
Clientes	clients-DD_MM_YYYY.csv	Si
Antenas	Antennas-DD_MM_YYYY.csv	Si
Ciudades	cities-DD_MM_YYYY.csv	si

Tabla 1. Ficheros Nombre y cabecera

A continuación se indican las rutas donde serán recibidos cada uno de los ficheros, así como la ruta HDFS a la que serán ingestados por Apache Flume para su procesamiento posterior:

Fichero	Ruta Recepción	Ruta HDFS
TelephoneEvents	…/ReceivedFiles/events	/input/events
Clientes	…/ReceivedFiles/clients	/input/clients
Antenas	…/ReceivedFiles/antennas	/input/antennas
Ciudades	…/ReceivedFiles/cities	/input/cities

Tabla 2. Ficheros rutas de recepción y HDFS

Con todo lo anterior se dispone de toda la información necesaria para crear los ficheros de configuración de los diferentes agentes de Apache Flume.

Todos los ficheros de configuración para agentes buscan detectar ficheros que llegan al directorio indicado y copian dicho fichero en la ruta configurada de HDFS. Detalle de las principales características:

- Source
  - Tipo: Spool Directory
  - Canal: Se indica el canal respectivo
  - Spooldir: Se indica el directorio origen
  - Fileheader: True (todo los ficheros tienen cabecera)
- Channel
  - Tipo: memory (antenas, ciudades) file (events, clients)
- Sink
  - Tipo: HDFS
  - Filetype: Datastream
  - Path: ruta HDFS
  - Prefix: prefijo (coincide con el tipo de fichero)
  - Sufix: .csv
  - Batchsize: entre 1000 y 10000 (events)
  - rollSize: 33554432 (events)

- rollInterval: 40 (events)
- rollCount: 200000 (events)
- writeFormat: Text (events)
- callTimeout: 30000 (events)

A continuación se detallan los ficheros de configuración:

- TelephoneEvents

```
spool_events.sources = src-4
spool_events.channels = channel-4
spool_events.sinks = sink_to_hdfs4

# Source
spool_events.sources.src-4.type = spooldir
spool_events.sources.src-4.channels = channel-4
spool_events.sources.src-4.spoolDir =
/home/acp/Documents/TFM/receivedFiles/events
spool_events.sources.src-4.fileHeader = true
# HDFS sinks
spool_events.sinks.sink_to_hdfs4.type = hdfs
spool_events.sinks.sink_to_hdfs4.hdfs.fileType = DataStream
spool_events.sinks.sink_to_hdfs4.hdfs.path=hdfs://localhost:9000/input/events
spool_events.sinks.sink_to_hdfs4.hdfs.filePrefix = events_
spool_events.sinks.sink_to_hdfs4.hdfs.fileSuffix = .csv
spool_events.sinks.sink_to_hdfs4.hdfs.batchSize = 200000
spool_events.sinks.sink_to_hdfs4.hdfs.rollSize = 33554432
spool_events.sinks.sink_to_hdfs4.hdfs.rollInterval = 40
spool_events.sinks.sink_to_hdfs4.hdfs.rollCount = 200000
spool_events.sinks.sink_to_hdfs4.hdfs.writeFormat=Text
spool_events.sinks.sink_to_hdfs4.hdfs.callTimeout=30000
# Channel
spool_events.channels.channel-4.type = file
spool_events.channels.channel-4.capacity = 1000000
spool_events.channels.channel-4.transactionCapacity = 400000
spool_events.channels.channel-
4.checkpointDir=/home/acp/Documents/TFM/checkpoints/events
spool_events.channels.channel-4.dataDirs=/home/acp/Documents/TFM/data/events
# Bind the source and sink to the channel
spool_events.sources.src-4.channels = channel-4
spool events.sinks.sink to hdfs4.channel = channel-4
```

- Clientes

```
spool_clients.sources = src-4
spool_clients.channels = channel-3
spool_clients.sinks = sink_to_hdfs
# Source
spool_clients.sources.src-4.type = spooldir
spool_clients.sources.src-4.channels = channel-3
spool_clients.sources.src-4.spoolDir =
/home/acp/Documents/TFM/receivedFiles/clients
spool_clients.sources.src-4.fileHeader = true
# HDFS sinks
spool_clients.sinks.sink_to_hdfs.type = hdfs
spool_clients.sinks.sink_to_hdfs.hdfs.fileType = DataStream
spool_clients.sinks.sink_to_hdfs.hdfs.path=hdfs://localhost:9000/input/client
s
spool_clients.sinks.sink_to_hdfs.hdfs.filePrefix = clients_
spool_clients.sinks.sink_to_hdfs.hdfs.fileSuffix = .csv
spool_clients.sinks.sink_to_hdfs.hdfs.batchSize = 10000
spool_clients.sinks.sink_to_hdfs.hdfs.rollSize = 33554432
spool_clients.sinks.sink_to_hdfs.hdfs.rollInterval = 10
spool_clients.sinks.sink_to_hdfs.hdfs.rollCount = 0
spool_clients.sinks.sink_to_hdfs.hdfs.writeFormat=Text
spool_clients.sinks.sink_to_hdfs.hdfs.checkpointDir=/home/acp/Documents/TFM/c
heckpoints/clients
spool_clients.sinks.sink_to_hdfs.hdfs.dataDirs=/home/acp/Documents/TFM/data/c
lients
# Channel
spool_clients.channels.channel-3.type = file
spool_clients.channels.channel-3.capacity = 100000
spool_clients.channels.channel-3.transactionCapacity = 10000
# Bind the source and sink to the channel
spool_clients.sources.src-4.channels = channel-3
spool_clients.sinks.sink_to_hdfs.channel = channel-3
```

- Antenas

```
spool_antennas.sources = src-2
spool_antennas.channels = channel-2
spool_antennas.sinks = sink_to_hdfs
# Source
spool_antennas.sources.src-2.type = spooldir
spool_antennas.sources.src-2.channels = channel-2
spool_antennas.sources.src-2.spoolDir =
/home/acp/Documents/TFM/receivedFiles/antennas
spool_antennas.sources.src-2.fileHeader = true
# HDFS sinks
spool_antennas.sinks.sink_to_hdfs.type = hdfs
spool_antennas.sinks.sink_to_hdfs.hdfs.fileType = DataStream
spool_antennas.sinks.sink_to_hdfs.hdfs.path=hdfs://localhost:9000/input/anten
nas
spool_antennas.sinks.sink_to_hdfs.hdfs.filePrefix = antennas_
spool_antennas.sinks.sink_to_hdfs.hdfs.fileSuffix = .csv
spool_antennas.sinks.sink_to_hdfs.hdfs.batchSize = 1000
# Channel
spool_antennas.channels.channel-2.type = memory

# Bind the source and sink to the channel
spool_antennas.sources.src-2.channels = channel-2
spool_antennas.sinks.sink_to_hdfs.channel = channel-2
```

- Ciudades

```
spool_cities.sources = src-1
spool_cities.channels = channel-1
spool_cities.sinks = sink_to_hdfs
# Source
spool_cities.sources.src-1.type = spooldir
spool_cities.sources.src-1.channels = channel-1
spool_cities.sources.src-1.spoolDir =
/home/acp/Documents/TFM/receivedFiles/cities
spool_cities.sources.src-1.fileHeader = true
# HDFS sinks
spool_cities.sinks.sink_to_hdfs.type = hdfs
spool_cities.sinks.sink_to_hdfs.hdfs.fileType = DataStream
spool_cities.sinks.sink_to_hdfs.hdfs.path=hdfs://localhost:9000/input/cities
spool_cities.sinks.sink_to_hdfs.hdfs.filePrefix = cities_
spool_cities.sinks.sink_to_hdfs.hdfs.fileSuffix = .csv
spool_cities.sinks.sink_to_hdfs.hdfs.batchSize = 1000

# Channel
spool_cities.channels.channel-1.type = memory

# Bind the source and sink to the channel
spool_cities.sources.src-1.channels = channel-1
spool_cities.sinks.sink to hdfs.channel = channel-1
```

Con el fin de facilitar la puesta en marcha de los agentes de Apache Flume se han creados scripts para el arranque, uno por cada tipo de fichero.

El detalle de los scripts se puede observar en el código fuente entregado en el apartado flumeStarters.

3.2 Limpieza y formato de datos

Para la limpieza de los datos recibidos en los ficheros se ha usado Apache Spark.

En resumen, la limpieza de los datos se realiza de la siguiente manera:

1. Se detecta si hay ficheros o no de ciudades
    a. En caso afirmativo, se leen los ficheros de la ruta input/cities y se da formato al DataFrame
    b. Se guarda el DataFrame en la ruta cleanData/cities sobrescribiendo
    c. Se historifican los ficheros recibidos

2. Se detecta si hay ficheros o no de antenas
    a. En caso afirmativo, se leen los ficheros de la ruta input/antennas y se da formato al DataFrame

   b. Se guarda el DataFrame en la ruta cleanData/antennas sobrescribiendo
   c. Se historifican los ficheros recibidos

3. Se detecta si hay ficheros o no de clientes
   a. En caso afirmativo, se leen los ficheros de la ruta input/antennas y se da formato al DataFrame
   b. Se guarda el DataFrame en la ruta cleanData/antennas sobrescribiendo
   c. Se historifican los ficheros recibidos

4. Se detecta si hay ficheros o no de eventos
   a. En caso afirmativo, se leen los ficheros de la ruta input/antennas y se da formato al DataFrame
   b. Se crea un nuevo DataFrame haciendo uso de la función pivot con el fin de conseguir el vector de horas de las semana para cada par antena, cliente.
   c. Se guardan ambos DataFrames en las rutas cleanData/antennas y modeldata/data sobrescribiendo
   d. Se historifican los ficheros recibidos

A continuación se comentan los detalles específicos para la carga y limpieza de cada uno de los ficheros mencionados, siempre y cuando exista el fichero que se procede a cargar, ya que de lo contrario, se indica que no hay ficheros de ese tipo para realizar la carga.

Ciudades:

1. Se lee el fichero en un DataFrame , se especifica como delimitador ";", se toman los registros únicos.
2. Se renombran las columnas para que los nombres estén siempre en minúsculas.
3. Los campos que tienen una coordenada son divididos en dos columnas para facilitar su tratamiento.
4. Se eliminan las columnas no necesarias.
5. se imprime su esquema y se imprime una muestra de los datos.
6. Se guarda el DataFrame en la ruta cleanData/cities/.
7. Los ficheros recibidos en la ruta input/cities se mueven a la ruta input/citiesOld para historificarlos.

Antenas:

1. Se crea un customSchema para definir los nombres en minúsculas.
2. Se lee el fichero en un DataFrame, se especifica como delimitador ";", se toman los registros únicos.
3. Se lee el fichero con los datos limpios de ciudades.
4. Se determina la ciudad de cada antena:
   a. Se hace un cross join con ciudades.

   b. Se filtran quedándo con aquellos registros en los cuales la coordenada de la antena se encuentra dentro del boundary box que define a la ciudad.
   c. Se dejan sólo con las columnas relativas a las antenas y **el identificador de la ciudad a la que pertenecen.**
5. Se eliminan las columnas no necesarias.
6. se imprime su esquema y se imprime una muestra de los datos.
7. Se guarda el DataFrame en la ruta cleanData/antennas/.
8. Los ficheros recibidos en la ruta input/antennas se mueven a la ruta input/antennasOld/ para historificarlos.

Clientes:

1. Se crea un customSchema para definir los nombres en minúsculas.
2. Se lee el fichero en un DataFrame, se especifica como delimitador ";", se toman los registros únicos.
3. Se imprime su esquema y se imprime una muestra de los datos.
4. Se guarda el DataFrame en la ruta cleanData/antennas/.
5. Los ficheros recibidos en la ruta input/clients se mueven a la ruta input/clientsOld/ para historificarlos.

Eventos:

1. Se crea un customSchema para definir los nombres en minúsculas.
2. Se lee el fichero en un DataFrame, se especifica como delimitador ";", se toman los registros únicos.
3. Se lee el fichero con los eventos.
4. Se crea un nuevo DataFrame unicamente con los registros válidos, descartando aquellos registros que no cumplen con el formato de fecha correcto "dd/MM/yyyy-HH:mm:ss.SSS" sin que aplicación genere un fallo por ello.
5. Se crean nuevas columnas desglosando el campo fecha:

```
.withColumn("time", split(col("Date"), "-").getItem(1))
.withColumn("date", split(col("Date"), "-").getItem(0))
.withColumn("day", split(col("Date"), "/").getItem(0))
.withColumn("month", split(col("Date"), "/").getItem(1))
.withColumn("year", split(col("Date"), "/").getItem(2))
.withColumn("dayofweek", date_format(to_date(col("Date"), "dd/MM/yyyy"),
"u"))
.withColumn("hour", split(col("Time"), ":").getItem(0))
.withColumn("minute", split(col("Time"), ":").getItem(1))
.withColumn("weekhour", concat(col("dayofweek"),lit("-"),col("hour")))
```

6. Se crea un nuevo DataFrame que va a contener los vectores por usuario y antena de total de eventos por cada hora de la semana:
   a. Se crea un DataFrame con las 168 horas de la semana.

   b. Se hace Right Join del DataFrame de registros válidos y el de horas de la semana (para garantizar tener todas las horas de la semana, incluso en las que no hay eventos).
   c. Se añade una nueva columna con el conteo de los eventos por hora de la semana agrupando por cliente y antena.
   d. Se hace un pivot del DataFrame para dejar los datos manera horizontal.
   e. Se agrega dejando solo el primero de los conteos.
   f. Se filtran aquellos registros que no tienen informados los campos cliente y antenaid.
   g. Se rellenan con 0 los nulos.
7. Se imprimen los esquemas de los DataFrame con datos válidos y el de vectores de horas de la semana y finalmente se imprime una muestra de los datos.
8. Se guarda el DataFrame de datos validos en la ruta cleanData/events/.
9. Se guarda el DataFrame de vector de horas de la semana en la ruta modeldata/data/.
10. Los ficheros recibidos en la ruta input/events se mueven a la ruta input/eventsOld/ para historificarlos.

3.3 Almacenamiento

Como se comentaba en el capítulo anterior, la tecnología de almacenamiento seleccionada ha sido Hadoop Distributed File System (HDFS).

Debido a que se dispone de un clúster con un solo nodo y que el objetivo final de este trabajo es académico, no se ha considerado necesaria la replicación.

Para determinar el tamaño de bloque, se han analizado los datos de entrada y se ha visto que los ficheros recibidos tienen un tamaño medio de 17 Mega Bytes aproximadamente, se ha decidido que el tamaño de bloque debe de ser de 16 Mega bytes. También se ha visto influida esta decisión ya que se trata de un entorno no productivo y que no se recibe una cantidad exagerada de datos.

Para guardar los datos generados por el software se ha optado por usar el formato Parquet, debido a su capacidad de compresión, lo que en un entorno productivo puede llegar a ahorrar muchísimo espacio y por tanto costes.

A continuación se muestra la estructura de directorios para el almacenamiento de los datos en HDFS que se ha definido:

Directorio	Subdirectorio	Uso
input		Guardar los ficheros que se reciben
input	cities	Guardar temporalmente los ficheros de ciudades que se reciben
input	clients	Guardar temporalmente los ficheros de clientes que se reciben
input	antennas	Guardar temporalmente los ficheros de antenas que se reciben
input	events	Guardar temporalmente los ficheros de eventos que se reciben
input	citiesOld	Guardar histórico de los ficheros de ciudades recibidos
input	clientsOld	Guardar histórico de los ficheros de clientes recibidos
input	antennasOld	Guardar histórico de los ficheros de antenas recibidos
input	eventsOld	Guardar histórico de los ficheros de eventos recibidos
cleanData		Guardar los datos limpios
cleanData	cities	Guardar los datos limpios de ciudades más actualizados
cleanData	clients	Guardar los datos limpios de clientes más actualizados
cleanData	antennas	Guardar los datos limpios y enriquecidos de antenas más actualizados
cleanData	events	Guardar los datos limpios y enriquecidos de eventos más actualizados
modeldata		Guardar los datos que se usan o se generan por el modelo de ML
modeldata	data	Guarda los datos que se usan para entrenar al modelo
modeldata	modelTrain	Guardar el modelo entrenado para un uso posterior
modeldata	predictions	Guardar las ultimas predicciones hechas por el modelo

Tabla 3. Estructura de directorios HDFS

Se ha determinado la necesidad de guardar los datos originales con el fin de poder usarlos en un futuro para la detección de cambios en los patrones. Teniendo en cuenta la normativa actual, es necesario poder tener un seguimiento de las transformaciones realizadas a los datos en cada momento, lo que hace necesario disponer de los datos originales para que en caso de una auditoría de las autoridades competentes, sea posible determinar el conjunto de las trasformaciones realizadas para llegar al resultado final, partiendo del dato original.

3.4 Machine Learning

En este apartado se detallan los pasos realizados para el entrenamiento, guardado y uso del modelo de Machine Learning K-means.

3.4.1 Entrenamiento del Modelo

Con el fin de poder crear, entrenar y guardar el modelo Machine Learning especificado por el cliente (K-means), se realizan los siguientes pasos.

1. Carga de los datos para el modelo en un DataFrame. Se trata de los datos generados que contienen por cada par cliente-antena un vector de horas de la semana.
2. Una vez cargados los datos en el DataFrame, se hace uso de uno de los transformadores de características disponibles en Spark, vectorAssembler. Con este transformador se consigue generar el vector de características en el formato que necesita el modelo. Como parámetro se le pasa el listado ordenado de las horas de la semana, que se corresponden con las columnas del DataFrame.
3. Se define la configuración del modelo que va a ser usado:
    a. K = 2
    b. Columna de características = features
    c. Nombre dela columna para albergar los resultados de la predicción = prediction
4. Se entrena el modelo ejecutando el método fit disponible en el modelo definido en el paso anterior.
5. Se imprimen los vectores centroides con el fin de poder interpretar el modelo generado.
6. Se guarda el modelo entrenado en la ruta modeldata/modelTrain, para que pueda ser usado posteriormente para realizar predicciones.

A continuación se muestra el fragmento de código que realiza los pasos comentados:

```
val assembler = new
VectorAssembler().setInputCols(common.utils.weekHoursList.toArray)/.setO
utputCol("features")
val K-means = new K-means()
.setK(2)
.setFeaturesCol("features")
.setPredictionCol("prediction")
dfEventsAntenasWeekHours.show()
val assembled = assembler
.transform(dfEventsAntenasWeekHours)
val K-meansPredictionModel = K-means
.fit(assembled)
for ( center <- K-meansPredictionModel.clusterCenters) println(center)
K-meansPredictionModel.write.overwrite().save(modelLocation)
```

3.4.2 Predicción

Una vez que el modelo está entrenado con los datos para el entrenamiento, lo siguiente es usarlo para realizar predicciones con nuevos datos. Para lo cual se realizan los pasos:

1. Carga de los datos para el modelo en un DataFrame. Se trata de los datos limpios, que se han agrupado para obtener por cada par cliente-antena un vector de horas de la semana.
2. Una vez cargados los datos en el DataFrame se hace uso de uno de los transformadores de características disponibles en Spark, vectorAssembler. Con este transformador conseguir generar el vector de características en el formato que necesita el modelo. Como parámetro se le pasa el listado ordenado de las horas de la semana que se corresponden con las columnas del DataFrame.
3. Se carga el modelo que había sido previamente guardado.
4. Se realiza la predicción pasando al modelo el DataFrame con el vector de características.
5. Se elimina la columna “features” ya que no es necesaria.
6. Se guardan los resultados de la predicción en la ruta modeldata/predictions.

A continuación se muestra el fragmento de código que realiza los pasos mencionados:

```
val dfEventsAntenasWeekHours =
sq.read.parquet(parameters.getString("hdfs.modeldata.data"))
val K-meansPredictionModel = K-meansModel.read.load(modelLocation)
val assembler = new VectorAssembler()
.setInputCols(common.utils.weekHoursList.toArray)
.setOutputCol("features")
val assembled = assembler
.transform(dfEventsAntenasWeekHours)
val predictionResult = K-meansPredictionModel
.transform(assembled)
.drop("features")
predictionResult.coalesce(1).write.mode(SaveMode.Overwrite).parquet(pred
ictionsData)
```

3.5 Funcionamiento

3.5.1 Modos de ejecución

Con el fin de desarrollar un solo software que permita realizar varias tareas, ha sido necesario la ejecución del software en modos que se detallan a continuación:

1. Modo 1: Carga de datos nuevos y entrenamiento
2. Modo 2: Carga de datos nuevos y predicción
3. Modo 3: Obtención de esquemas de los datos guardados
4. Modo 4: Carga de datos únicamente
5. Modo 5: Solo entrenamiento con datos ya existentes
6. Modo 6: Solo predicción con datos ya existentes

Para ejecutar el software en cada uno los modos indicados, solo hace falta pasar el número del modo deseado por parámetro.

3.5.2 Lanzamiento de procesos

Aunque la ejecución automática de los procesos que se desarrollaron en este trabajo se encuentra fuera del alcance, es muy importante tenerlo en cuenta, ya que es un punto determinante para la elección de tecnologías a usar.

Para la ejecución de los procesos hace falta tener en cuenta los siguientes aspectos:

- Frecuencia de ejecución

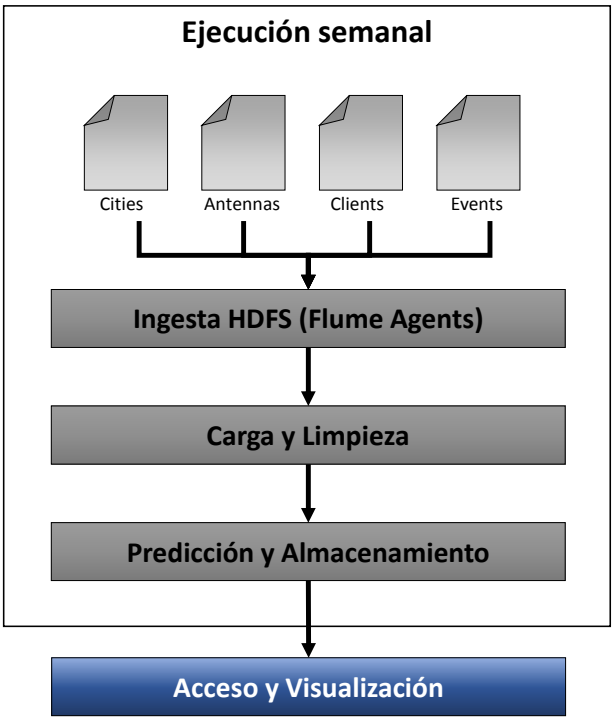
- Dependencias entre procesos
- Dependencia de datos de entrada
- Si existen o no ejecuciones únicas o cargas iniciales
- Modos de ejecución del software desarrollados

Con todo lo anterior en mente se definen las siguientes necesidades de ejecución:

- Ejecución inicial: esta ejecución se debe de realizar de manera manual en el momento de la puesta en producción del software desarrollado. Consiste en una primera ejecución en la cual se cargan los datos y se entrena el modelo por primera vez. Se debe realizar en el **Modo 1** de ejecución.

- Ejecución semanal: Se trata de la ejecución normal del software. Se realiza una vez a la semana el día en el que los datos que se reciben los datos semanales (eventos). Existe una dependencia con la llegada de los ficheros. Se debe de ejecutar en el **Modo 2** de ejecución.

  A continuación diagrama que explica las dependencias y el flujo de los datos:



- Entrenamientos posteriores: Se trata de una ejecución mensual, cuando los datos de clientes son actualizados. Se debe ejecutar en **Modo 1** de ejecución.

- Ejecuciones Manuales: Se trata de ejecuciones puntuales en las que sea necesario ejecutar el software en cualquiera de sus modos con el fin de resolver una posible incidencia o por cualquier otro motivo.

- Servicios Activos: siempre debe estar activo el servicio de SparkThriftServer para que así el usuario pueda disponer de los datos en cualquier momento para su análisis mediante Tableau.

Como ejemplo, se muestra el comando con el que se ejecuta el SW en Modo 1. Este mismo comando se puede usar con diferentes herramientas:

```
spark-submit --class Main
/home/acp/Documents/TFM/Codigo/antenas/target/scala-2.11/antenas-
assembly-0.1.jar 1
```

Para conseguir realizar de manera automática las ejecuciones con la planificación indicada, en el mercado hay disponibles muchas herramientas como las que se listan a continuación:

- Apache Oozie
- Chronos
- Crontab
- Control-M
- WizeHive
- Etc…

En una primera implantación de este software en un entorno productivo, se recomienda el uso de una de herramienta genérica, como puede ser crontab, para automatizar la ejecución de este software.

En una futura fase de mejora de este trabajo se recomienda el uso una herramienta más específica. Antes de tomar cualquier decisión respecto a la herramienta a usar, es necesario realizar un análisis previo para determinar la idoneidad y la viabilidad de cada una de las herramientas y así tomar la mejor decisión.

3.6 Corrección de los datos recibidos

Durante la fase de pruebas se ha detectado que los datos contenían un error que hacía que las conclusiones pudiesen no ser las correctas, además de dificultar el análisis de los datos.

El problema en los datos consistía en que el campo "Date" del fichero de eventos, en donde se guarda la fecha y hora del día, no tenía el formato correcto y no permitía diferenciar si una hora era AM o PM, haciendo que el sistema interpretara la hora siempre como AM. Por ejemplo, si un evento se generaba a las 3 PM, este evento era clasificado como a las 3 AM ocasionando una gran distorsión que afectaría a las conclusiones.

El "cliente" (tutor), manifestó que no era posible volver a generar los datos de manera correcta lo que me llevó a realizar un análisis extra de los datos para determinar si era posible corregir los datos. He considerado muy importante poder realizar la interpretación de la manera más natural posible sin distorsionar los resultados y conclusiones.

Una vez terminado el análisis detallado del formato de los datos se determinó que sí era posible arreglarlos, lo que me ha llevado a desarrollar un módulo extra para proceder con la corrección.

Este nuevo módulo se ha realizado también en Scala, sin ser incluido en el proyecto principal. El algoritmo para la corrección de cada uno de los registros que vienen en los ficheros es el siguiente:

Se crean las siguientes variables:
antenaActual: Id de antena registro en curso
antenaAnterior: Id de antena de registro anterior
suma: Booleano que indica si se debe sumar 12 o no
horaActual: Campo hora del registro actual
horaAnterior: Campo hora del registro anterior

1. Si es la primera línea, se trata de la cabecera del fichero y por tanto debe de ser escrita directamente en el nuevo fichero.
2. En caso de tratarse de la segunda línea del fichera, esta representa la primera de datos reales, se guarda directamente en el fichero y con sus datos se informan las variables antenaAnterior y horaAnterior.
3. Para el resto de líneas se procede de la siguiente manera:
    a. Se establecen las variables antenaActual y horaActual con los datos del registro en curso.
    b. Si en caso de que la antena anterior y la actual sean la misma y la hora actual sea menor que la anterior, se

informa la variable suma = TRUE para indicar que se deben sumar 12 horas la hora.

c. Si en caso de que la antena anterior y la actual sean diferentes, se informa la variable suma = FALSE para indicar que se no deben sumar 12 horas la hora.

d. En caso de ser necesario sumar, se guarda el registro con la corrección y en caso contrario, se guarda el mismo registro.

e. Las variables antenaAnterior y horaAnterior pasan a tener los valores de las variables antenaActual y horaActual respectivamente.

Todo esto teniendo siempre en cuenta que algunos de los registros pueden tener un formato erróneo a propósito, ya que estos registros sirven para comprobar la tolerancia a fallos del sistema.

Se realizaron pruebas unitarias de corrección de los datos y final se procedió a la corrección masiva de todos los datos.

El código fuente que contiene la solución completa se encuentra en el siguiente repositorio de GitHub:
https://github.com/andrescardenasp/FixTFMData

3.7 Agrupación de los datos

Tal y como se ha detallado en el apartado "3.2. Limpieza y formato de datos", se realiza una agrupación de los datos con el fin de obtener un Dataset en el formato requerido por el modelo K-means.

La agrupación consiste en generar un vector de actividad por cada hora de la semana por cada par –cliente antena. Se está manera se obtiene un registro que muestra el número total de eventos de cada persona en una antena determinada en cada hora de la semana.

La agrupación se realiza a partir de los datos ya cargados en HDFS que han sido previamente formateados y se han eliminado los registros que tienen un formato erróneo. Para conseguir la agrupación mencionada se han aprovechado las herramientas provistas por Spark para el tratamiento y análisis de los datos, como se detalla a continuación:

1. Se crea un nuevo DataFrame manualmente que contiene una columna por cada hora de la semana, en total son 168 (7 días de la semana * 24 horas del día).

2. Se hace Right Join del DataFrame de registros válidos con el creado en el paso anterior. De esta manera es posible garantizar tener todas las horas de la semana en el vector de actividad, incluso las horas en las que no hay actividad.
3. Se añade una nueva columna con el conteo de los eventos por hora de la semana particionando por cliente y antena.
4. Se pivota el DataFrame para conseguir que en las columnas queden disponibles las horas de la semana.
5. Se usa la función de agregación dejado solo la primera ocurrencia de la columna de conteo previamente creada.
6. Se eliminan aquellos registros que no tienen informados los campos cliente y antenaid. Es posible que se de esta casuística debido al Right Join realizado previamente.
7. Finalmente se rellenan con ceros los campos nulos.

Al realizar esta agrupación se consigue que el modelo K-means se capaz de generar los vectores centroides basándose en la actividad que tiene cada persona en cada antena. Lo que finalmente permitirá identificar patrones de comportamiento y así diferenciar si antena se encuentra ubicada en una zona residencial o de trabajo.

Además de conseguir el funcionamiento del modelo, al agruparse los datos se consigue reducir el número de registros, lo que significa una reducción muy importante del tamaño que ocupan los datos.

Para poder obtener aún más valor de los datos, estos se deben de agrupar basándose en las diferentes variables categóricas que tenemos disponibles, nacionalidad, género, nivel socioeconómico, estado civil y edad. Con estas diferentes agrupaciones es posible obtener mucha más información de la que ya se ha podido conseguir y además, abrirá las puertas a nuevas posibilidades para la obtención de un beneficio económico de los datos.

3.8 Interpretación del modelo

Una vez que se ha entrenado el modelo, es necesario observar y comparar los dos vectores que definen los centroides de cada clúster definido por el modelo K-means y que además se usarán para las predicciones futuras.

El análisis consiste en conseguir distinguir y describir las diferencias en cada uno los vectores intentando inferir el significado de estas diferencias. De esta manera, va a ser posible indicar las características que hacen diferentes un clúster de otro, ya que hasta este momento lo

único que se tiene es un resultado numérico que es difícilmente explicable a otra persona.

Como se comentó en el apartado 3.4.1, durante la ejecución se han mostrado los vectores que definen a los clústeres.

A continuación se muestran dichos vectores, tal y como los muestra el sistema.

Vector clúster 0:

```
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,199.99885,199.99885,0.0,0.0,0.0,0.0,0.0,199.99885,0.0,0
.0,0.0,0.0,199.99885,199.99885,199.99885,199.99885,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,199.99885,199.99885,0.0,0.0,0.0,0.0,199.99885,199.99885,199.99885,199.99885,19
9.99885,199.99885,199.99885,199.99885,199.99885,199.99885,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,199.99885,199.99885,0.0,0.0,0.0,0.0,199.99885,199.99885,199.99885,199
.99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,199.99885,199.99885,0.0,0.0,0.0,0.0,199.99885,199.99885,199.9
9885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,199.99885,199.99885,0.0,0.0,0.0,0.0,199.99885,199.99
885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,19
9.99885,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,100.99945000000001,199.
99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,100.99945000000001,199.9
9885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,199.99885,0
.0,0.0,0.0]
```

Vector clúster 1:

```
[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.99945000000001,119.99945000000001,119.9
9945000000001,119.99945000000001,119.99945000000001,0.0,119.9994500000000
1,119.99945000000001,119.99945000000001,119.99945000000001,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.99945000000001,119.99945000000001,119
.99945000000001,119.99945000000001,119.99945000000001,0.0,119.99945000000
001,119.99945000000001,119.99945000000001,119.99945000000001,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.99945000000001,119.99945000000001,1
19.99945000000001,119.99945000000001,119.99945000000001,0.0,119.999450000
00001,119.99945000000001,119.99945000000001,119.99945000000001,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.99945000000001,119.9994500000000
1,119.99945000000001,119.99945000000001,119.99945000000001,0.0,119.9994500
0000001,119.99945000000001,119.99945000000001,119.99945000000001,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.99945000000001,119.999450000000
01,119.99945000000001,119.99945000000001,119.99945000000001,0.0,119.999450
00000001,119.99945000000001,119.99945000000001,119.99945000000001,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0]
```

Como se puede apreciar, a simple vista es muy difícil comparar ambos vectores y llegar a describirlos, así que para ello he decidido darles un formato como si fuesen una tabla, separarlos por días de la

semana y hora del día, marcar en color aquellos valores que tengan valor (mayor a cero) y quitar los decimales si no aportan valor, quedando de la siguiente manera:

	Lunes																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	*200*	*200*	0	0	0	0	0	*200*	0	0	0	0	*200*	*200*	*200*	*200*	0
1	0	0	0	0	0	0	0	0	0	*120*	*120*	*120*	*120*	*120*	0	*120*	*120*	*120*	*120*	0	0	0	0	0

	Martes																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	*200*	*200*	0	0	0	0	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0
1	0	0	0	0	0	0	0	0	0	*120*	*120*	*120*	*120*	*120*	0	*120*	*120*	*120*	*120*	0	0	0	0	0

	Miércoles																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	*200*	*200*	0	0	0	0	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0
1	0	0	0	0	0	0	0	0	0	*120*	*120*	*120*	*120*	*120*	0	*120*	*120*	*120*	*120*	0	0	0	0	0

	Jueves																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	*200*	*200*	0	0	0	0	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0
1	0	0	0	0	0	0	0	0	0	*120*	*120*	*120*	*120*	*120*	0	*120*	*120*	*120*	*120*	0	0	0	0	0

	Viernes																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	*200*	*200*	0	0	0	0	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0
1	0	0	0	0	0	0	0	0	0	*120*	*120*	*120*	*120*	*120*	0	*120*	*120*	*120*	*120*	0	0	0	0	0

	Sábado																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	0	0	0	0	0	*101*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Domingo																							
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	0	0	0	0	0	*101*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	*200*	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Después de representar los vectores de esta manera, es posible observar y definir características de cada uno de los vectores:

Características del vector k=0:

- Existen valores los fines de semana entre las 12h y las 20h.
- En días laborables, suele tener valores entre las 7h y las 8h (mañana), las 14h y a partir de las 19h.
- No tiene valores en horas de la noche.
- En días laborables suele tener el mismo patrón, pero en fines de semana cambia.

Características del vector k=1:

- No tiene información para los fines de semana.
- En días laborables, el patrón es el mismo.

- En días laborables, hay información en dos bloques horarios, de 9h a 14h y de 15h a 18h.

Con estas características es posible realizar las siguientes afirmaciones:

- El vector K = 0, dispone de más información en horario no laboral que en horario laboral.
- El vector K = 1 solo dispone de información en días laborables.
- Una posible explicación a que el vector K = 1 no dispone de información entre semana entre las 14h y las 15h podría ser que la gente se desplaza a otra zona de la ciudad.
- Una posible explicación a que el que vector K = 0 disponga de información entre semana entre las 14h y las 15h podría ser que la gente se desplaza a esta zona de la ciudad.
- Una posible explicación a que el que vector K = 0 disponga de información entre semana entre las 15h y las 18h, pero no entre las 9h y las 13h, podría ser que hay personas que trabajan únicamente media jornada, y por tanto se desplazan a su casa.

Después de haber analizado las características que describen a cada vector y las posibles afirmaciones, se puede concluir que el vector K = 0 se trata de una antena ubicada en una zona residencial, mientras que el vector K = 1 se trata de una zona de trabajo.

Vector clúster K = 0 → Antena ubicada en zona residencial
Vector clúster K = 1 → Antena ubicada en zona de trabajo

3.9 Representación de los datos

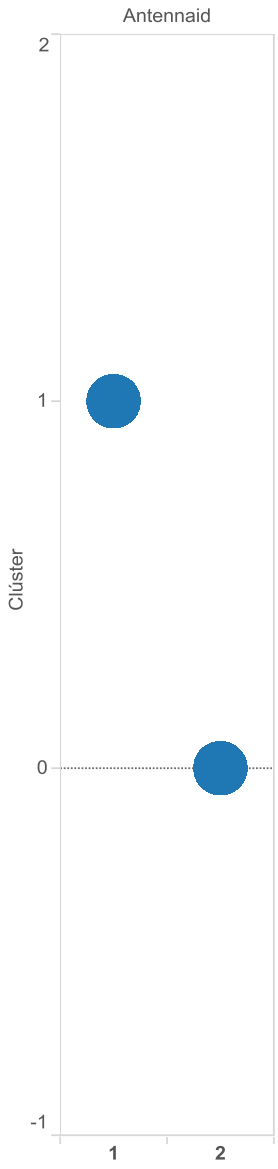
Para poder realizar mejor análisis de los datos es necesario que la salida de estos ayude a hacer más fácil dicho análisis.

Para facilitar el análisis de los datos se dispone de dos tipos de salidas. La primera con los datos originales, con un formato homogéneo y enriquecido. La segunda que proporciona los datos agregados en vectores de hora de la semana por cliente y antena, que incluye el clúster asignado una vez que los datos han sido procesados usando el modelo de ML.

A continuación se muestran las diferentes representaciones visuales que se han realizado de los datos y qué información proporciona cada una de ellas:

3.9.1 Antena Vs Clúster

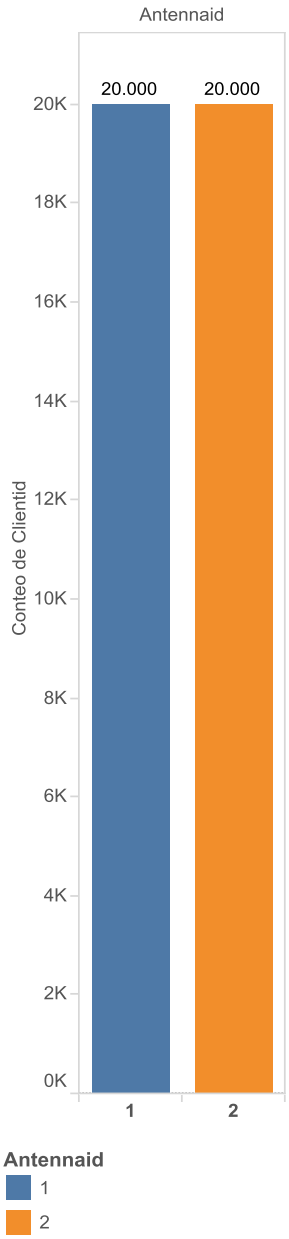
Para determinar el clúster al que pertenece cada antena se ha representado mediante un diagrama de puntos en el que se cruzan el identificador de cada antena y el clúster



Se ve claramente que el la antena A01 hace referencia al clúster 1, por tanto es la antena ubicada en una zona de trabajo. Así mismo se ve que la antena A02 es la que hace referencia al clúster 0 y por tanto está ubicada en una zona residencial.

3.9.2 Total de clientes por antena

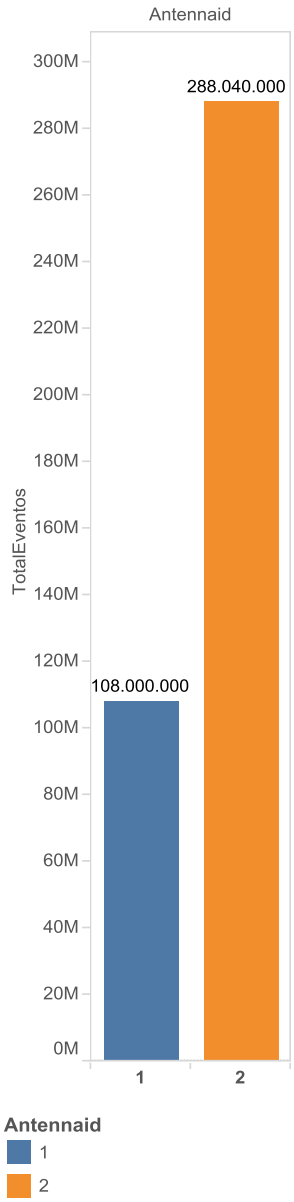
Se ha representado mediante un diagrama de barras la cantidad de clientes que hacen uso de cada antena:



Se observa que es exactamente el mismo número de clientes en cada antena, lo que indica que todas las personas que viven en la zona residencial trabajan en la zona de trabajo. En caso de tener más antenas y haciendo uso de los datos demográficos, podría ser posible llegar a determinar el tipo de trabajo que se desarrolla cerca de cada antena y por qué clase de personas.

3.9.3 Total de eventos por antena

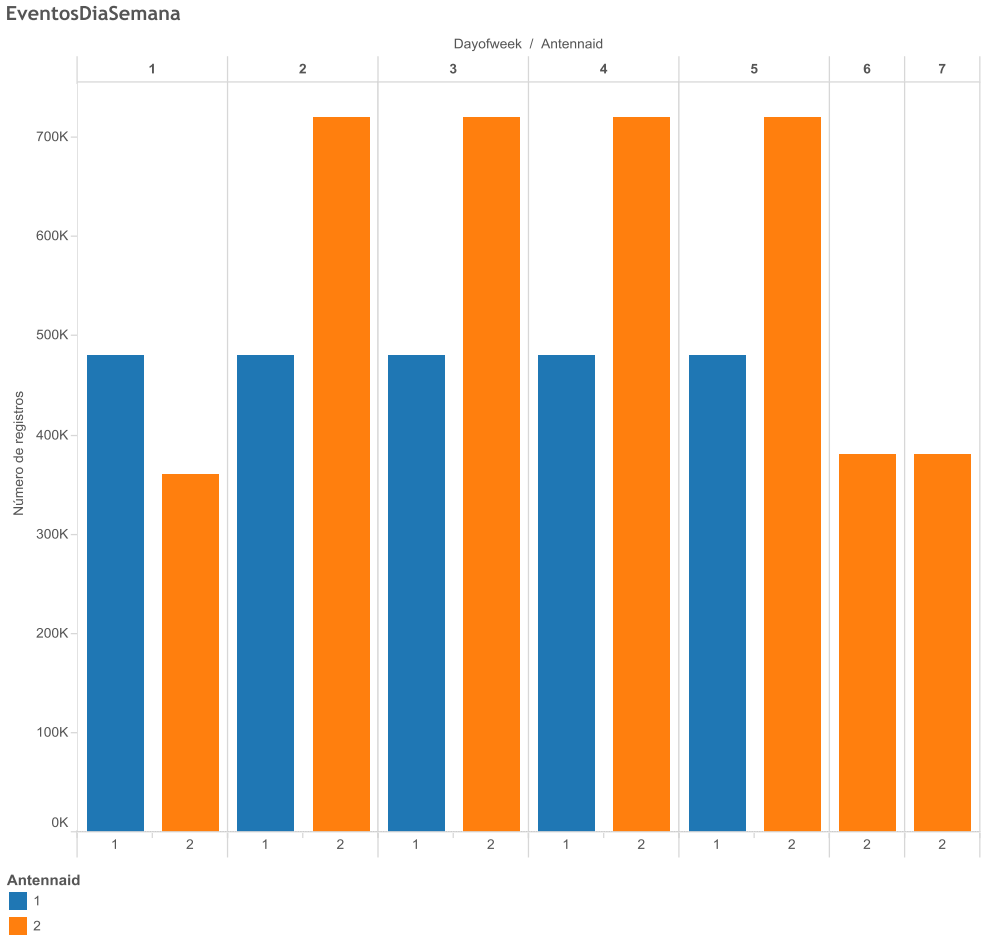
Se ha elegido un diagrama de barras para comparar el total de eventos por cada una de las antenas.



Como se puede observar, hay mucha más actividad en la antena ubicada en la zona residencial. Esto ayuda a corroborar que efectivamente se trata de una antena en una zona residencial, ya que entre los fines de semana, las mañanas y las noches, las personas suelen pasar, en total, más tiempo en la zona residencial que en el trabajo.

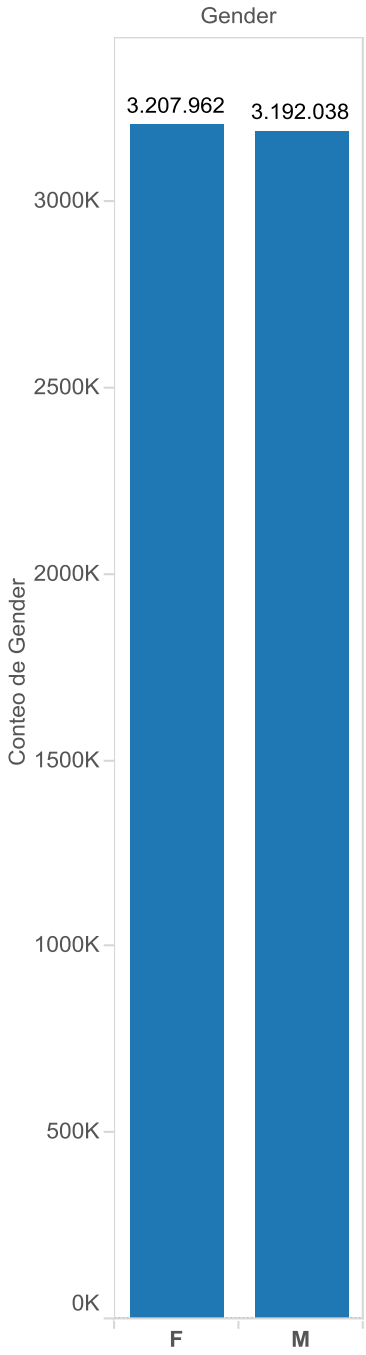
3.9.4 Eventos día de la semana por antena

Con el fin de comparar la cantidad de eventos por día en cada una de las antenas, se ha decido usar un diagrama de barras.



La imagen muestra claramente que la antena A02 es la única que tiene eventos durante los fines de semana, indicio claro que se trata de una antena ubicada en zona residencial y que la antena A01 se trata de una antena en zona de trabajo.

3.9.5 Eventos por género.

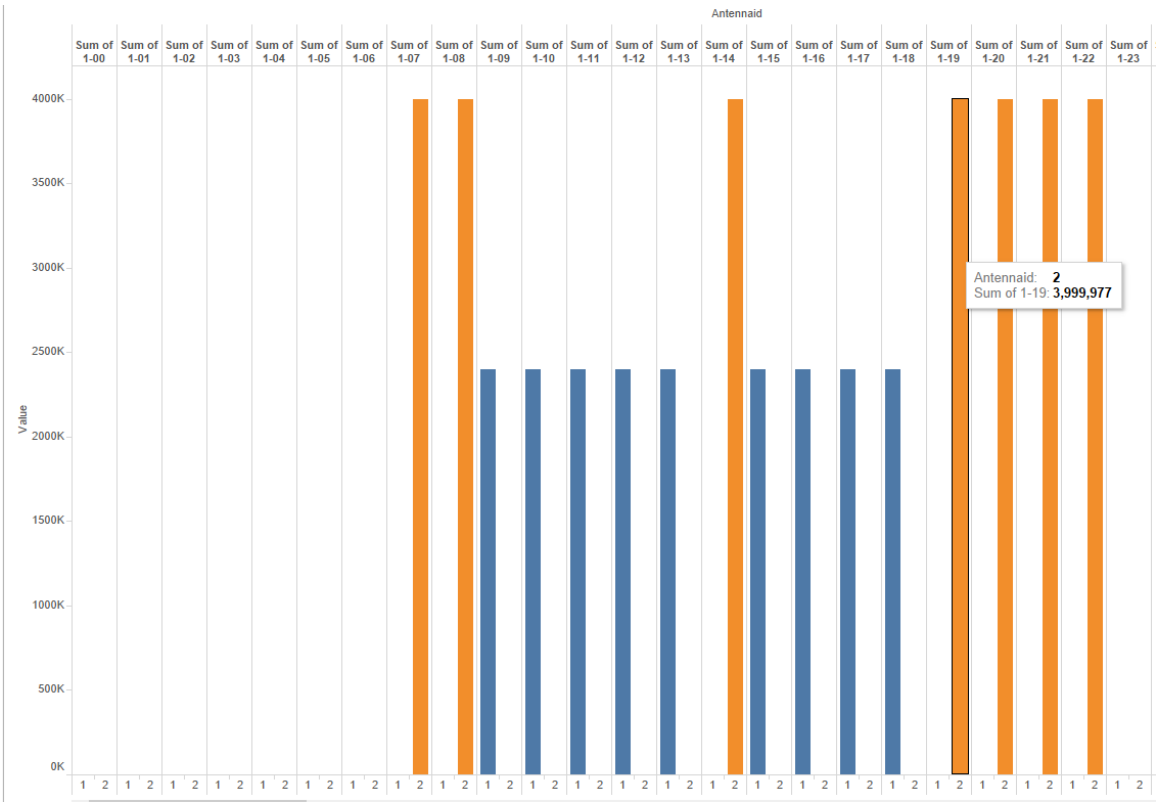


Como se poder ver, tanto los hombres como las mujeres tienen una actividad similar, lo que al final no brinda mucha información, si es que se quisiese hacer una clasificación basándose en el género.

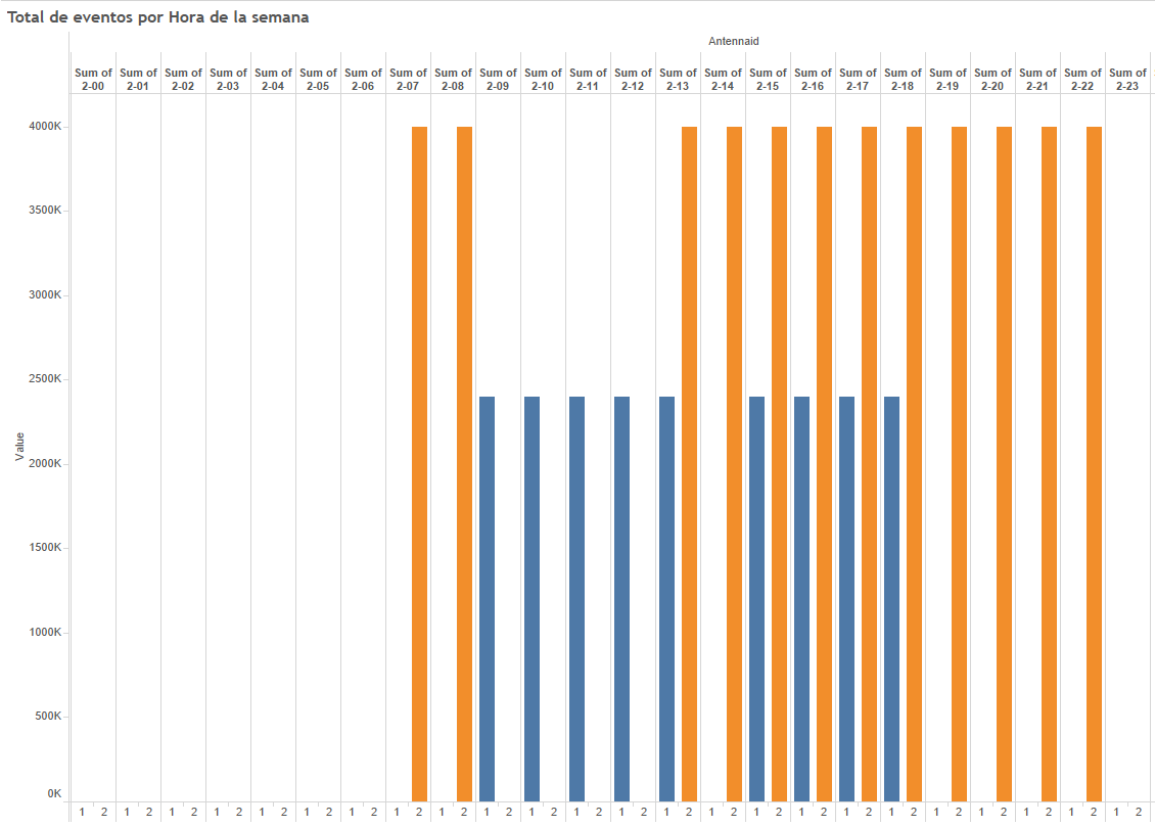
3.9.6 Eventos por antena y hora de la semana

Durante la representación se ha visto que el grafico se hace muy amplio, por tanto se ha decidido partirlo por los días de la semana, lo que su vez ayuda a comparar y detectar patrones.

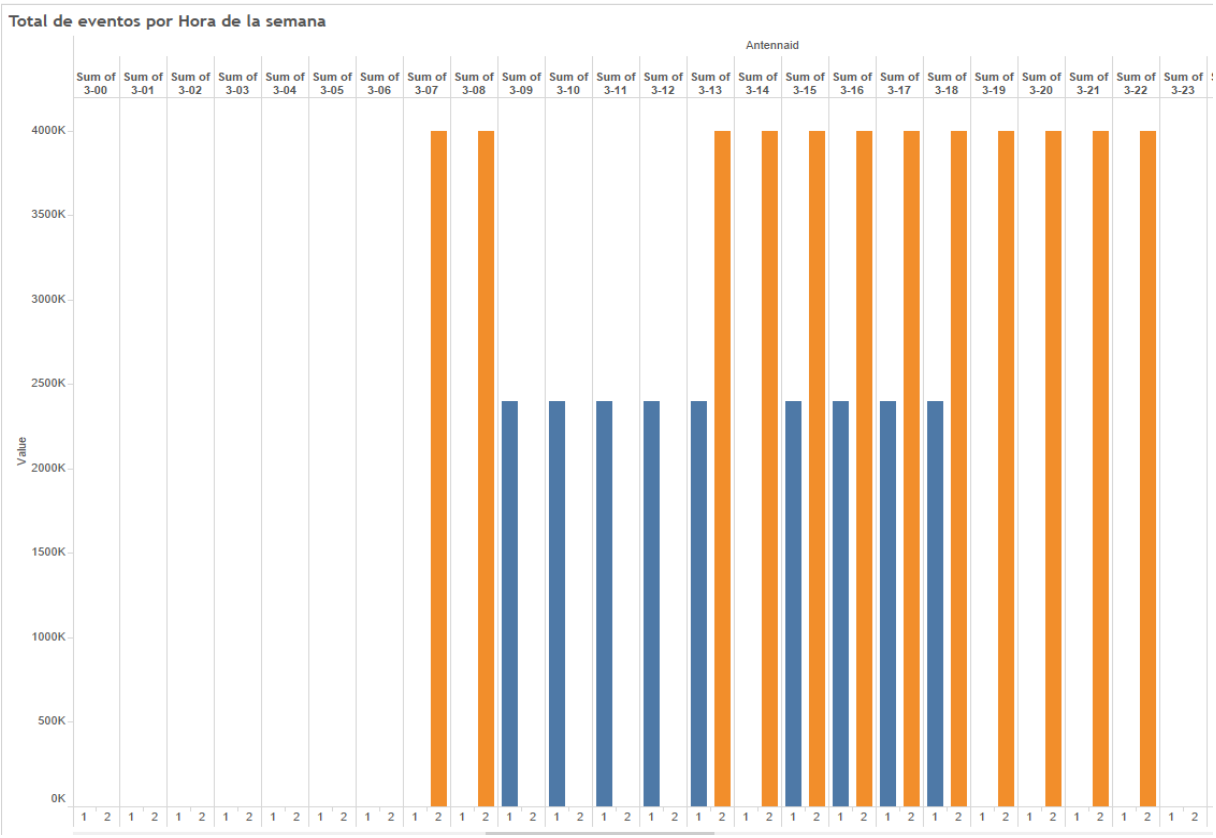
Lunes:



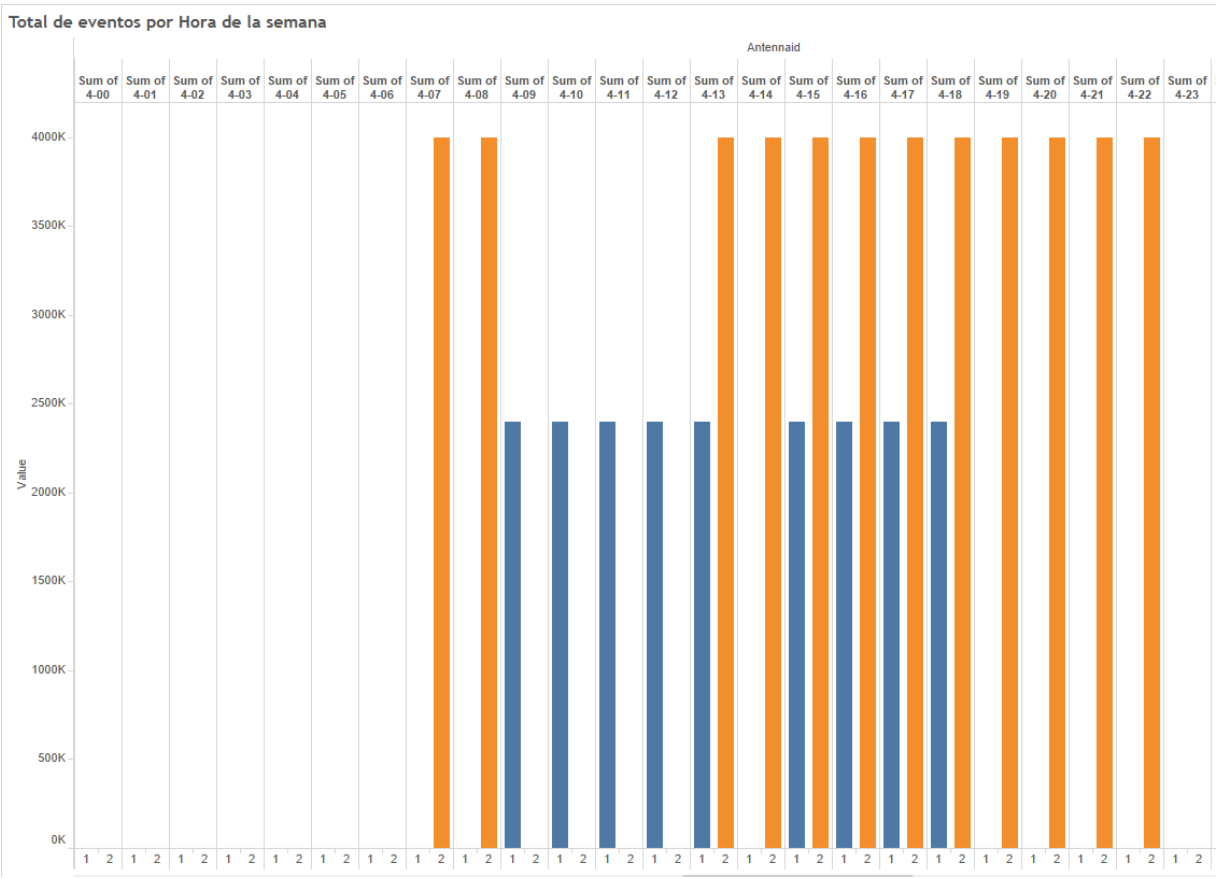
Martes:



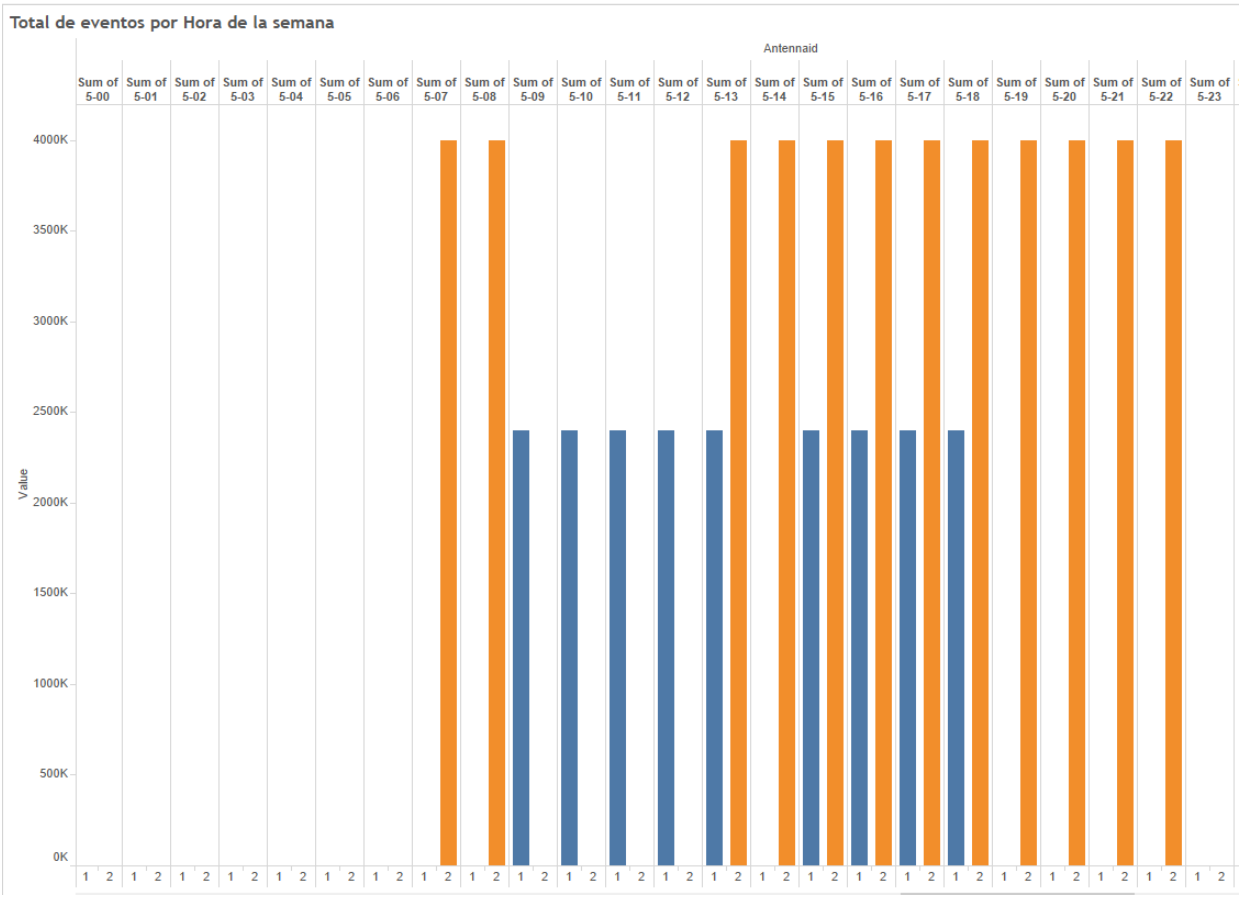
Miércoles:



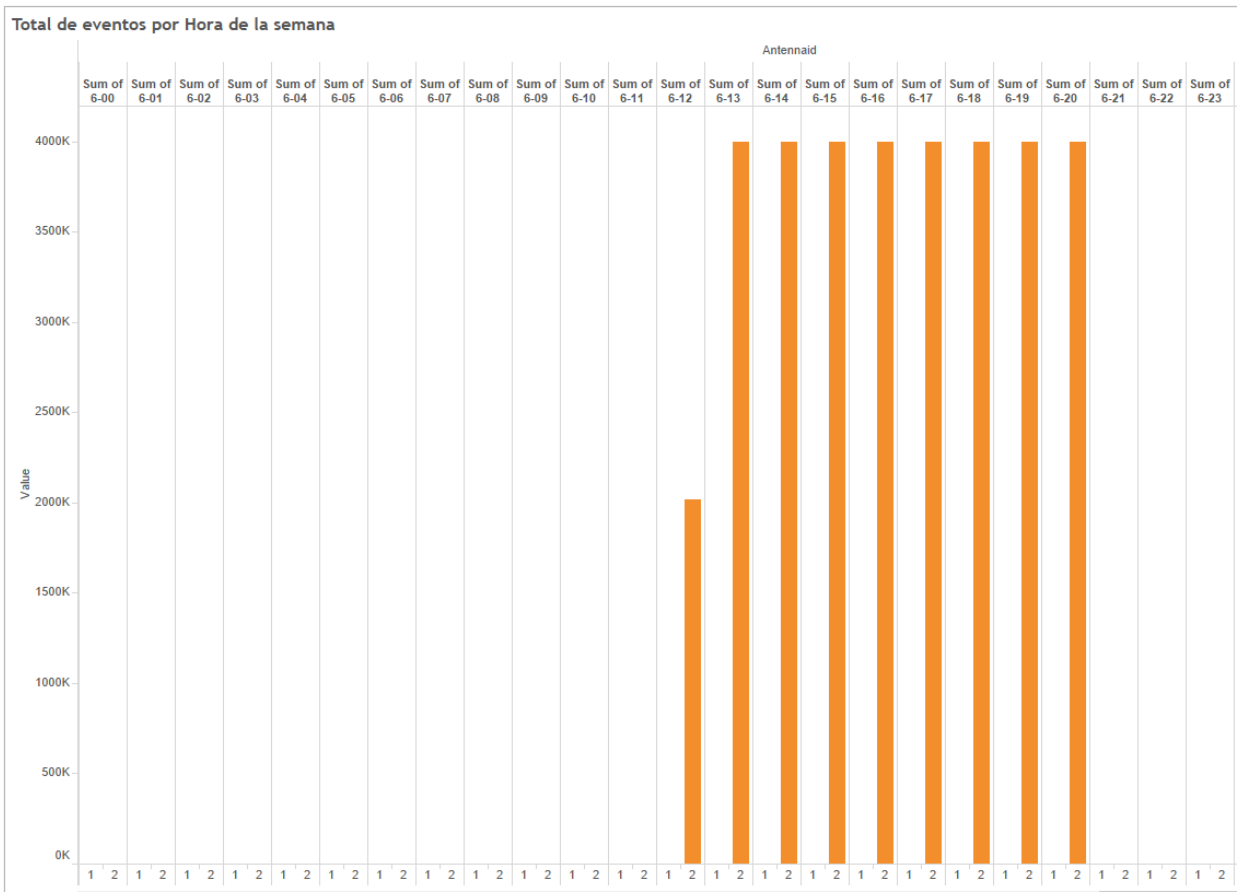
Jueves:



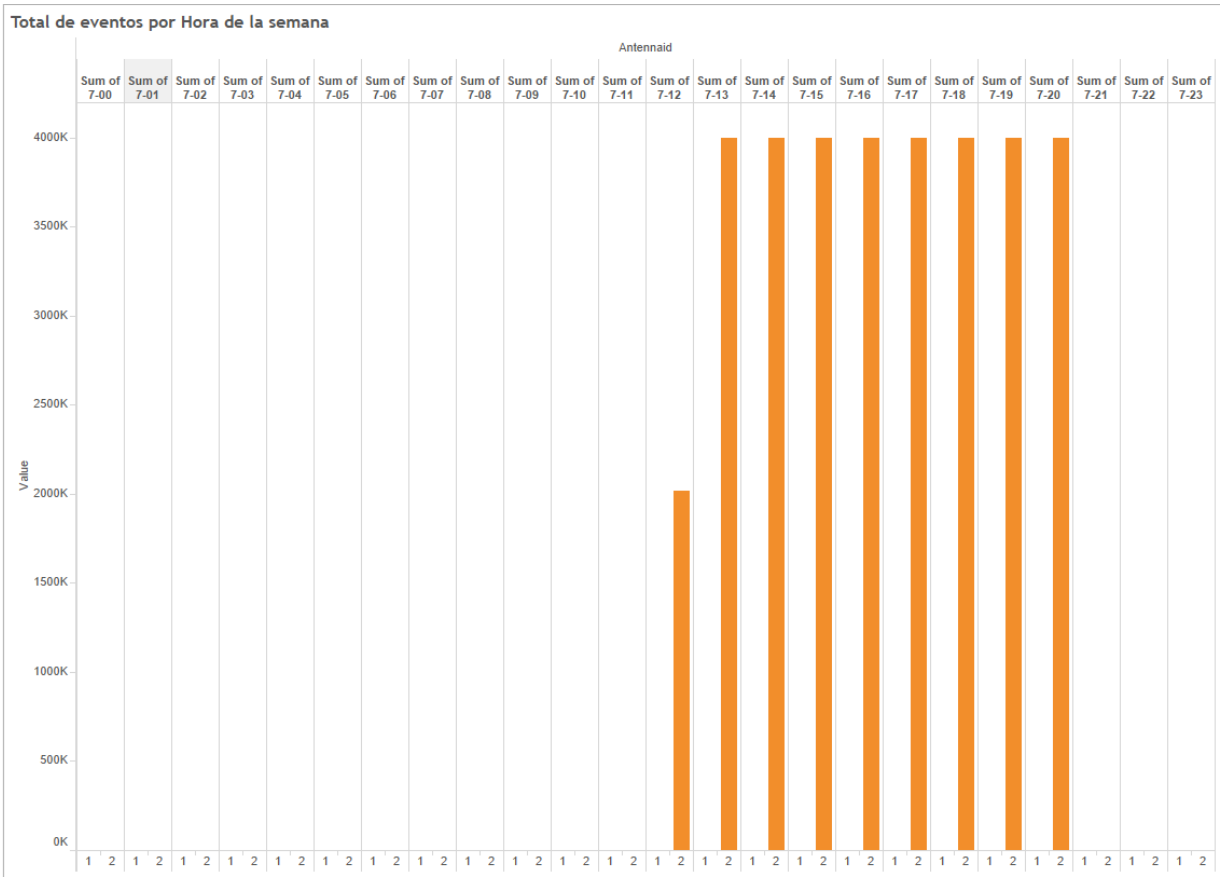
Viernes:



Sábado:

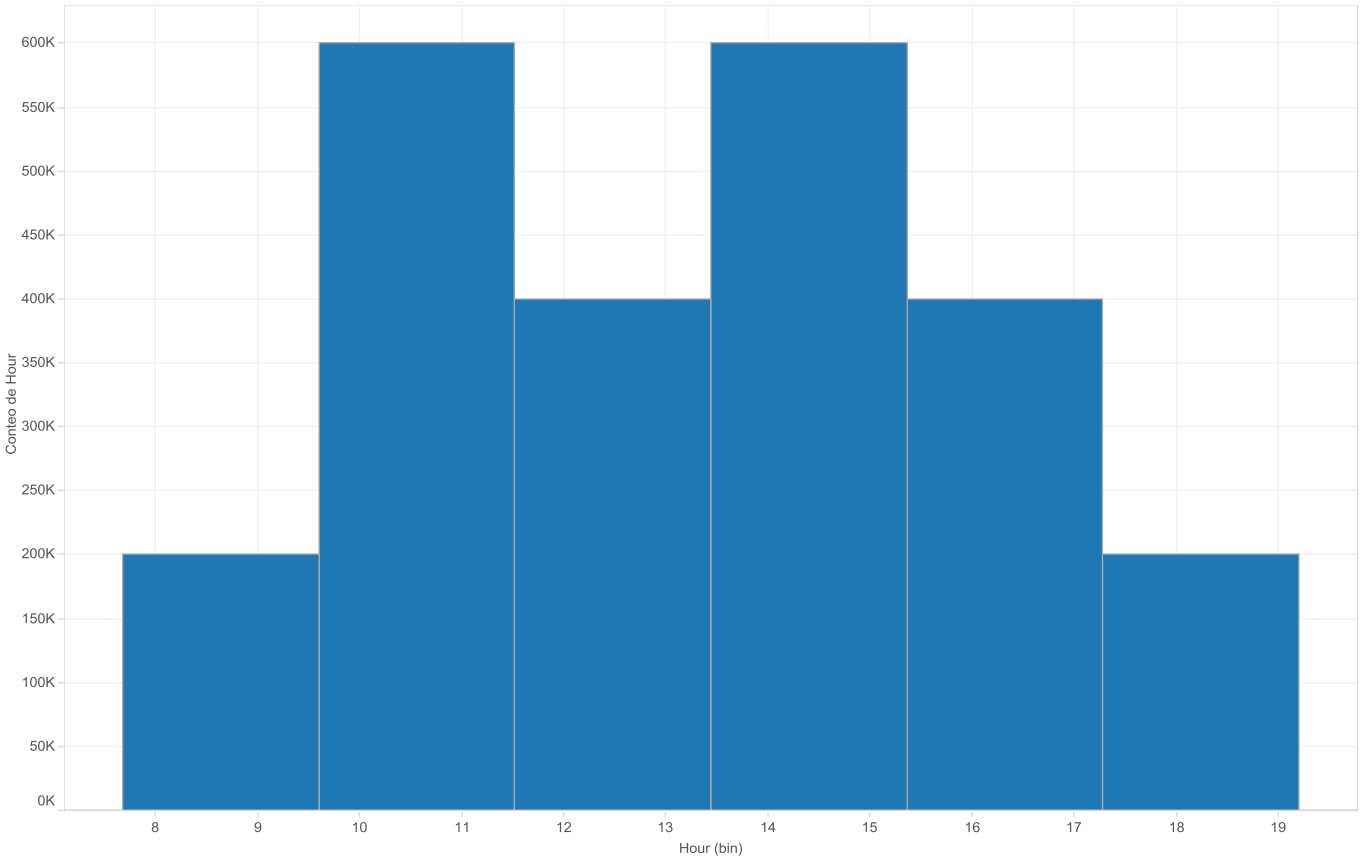


Domingo:



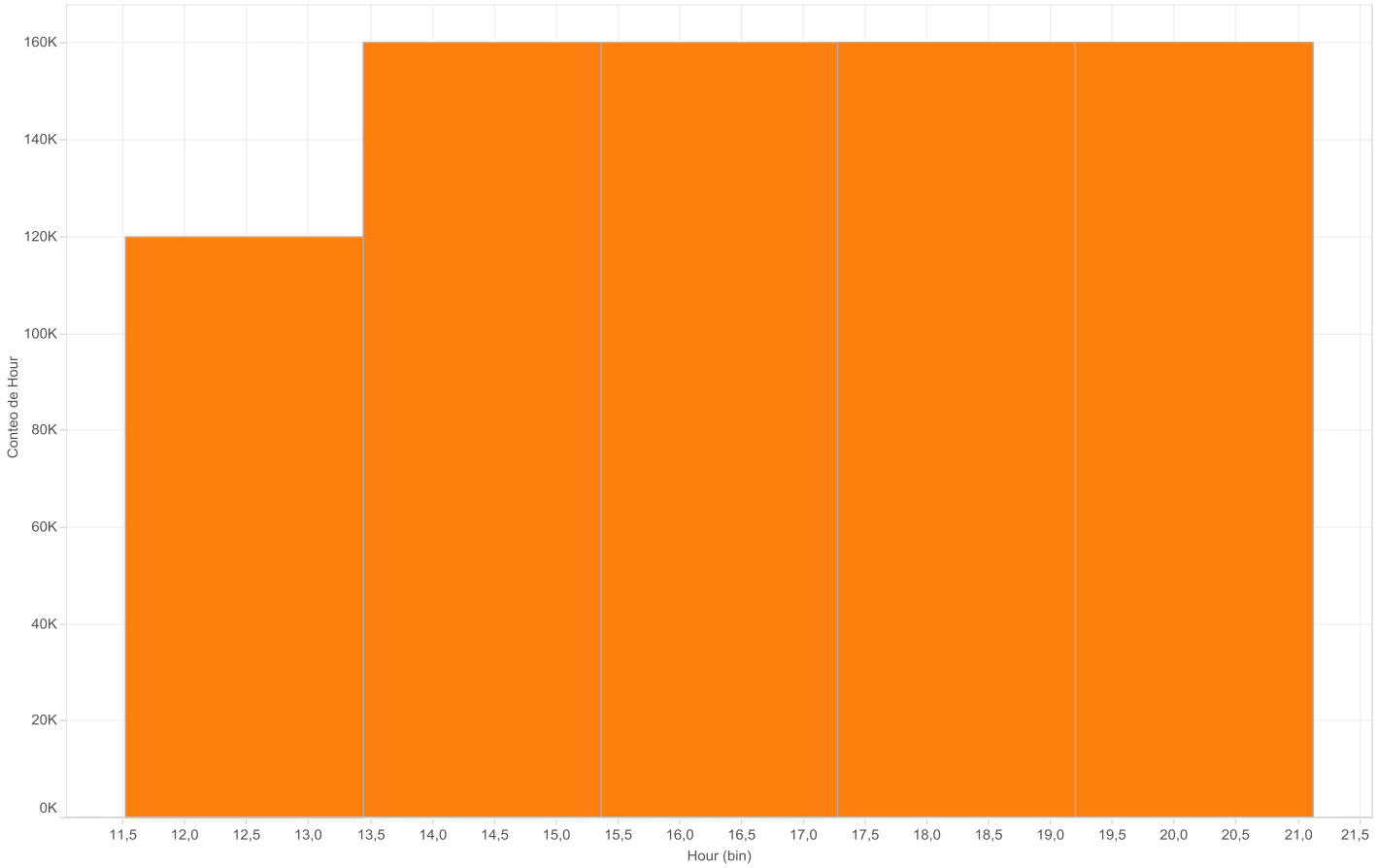
Una vez comparados los días de la semana, se observa que los días laborables tienen un patrón similar entre ellos. En los fines de semana el patrón de eventos cambia siendo prácticamente el mismo entre sábado y domingo. También se observa que de lunes a viernes hay actividad en ambas antenas, mientras que en fines de semana solo se detecta actividad en la antena A02. Esto indica que la antena A01 se trata de una antena ubicada en zona de trabajo mientras que la antena A02 se ubica en una zona residencial.

3.9.7 Histograma eventos Antena 1 (Días laborables)



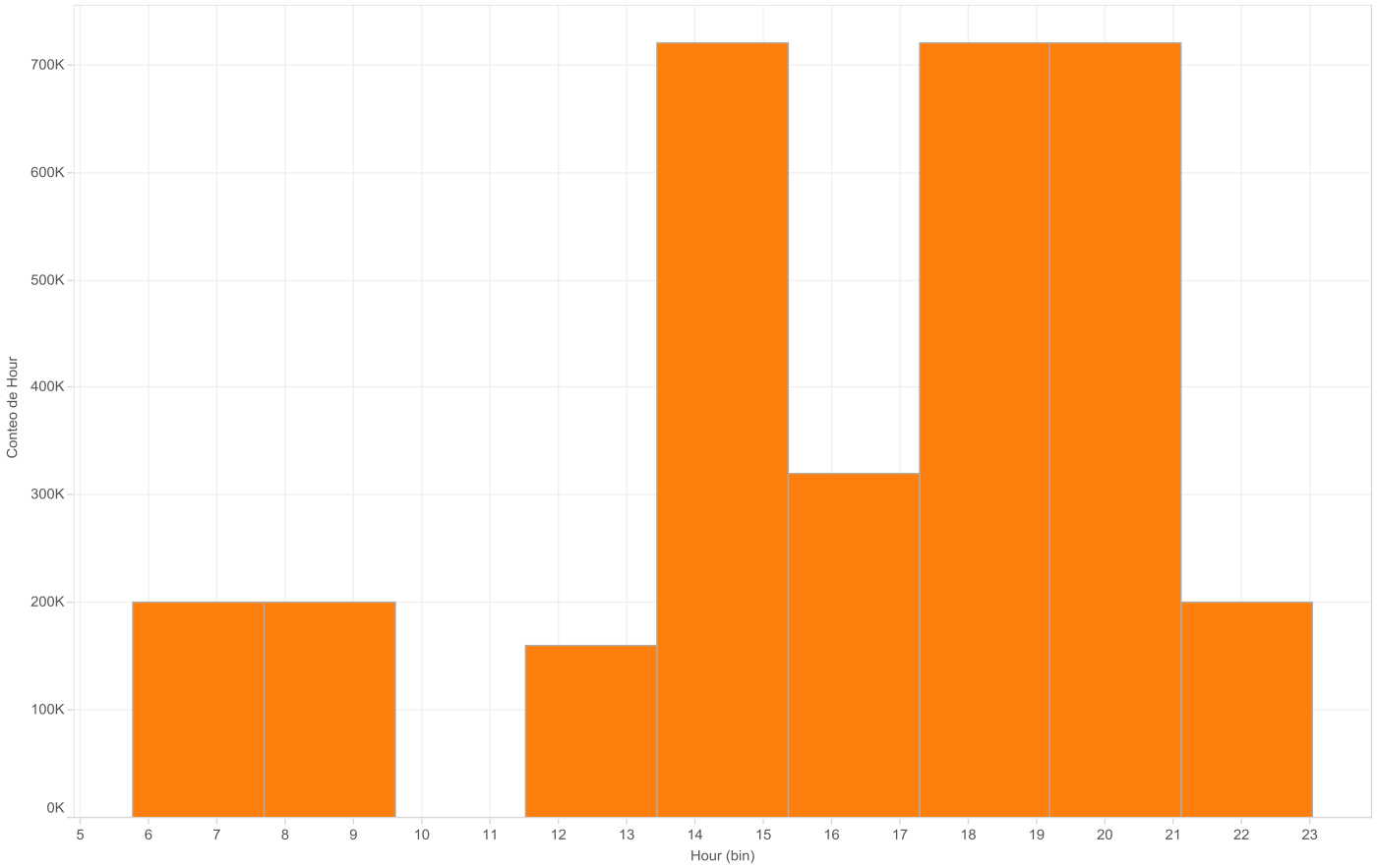
Esta gráfica muestra la distribución de los eventos por hora del día en días laborables de la antena A01, como se puede observar, la mayor concentración de actividad se presenta entre las 10h y las 17h, lo que es un claro indicio de tratarse de una antena ubicada en lugar de trabajo. A medio día se observa que hay una disminución de la actividad que podría ser explicada debido a que hay gente que se desplaza a comer a su casa.

3.9.8 Histograma Antena 2 (Fines de semana)



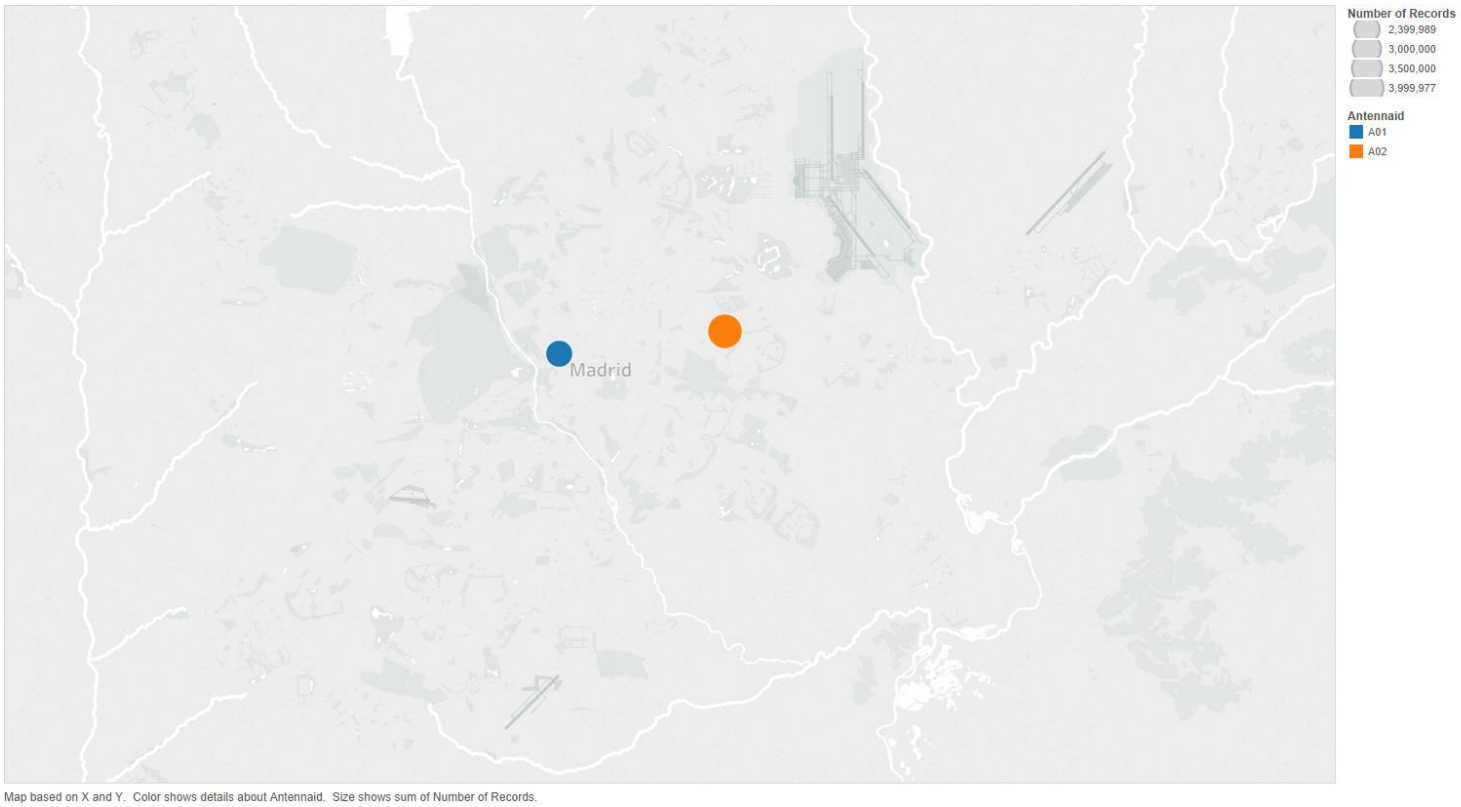
Este histograma muestra la distribución de los eventos por hora del día en días de fin de semana para la antena A02. Como se puede observar, hay actividad desde aproximadamente las 11h hasta las 21h, siendo prácticamente la misma durante todo el día, lo que podría indicar que se trata de una antena ubicada en zona residencial.

3.9.9 Histograma antena 2 (Laborables)



Esta gráfica muestra la distribución de los eventos por hora del día en días laborables de la antena A02. Al haber más actividad en las primeras y últimas horas del día, con un pico a medio día y al final de la tarde, es posible decir que se trata de una antena ubicada en una zona residencial.

3.9.10 Antenas: Ubicación y total de eventos



Como se ve en la imagen, Ambas antenas se encuentran ubicadas en la misma localidad, Madrid España. Además, al asociar el total de eventos al tamaño del punto, es posible ver claramente que la antena A02 es la que presenta más actividad. Teniendo en cuenta lo comentado anteriormente, esto es un posible indicador de que la antena A02 se trata de una antena ubicada en una zona residencial y que la entena A01 es la ubicad en una zona de trabajo.

3.10 Obtención de valor

Como indica el título del proyecto, la intención del mismo es poder llegar a obtener un valor monetario de los datos.

Para llegar a este punto ha sido necesario realizar todos los pasos que se han ido mencionado a lo largo de este documento, analizándolos en búsqueda de patrones en los datos y tratando de encontrar posibles explicaciones a estos patrones.

Ya que ahora se dispone de mucha más información, no únicamente de la hora en la que han ocurrido los eventos, es necesario usar esta información para encontrar formas de aplicar la nueva información de manera que sea posible obtener un beneficio económico.

A continuación se indican algunas posibles aplicaciones de los datos que podrían ser vendidas por nuestro cliente, ya sea como productos o como servicios y así obtener el beneficio económico esperado:

- Al saber las horas en las que hay menos afluencia de gente en cada una de las antenas, se podría hacer que las antenas estén en un modo de bajo consumo durante estas horas debido al poco uso, lo que permite ahorrar en costes de energía eléctrica.

- Crear un servicio que permita al estado saber cuál es el comportamiento de las personas en un entorno laboral. Esto permite detectar anomalías y así descubrir posibles asentamientos ilegales en zonas que no están habilitadas para vivienda. Este tipo servicios se podrían prestar al estado y así poder ayudar a reubicar en zonas seguras a gente con escasos recursos.

- Al disponer de la información de muchas antenas, es posible que la empresa de telecomunicaciones pueda detectar cuales son las antenas más y menos saturadas y así poder hacer una mejor distribución intentando aprovechar al máximo las antenas y garantizando un servicio de calidad.

- Crear un servicio enfocado al estado, ya que es posible determinar desde qué lugar de trabajo específico se desplaza más gente a lugares residenciales, con lo que es posible intentar pronosticar las vías que van a tener más tráfico y en que horario. Esto podría ayudar a determinar mejores rutas o permite al estado determinar necesidad de crear de nuevas vías de circulación o modificar las existentes.

- Determinar las zonas en donde la compañía de telecomunicaciones tiene más afluencia de gente pero con escaza cobertura, lo que permitiría mejorar el servicio prestado, he incluso ofrecer sus servicio a más personas.

- Determinar en cada zona la influencia de cada género, lo que permitiría por ejemplo, enfocar el comercio en una zona determinada en donde hay más mujeres que hombres (con una diferencia significativa). Esto podría ser un servicio que se preste a terceros que sirve para decidir en donde abrir un local.

- Enfoque de las campañas de marketing. por ejemplo si se observa que determinada zona hay un gran número de personas con un nivel económico alto, se podrían publicitar productos de lujo.

- Venta de los datos agrupados y geoposicionados a terceros. Estos se podrán aprovechar también la información obtenida con los datos y generar un beneficio económico propio aparte del generado directamente por el cliente al realizar la venta.

Capítulo 4

Gestión del Proyecto

En este Capítulo se muestran los medios empleados para la ejecución de este proyecto y la distribución de tareas.

3.11 Medios empleados

Para la realización de este proyecto se han empleado los siguientes componentes Hardware:

- Ordenador Sobremesa CPU AMD A10-7870K – 16GB RAM

Y el siguiente Software:

Software	Uso	Tipo de licencia
Apache Spark 2.3.1	Procesamiento distribuido y Machine Learning	Gratuito Código Abierto
Apache Flume 1.8.0	Ingesta de datos	Gratuito Código Abierto
Apache Hadoop 3.1.0	Almacenamiento	Gratuito Código Abierto
Tableau Desktop 2018.2.0	Visualización	Student
Linux – Ubuntu 18.04	SO para el desarrollo del SW e instalación de herramientas	Gratuito
Microsoft Office Word 2013	Documentación	Pago
Microsoft Office PowerPoint 2013	Creación Presentación y Figuras	Pago
Microsoft Office Excel 2013	Creación de Tablas	Pago
Microsoft Windows 10	Creación de la documentación	Pago
VMware Player versión 12.5.6	Sistema de Virtualización	Gratuito

Tabla 4. Software utilizado para el desarrollo del proyecto

Este último, es un software gratuito de virtualización desarrollado por VMware, Inc. Con este programa se pueden crear máquinas virtuales permitiendo la instalación de varios sistemas operativos como por ejemplo cualquier distribución de Linux o Microsoft Windows en cualquiera de sus versiones.

3.12 Definición de tareas

En esta sección se muestra la distribución y la planificación de las diferentes tareas en que está compuesto este proyecto.

Para el correcto desarrollo de este proyecto se ha realizado una planificación de cada una de las tareas que serían realizadas y posteriormente se han agrupado en diferentes fases, distinguiendo las siguientes: **análisis, diseño, implementación, pruebas y documentación**.

La primera fase de análisis consistía en buscar y revisar documentación acerca de las diferentes herramientas relacionadas con Big Data y análisis de datos, se determinaron que herramientas usar y que herramientas descartar para la realización de este trabajo de fin de máster.

Una vez terminada la fase de análisis, se ha pasado a la fase de diseño, donde se ha tenido en cuenta el diseño de la máquina virtual: instalación de las herramientas necesarias. También se determinó el entorno de desarrollo y cómo se estructuraría el proyecto en Scala.

En la fase de implementación se creó el código necesario para el tratamiento de los datos y el modelo de Machine Learning, así como también los ficheros de configuración para Apache Flume y el script SQL para la creación de las tablas en Spark SQL.

A continuación, en la fase de pruebas se comprobó el correcto funcionamiento del desarrollo realizado. Ha sido durante esta fase donde se detectó que los datos proporcionados por el "cliente" (Tutor), eran erróneos y se procedió a la corrección de los mismos.

Para terminar, en la fase de documentación se realizó este documento.

Las tareas definidas para este proyecto y su agrupación pueden observarse en la siguiente tabla (Tabla 5).

	Nombre de tarea
1	Inicio del proyecto
2	Análisis
3	Análisis Herramientas y tecnologías
4	Apache Flume
5	Apache Nifi
6	Apache Hive

7	Apache Hadoop
8	Apache Spark
9	Scala
10	Apache Parquet
11	Apache Kafka
12	D3.js
13	Log4j
14	Magellan
15	GeoSpark
16	Librería scala-point-in-polygon
17	Análisis inicial de los datos recibidos
18	Diseño
19	Creación y configuración máquina Virtual
20	Descarga, instalación y configuración de Herramientas
21	Pruebas de funcionamiento
22	Descarga, instalación y configuración de entorno desarrollo
23	Diseño de arquitectura del sistema
24	Diseño de la estructura de directorios HDFS
25	Diseño de estructura de directorios HDFS
26	Implementación
27	Configuración Flume
28	Fichero configuración Agente ciudades
29	Fichero configuración Agente clientes
30	Fichero configuración Agente antenas
31	Fichero configuración Agente eventos
32	Desarrollo
33	Fichero Configura rutas HDFS
34	Módulo de utilidades comunes
35	Módulo de Carga y limpieza de datos
36	Módulo de Carga y limpieza de datos
37	Módulo de Machine Learning (Entrenamiento)
38	Módulo de Machine Learning (Predicción)
39	Pruebas
40	Pruebas de carga
41	Pruebas de entrenamiento de modelo
42	Pruebas de predicción con modelo entrenado
43	Pruebas de conexión Tableau – Spark SQL
44	Documentación
45	Elaboración memoria
46	Fin del proyecto

Tabla 5. Relación de tareas del proyecto

Capítulo 5

Conclusiones y Líneas Futuras

Este proyecto ha abarcado la creación de un entorno virtual Big Data, teniendo en cuenta el almacenamiento, tratamiento, limpieza, análisis y representación de datos.

Para poder conseguirlo se han alcanzado las metas que se presentan a continuación:

En primer lugar fue necesaria la realización de un análisis de las diferentes herramientas Big data disponibles en el mercado. Con este análisis se consiguió obtener un listado con las herramientas que mejor se adaptaban a este caso de uso, lo que ayuda a garantizar un mejor funcionamiento del sistema.

Como segunda parte se realizó el desarrollo de una aplicación usando Spark + Scala que realizaba el tratamiento de los datos, generaba un modelo y hacía predicciones, con las que finalmente era posible diferenciar si la ubicación de una antena hace referencia a lugares de trabajo o a zonas residenciales.

Una vez alcanzadas estas metas se consiguió representar los datos mediante el uso de una herramienta de visualización. Lo que ayuda a entender mejor los datos y a obtener posibles conclusiones sobre los datos que, quizá, sin este tipo de herramientas no hubiese sido posible conseguir.

Primeramente se concluye que para la implementación de este tipo de sistemas es necesario realizar un análisis previo con el fin de poder identificar las tecnologías que mejor se adaptan a las necesidades. También es muy importante que los datos no se vean alterados, ya que como se ha visto, un error puede llegar a ocasionar conclusiones diferentes y por tanto llevar al usuario de este sistema a tomar decisiones que no se corresponden con la realidad de los datos.

Finalmente este sistema en su conjunto es una herramienta que ayuda a la toma decisiones, además de que con la realización de análisis exhaustivos de los resultados y buscando la manera de aplicarlos, es posible obtener un retorno monetario.

Es muy importante tener en cuenta que para poder tomar las decisiones finales a partir de los datos, se requiere de la intervención de expertos, quienes son los encargados de la identificación de los patrones

y obtención de conclusiones sobre los datos, lo que ocasiona que el proceso no sea 100% automático, debido a la necesidad de participación de una persona.

Se puede considerar los siguientes puntos como posibles líneas futuras para mejorar y complementar el proyecto:

- Automatización de la ejecución de los diferentes componentes (Flume, aplicación) mediante herramientas de planificación como se explicó en el apartado "3.5.2. Lanzamiento de procesos".

- Aumento del número de antenas y de clientes con el fin de poder realizar análisis que lleven a identificar otros tipos de actividades, por ejemplo, ocio.

- Realizar pruebas de rendimiento con conjuntos de datos mucho más grandes.

- Configuración en un entorno con más nodos para poder realizar replicación de los datos y procesar de manera realmente distribuida.

- Buscar nueva información de los datos teniendo en cuenta otros parámetros (como la edad, sexo, etc.) de los usuarios.

- Incluir seguridad a los datos (Kerberos, por ejemplo).

- Implementación del sistema en servicios en la Nube (AWS, GCP etc.)

Si se uniesen todas estas mejoras se dispondría de un sistema mucho más completo y robusto que permitirá tener una visión más global y completa de lo que son este tipo de sistemas y los retos que supone su implementación.

Glosario

WCSS	Within-Cluster Sums of Squares
ASCII	American Standard Code for Information Interchange
HDFS	Hadoop Distributed File System
ML	Machine Learning
Viz	Visualización
DF	Spark DataFrame
SW	Software
SO	Sistema Operativo
API	Application Programming Interface

Referencias

[1] Kanungo, T.; Mount, D. M.; Piatko, C. D.; Silverman, R.; Wu, A. Y. (2002). *«An efficient k-means clustering algorithm: Analysis and implementation».* Disponible [Internet]: <http://www.cs.umd.edu/~mount/Papers/pami02.pdf> [27 de Septiembre de 2018]

[2] *K-means.* Disponible [Internet]: <https://es.wikipedia.org/wiki/K-means> [27 de Septiembre de 2018]

[3] *Apache flume.* Disponible [Internet]: <https://flume.Apache.org/> [11 de Septiembre de 2018]

[4] *Apache Hadoop*. Disponible [Internet]: <http://hadoop.Apache.org/> [11 de Septiembre de 2018]

[5] *K-means*. Disponible [Internet]:<https://es.wikipedia.org/wiki/K-means> [11 de Septiembre de 2018]

[6] *Apache Nifi*. Disponible [Internet]: <https://en.wikipedia.org/wiki/Apache_NiFi> [19 de Septiembre de 2018]

[7] *Apache Spark*. Disponible [Internet]:

< http://spark.Apache.org/ > [19 de Septiembre de 2018]

[8] *Parquet*. Disponible [Internet]: <http://parquet.Apache.org/> [19 de Septiembre de 2018]

[9] *Tableau Desktop*. Disponible [Internet]: <https://www.tableau.com/es-es/products/desktop> [19 de Septiembre de 2018]

[10] *Apache Hive*. Disponible [Internet]: <https://hive.Apache.org/ > [19 de Septiembre de 2018]

[11] *Apache Kafka*. Disponible [Internet]: <https://kafka.Apache.org/> [19 de Septiembre de 2018]

[12] D3 Data Driven Documents. Disponible [Internet]: <https://d3js.org/> [19 de Septiembre de 2018]

[13] The Scala Programming Language. Disponible [Internet]: <https://d3js.org/> [19 de Septiembre de 2018]

[14] Apache Flink. Disponible [Internet]: <https://flink.Apache.org/> [25 de Septiembre de 2018]

[15] Why Apache Flink. Disponible [Internet]: <https://data-artisans.com/why-Apache-flink > [25 de Septiembre de 2018]

[16] Magellan: Geospatial Analytics Using Spark: <https://github.com/harsha2010/magellan> [26 de Septiembre de 2018]

[17] GeoSpark [Internet]: <http://datasystemslab.github.io/GeoSpark/> [26 de Septiembre de 2018]

Anexo A

Fichero de configuración log4j (log4j.properties):

```
# Define the root logger with appender file
log = /home/acp/Documents/TFM/logs
log4j.rootLogger = INFO, FILE

# Define the file appender
log4j.appender.FILE=org.Apache.log4j.RollingFileAppender
log4j.appender.FILE.File=${log}/antennas.log
log4j.appender.FILE.Append=true
log4j.appender.FILE.MaxFileSize=10MB
log4j.appender.FILE.MaxBackupIndex=12

# Define the layout for file appender
log4j.appender.FILE.layout=org.Apache.log4j.PatternLayout
log4j.appender.FILE.layout.conversionPattern=%d %d{Z} [%t] %-5p (%F:%L)
- %m%n
```

Anexo B

Fichero de configuración aplicación (properties.conf):

```
hdfs {
  url = "hdfs://localhost:9000"

  input {
    citiesPath = "/input/cities/"
    clientsPath = "/input/clients/"
    antennasPath = "/input/antennas/"
    eventsPath = "/input/events/"

    cities = ${hdfs.url}${hdfs.input.citiesPath}
    clients = ${hdfs.url}${hdfs.input.clientsPath}
    antennas = ${hdfs.url}${hdfs.input.antennasPath}
    events = ${hdfs.url}${hdfs.input.eventsPath}

    old {
      citiesPath = "/input/citiesOld/"
      clientsPath = "/input/clientsOld/"
      antennasPath = "/input/antennasOld/"
      eventsPath = "/input/eventsOld/"

      cities = ${hdfs.url}${hdfs.input.old.citiesPath}
      clients = ${hdfs.url}${hdfs.input.old.clientsPath}
      antennas = ${hdfs.url}${hdfs.input.old.antennasPath}
      events = ${hdfs.url}${hdfs.input.old.eventsPath}
    }
  }
  cleanData {
    citiesPath = "/cleanData/cities/"
    clientsPath = "/cleanData/clients/"
    antennasPath = "/cleanData/antennas/"
    eventsPath = "/cleanData/events/"

    cities = ${hdfs.url}${hdfs.cleanData.citiesPath}
    clients = ${hdfs.url}${hdfs.cleanData.clientsPath}
    antennas = ${hdfs.url}${hdfs.cleanData.antennasPath}
    events = ${hdfs.url}${hdfs.cleanData.eventsPath}
  }

  modeldata {

    modelPath = "/modeldata/modelTrain/"
    pipelinePath = "/modeldata/pipeline"
    predictionsPath = "/modeldata/predictions/"
    dataPath = "/modeldata/data/"

    model = ${hdfs.url}${hdfs.modeldata.modelPath}
    pipeline = ${hdfs.url}${hdfs.modeldata.pipelinePath}
    predictions = ${hdfs.url}${hdfs.modeldata.predictionsPath}
    data = ${hdfs.url}${hdfs.modeldata.dataPath}
  }
}
```