# Análisis de datos y modelos de aprendizaje para monitorizar el consumo de agua en redes de abastecimiento usando tecnologías Big Data.


**Trabajo de Fin de Máster**

**Gloria Ayde Leguizamón Rojas**

**Máster de Ingeniería Informática - Especialidad en Big Data**
**Escuela de Ingeniería Informática**
**Universidad de Valladolid**

**Julio 2018**

# Análisis de datos y modelos de aprendizaje para monitorizar el consumo de agua en redes de abastecimiento usando tecnologías Big Data.

*TFM para la obtención del título de Máster de Ingeniería Informática*

**Gloria Ayde Leguizamón Rojas**

*Tutores*

**Dr. D. Aníbal Bregón Bregón**

**Dr. D. Miguel Ángel Martínez Prieto**

**Máster de Ingeniería Informática - Especialidad en Big Data**

**Escuela de Ingeniería Informática**

**Universidad de Valladolid**

**Julio 2018**

*Un hombre sólo aprende de dos formas;*
*una es leyendo y la otra asociándose*
*con gente más sabia.*
*-Will Rogers.*

# Agradecimientos

*"Mientras el río corra, los montes hagan sombra*
*y en el cielo haya estrellas, debe durar la*
*memoria del beneficio recibido,*
*en la mente del hombre agradecido".*

Publio Virgilio Marón

En primer lugar agradezco a la Universidad de Valladolid por haber aceptado hacer parte de ella y abrirme sus puertas para continuar mi formación y actualizar mis conocimientos. Así como también a su cuerpo docente que me brindaron sus conocimientos y apoyo durante estos años.

Un sincero agradecimiento a mis tutores Anibal Bregon y Miguel Ángel Martínez por aceptarme para realizar este TFM. Su apoyo y confianza en mi trabajo y su capacidad para guiarme ha sido un aporte invaluable, no solamente en el desarrollo de este trabajo, sino también en mi formación. Les agradezco también el haberme facilitado siempre los medios suficientes para llevar a cabo todas las actividades propuestas durante el desarrollo de este trabajo.

A mi esposo Rubén, que con todo su amor y paciencia me ha apoyado incondicionalmente para alcanzar mis metas y específicamente lograr este objetivo. Quien me ha acompañado en todo el proceso, con sus palabras de aliento, motivación y sus consejos siempre oportunos. Quien me da ejemplo de esfuerzo y dedicación con la realización de su tesis doctoral y de quien estoy muy orgullosa.

A mi hijo Andrés Felipe, quien ha sido siempre mi motivación para ser mejor y quien ha sabido comprender mi ausencia en estos años, pero que ha estado conmigo de corazón y de pensamiento.

A mi familia (incluido Alejandro) quienes siempre han incentivado en mí, el valor de la educación, el aprendizaje y la constante búsqueda del conocimiento. Además de inculcarme valores como el esfuerzo, perseverancia y luchar por lo que quieres. Y que a pesar de la distancia, siento su apoyo todos los días.

# Resumen

*Lo importante en la ciencia no es tanto obtener nuevos datos, sino descubrir nuevas formas de pensar sobre ellos.*

William Lawrence Bragg

Todo sistema de abastecimiento de agua está integrado por una estructura compleja, en la que se llevan a cabo procesos de producción, transporte y distribución de agua, entre otros. Dichos procesos, tienen como fin satisfacer la demanda de los consumidores en tiempo real. Ello implica proporcionar continuamente a los usuarios agua de calidad en volúmenes adecuados a una presión razonable, asegurando así una distribución de agua confiable. Por tanto, el diseño de planes de mantenimiento y gestión, tanto para las condiciones actuales como las futuras demandas que afectan las presiones en la red de tuberías -en ciudades cada vez más diversas y aglomeradas-, así como la reducción de costes para las empresas, representa una parte capital dentro de la estrategia de gestión de toda red de suministro.

Para enfrentar esta necesidad, la disponibilidad de datos es imprescindible. Así, las empresas abastecedoras de agua han implementado variadas estrategias para la recolección de los mismos casi en tiempo real, donde estos son obtenidos directamente desde los sensores. Antes de esta sustancial mejora, las lecturas disponibles solían ser mensuales o bimensuales y se recopilaban in situ, lo que dificultaba el análisis adecuado del comportamiento del cliente y la evaluación para ampliar el estado de la infraestructura y mejorar su calidad.

Así, en este nuevo escenario, si bien el conocimiento de los datos mejora la capacidad de anticipación a los problemas, la acumulación de toda esa información, supera los límites de cualquier herramienta de análisis estadístico convencional. Por ello se abre, no sólo la posibilidad, sino la conveniencia, de la implementación de tecnologías Big Data.

En este trabajo se plantea el procesamiento, análisis y la creación de modelos de aprendizaje automático para monitorizar la demanda en redes de agua utilizando datos recopilados de los sensores de caudal y presión, haciendo uso del paradigma del descubrimiento de conocimiento en base de datos (KDD), junto con el proceso extracción, carga y transformación (ELT) para la construcción del Data Lake, en un entorno Big Data.

El proceso se inicia con la extracción y carga de los datos fuente, que hace parte de la construcción del Data Lake. Luego se realiza el análisis de los datos en bruto con el modelado conceptual, donde se construye el diccionario de datos y se diseña el modelo entidad relación y el mapa lógico de datos que describe las características que relacionan los datos en bruto y los datos refinados. Así, se realizan todas las transformaciones necesarias, hasta obtener el archivo que se usará en el modelo de aprendizaje y cuya estructura fue definida en el mapa lógico.

En la etapa de aprendizaje, se usan 3 métodos de regresión disponibles en la biblioteca ML de Spark, Decision Tree, Random Forest y Gradient-boosted tree y se evalúan en los conjuntos de prueba con los parámetros óptimos obtenidos con validación cruzada.

Finalmente se visualizan los resultados de las predicciones de los modelos entrenados en la etapa de aprendizaje, usando la herramienta *Qlik*, donde se muestra la comparación de los 3 modelos entrenados y la contrastación empírica de las hipótesis propuestas.

# Abstract

*The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.*

William Lawrence Bragg

Every water supply system is integrated by a complex structure, in which water production, transport and distribution processes are carried out, among others. These processes are intended to satisfy consumer demand in real time. This means providing users with quality water continuously at adequate volumes at a reasonable pressure, thus ensuring a reliable water distribution. Therefore, the design of maintenance and management plans, both for current conditions and future demands that affect pressures in the pipeline network -in cities increasingly diverse and agglomerated-, as well as reducing costs for companies , represents a key part of the management strategy of any supply network.

To face this need, the availability of data is essential. Thus, the water supply companies have implemented various strategies for collecting them almost in real time, where they are obtained directly from the sensors. Before this substantial improvement, the available readings were usually monthly or bimonthly and were collected in situ, which made it difficult to adequately analyze the client's behavior and the evaluation to expand the state of the infrastructure and improve its quality.

Thus, in this new scenario, although the knowledge of the data improves the ability to anticipate problems, the accumulation of all this information exceeds the limits of any conventional statistical analysis tool. That is why it opens up, not only the possibility, but the convenience, of the implementation of Big Data technologies.

In this work the processing, analysis and the creation of automatic learning models to monitor the demand in water networks using data collected from the flow and pressure sensors, making use of the paradigm of knowledge discovery in database (KDD) is proposed, together with the extraction, load and transformation process (ELT) for the construction of the Data Lake, in a Big Data environment.

The process begins with the extraction and load of the source data, which is part of the construction of the Data Lake. Then the analysis of the raw data is carried out with conceptual modeling, where the data dictionary is constructed and the relationship entity model and the logical data map that describes the characteristics that relate the raw data and the refined data are designed. Thus, all the necessary transformations are carried out, until the file that will be used in the learning model is obtained and whose structure was defined in the logical map.

In the learning stage, 3 regression methods available in the Spark ML library are used:

Decision Tree, Random Forest and Gradient-boosted tree for are evaluated in the test sets with the optimal parameters obtained with cross-validation.

Finally, the results of the predictions of the models trained in the learning stage are visualized, using the Qlik tool, which shows the comparison of the 3 trained models and the empirical testing of the proposed hypotheses.

# Índice

**Agradecimientos** vii

**Resumen** ix

**Abstract** xi

**1. Introducción** 1
- 1.1. Introducción . . . 1
- 1.2. Motivación . . . 2
- 1.3. Objetivo Principal . . . 3
- 1.4. Estructura de capítulos . . . 4

**2. Marco teórico y Estado del arte** 5
- 2.1. Redes de distribución de agua . . . 5
  - 2.1.1. Distrito Hidrométrico . . . 6
  - 2.1.2. Agua no registrada . . . 6
  - 2.1.3. Fugas . . . 7
- 2.2. Técnicas de aprendizaje para series temporales . . . 7
  - 2.2.1. Árbol de Decisión . . . 7
  - 2.2.2. Random Forest . . . 9
  - 2.2.3. Gradient Boosted Tree . . . 9
- 2.3. Aproximaciones de Big Data para WDN . . . 11

**3. Metodología** 15
- 3.1. Descripción del problema . . . 15
- 3.2. Metodología . . . 16
- 3.3. Data Lake . . . 18
  - 3.3.1. Ventajas . . . 19
  - 3.3.2. Contraste con EDW . . . 20
- 3.4. Arquitectura del Data Lake propuesto . . . 21

**4. Construcción del Data Lake** 25
- 4.1. Arquitectura de datos . . . 25
  - 4.1.1. Fuentes de datos . . . 25
  - 4.1.2. Modelo conceptual . . . 26
  - 4.1.3. Datos refinados . . . 27
- 4.2. Ingesta de Datos . . . 28

4.3. Almacenamiento . . . . . 29
4.4. Preprocesamiento y Transformación . . . . . 30
4.4.1. Limpieza: interpolación . . . . . 31
4.4.2. Transformación 1: extraer fecha . . . . . 33
4.4.3. Transformación 2: incluir weekday . . . . . 33
4.4.4. Transformación 3: incluir holiday . . . . . 34
4.4.5. Transformación 4: unión de sensores . . . . . 35
4.4.6. Salida: de tabla a archivo plano . . . . . 36

**5. Modelos de Aprendizaje** **39**
5.1. Introducción . . . . . 39
5.2. Exploración dataset refinado . . . . . 40
5.3. Selección del modelo . . . . . 42
5.4. Evaluación del modelo . . . . . 48

**6. Visualización** **53**
6.1. Introducción . . . . . 53
6.2. Predicciones . . . . . 53
6.2.1. Predicción días laborales . . . . . 54
6.2.2. Predicción fin de semana . . . . . 56
6.2.3. Predicción fechas especiales . . . . . 57

**7. Conclusiones y trabajo futuro** **61**
7.1. Trabajo futuro . . . . . 62

**Bibliografía** **64**

**A. Flume** **69**

**B. Oozie** **71**

**C. Breeze** **73**

**D. Creación de Tablas Hive** **75**

# Índice de figuras


2.1. Ciclo integral del agua [24] . . . . . 6
2.2. Representación de un árbol de decisión [8] . . . . . 8
2.3. Representación de un Random Forest de 2 árboles [8] . . . . . 9
2.4. Representación de un Gradient Boosted Tree [12] . . . . . 10

3.1. Metodología KDD + ELT (Fuente: elaboración propia) . . . . . 17
3.2. Representación de un Data Lake [5] . . . . . 19
3.3. Arquitectura del Data Lake. (Fuente: elaboración propia) . . . . . 22

4.1. Entidad de los datos refinados . . . . . 27
4.2. Arquitectura del agente flume. (Fuente: elaboración propia) . . . . . 28
4.3. Esquema de Almacenamiento HDFS . . . . . 29
4.4. Esquema del preprocesamiento de datos. (Fuente: elaboración propia) . . . . 31
4.5. Representación del bucle para el proceso de limpieza de series. (Fuente: elaboración propia) . . . . . 31
4.6. Gráfica del vector resultante de la interpolación lineal para el día 2015-02-03. 32

5.1. Tendencia de consumo por meses y por estaciones . . . . . 40
5.2. Relación de consumo con presión . . . . . 41
5.3. Tendencia de consumo en días laborales vs festivos . . . . . 41
5.4. Tendencia de consumo semanal . . . . . 42
5.5. Representación de la validación cruzada (Wikipedia) . . . . . 42
5.6. Gráfico del coeficiente de determinación para Decision Tree . . . . . 44
5.7. Gráfico del coeficiente de determinación para Random Forest . . . . . 45
5.8. Gráfico del coeficiente de determinación para GBT . . . . . 47
5.9. MSE diario del conjunto de datos test por método . . . . . 49
5.10. Predicción para el día 11/11/2015 por el método DT . . . . . 49
5.11. Predicción para el día 16/07/2015 por el método DT . . . . . 50
5.12. Predicción para el día 11/11/2015 por el método RF . . . . . 50
5.13. Predicción para el día 07/08/2015 por el método RF . . . . . 50
5.14. Predicción para el día 25/11/2015 por el método GBT . . . . . 51
5.15. Predicción para el día 07/08/2015 por el método GBT . . . . . 51

6.1. Comparación de resultados de los 3 métodos de regresión- Promedio anual . 54
6.2. Comparación de resultados para días laborales de abril . . . . . 54
6.3. Comparación de resultados para días laborales de octubre . . . . . 55
6.4. Comparación de resultados para días laborales de junio . . . . . 55

6.5. Comparación de resultados en fin de semana de mayo. . . . . 56
6.6. Comparación de resultados en fin de semana de invierno. . . . . 57
6.7. Comparación de resultados en fin de semana de verano. . . . . 57
6.8. Comparación de resultados en navidad . . . . . 58
6.9. Comparación de resultados en Pascua . . . . . 58
6.10. Comparación de resultados en año nuevo . . . . . 59

# Índice de Tablas


4.1. Detalles de los datos de consumo . . . . . 26
4.2. Detalles de los datos de presión . . . . . 26
4.3. Información de días festivos de 2015 . . . . . 26
4.4. Mapa lógico de datos que relaciona los datos en bruto con los datos refinados 27
4.5. Estructura de la tabla Hive para la transformación 1 . . . . . 33
4.6. Estructura de la tabla Hive para la transformación 2 . . . . . 34
4.7. Estructura de la tabla Hive para la transformación 3 . . . . . 35
4.8. Estructura de la tabla Hive para la transformación 4 . . . . . 36

5.1. Rangos de cada atributo . . . . . 40
5.2. Resultados del modelo Decision Tree . . . . . 43
5.3. Resultados del modelo Random Forest . . . . . 44
5.4. Resultados del modelo Gradient Boost Tree . . . . . 46
5.5. Resultados de parámetros óptimos obtenidos con validación cruzada. . . . . 48
5.6. Resultados de parámetros óptimos vs. tiempo de entrenamiento . . . . . 48
5.7. Resultado de evaluación de modelos . . . . . 48
5.8. Error mínimo, máximo y medio de los métodos en el conjunto test . . . . . 49
5.9. Meses con mejor y peor predicción . . . . . 52

# Capítulo 1

# Introducción

*Comenzar bien no es poco,*
*pero tampoco es mucho.*

Sócrates.

**Resumen:** Este capítulo presenta una breve introducción al trabajo, donde se podrá hacer una idea general del contexto que enmarca el proyecto, la motivación y los objetivos que esperamos alcanzar. También se encuentra aquí una descripción del resto de capítulos del documento

## 1.1. Introducción

EL valor del agua, como recurso fundamental y escaso, es hoy aceptado por todos los organismos internacionales, siendo un elemento crítico en la planificación del futuro de la humanidad [24]. Concretamente, asegurar la optimización del uso de tan preciado recurso, implica poner el foco de atención en aspectos tales como el suministro y consumo, así como el gasto energético y tratamiento químico que la hace útil para los diferentes usos requeridos de ella, tanto humanos como industriales.

Dentro de este proceso de optimización, la vigilancia del correcto funcionamiento de las redes de abastecimiento y distribución de agua -evitando fugas y asegurando el caudal- cobra una importancia capital [44] [45] [46]. Las pérdidas de agua que se producen durante el recorrido del fluido, desde los elementos reguladores situados en cabecera de red hasta los consumidores finales y el envejecimiento de las redes de agua desafían el control de las infraestructuras de distribución de agua. También el aumento de la demanda, causado por el crecimiento de la población y la constante edificación. Por tanto, uno de los principales desafíos que deberán afrontar los gestores de estas redes, será la capacidad de detectar anomalías causadas por ráfagas, fugas, pérdida de agua, actividad no contabilizada u otros factores. Esto "presenta una dificultad añadida cuando se desea detectar estas anomalías en tiempo real"[1].

Disponer de sistemas que permitan el seguimiento de la red de agua, no sólo ayuda a garantizar la distribución adecuada al usuario final, sino que también fomenta la sostenibilidad al reducir las pérdidas en las diferentes etapas de bombeo y distribución, además de disminuir el gasto energético y económico en las empresas abastecedoras.

Los importantes avances tecnológicos de las dos últimas décadas en el campo de las tecnologías de la información, han originado un cambio de paradigma en la gestión y la explotación de los sistemas urbanos de abastecimiento de agua por parte de las empresas gestoras. Las tradicionales políticas de gestión pública, han evolucionado hacia nuevos modelos más eficientes de gestión mixta o privada, con el objeto de mejorar no solo la calidad del servicio que se ofrece a los usuarios, sino también optimizar los recursos reduciendo el coste económico y ocasionando el menor impacto ambiental posible [32].

Actualmente, gracias a los datos recogidos por sensores y a la mejora en las comunicaciones, se dispone de un gran volumen de información. Por tanto, el reto que ha de afrontarse, consiste en cruzar esos datos y traducirlos en información relevante que permita un análisis descriptivo de lo que está ocurriendo en la red, obteniendo el resultado que facilite un análisis predictivo, que simule lo que ocurrirá, sobre la base de la integración de toda esta información junto con las herramientas que se disponen.

En este trabajo se plantea el procesamiento, análisis y la creación de modelos de aprendizaje automático para monitorizar la demanda en redes de agua, utilizando datos recopilados de los sensores de caudal y presión de un distrito hidrométrico. En la metodología usada se toma como referencia el paradigma del descubrimiento de conocimiento en base de datos (KDD), junto con el proceso extracción, carga y transformación (ELT) para la construcción del Data Lake, en un entorno Big Data.

## 1.2. Motivación

La gestión del sistema integral de abastecimiento, constituye uno de los principales desafíos en la gestión pública en todos los niveles de poder territorial. Así, las autoridades y los operadores de agua enfrentan retos a diario, como preservar el suministro de agua, controlar el volumen transportado por las redes, detectar fugas, reducir el porcentaje de agua no registrada (ANR), etc. Por lo tanto, al generar y proporcionar una predicción de consumo con un porcentaje de precisión alto, se logrará:

- Mejorar la operación de la gestión diaria.
- Programar el bombeo de forma más óptima.
- Controlar los niveles de almacenamiento en los tanques.
- Alertar rápidamente de fugas repentinas.

En general un uso eficiente del modelo hidráulico de la red, lo cual se refleja en el ahorro de consumo y costes energéticos para las empresas abastecedoras de agua y las administraciones públicas competentes en la gestión de este recurso.

Asimismo, la presencia de fugas obliga a producir y distribuir más agua de la necesaria, y a aumentar la presión del sistema para garantizar que el producto llegue al consumidor, lo que genera, paradógicamente, una presión más alta en el sistema que agrava las fugas. Ello se traduce en un mayor desperdicio de agua potable y, por consiguiente, de energía. Reducir las pérdidas de agua, es una forma directa de disminuir el consumo eléctrico. Así, para lograr dicha reducción de perdidas de agua, el uso de la predicción de demanda, jugará un papel muy importante, ya que en el momento en que el consumo de agua supere umbrales definidos de normalidad de acuerdo a la predicción, se podrá sospechar de la ocurrencia de eventos y la detección anomalías de forma inmediata, que activarán las alertas de fugas de agua.

La idea que acaba de plantearse, se refrenda con el siguiente dato. “El porcentaje relativamente bajo de consumo de energía por parte de los operadores en comparación con el consumo energético de los usuarios, indica que el mayor potencial de ahorro viene dado por la disminución del consumo de agua en los usos de mayor intensidad energética, que en el caso residencial (el 73 % del consumo de agua total para usos urbanos, siendo el 11 % y el 16 % restantes lo que consumen industria-comercios y otros usos urbanos, respectivamente) corresponden al agua caliente para higiene (ducha, baño y lavado de manos) y electrodomésticos (lavadoras y lavavajillas). En línea con la idea de redes inteligentes, la instalación de contadores inteligentes permitiría desagregar la información de consumo por elemento de uso final (ducha, inodoro, lavadora, etc.) y tener una radiografía completa de la demanda: cuánto (volumen consumido), dónde (información georreferenciada) y cuándo (franjas horarias y estacionales punta y valle). Por este motivo la gestión de la demanda (y la introducción de incentivos adecuados) es también clave en lo que respecta al impacto global del ciclo urbano del agua”[43].

## 1.3. Objetivo Principal

El objetivo principal de este trabajo es, a partir de los sensores de caudal y presión de la red de distribución de agua (WDN), extraer conocimiento útil sobre las variables asociadas al consumo de los usuarios y su comportamiento al consumir agua. Para ello se hará uso del paradigma KDD, junto con el proceso ELT, y aprendizaje automático.

En el proceso de extracción del conocimiento, se contrastará o validará algunas de las hipótesis que podrían influir en la demanda de agua en la ciudad:

- **Tendencia horaria**: debe haber una variación en la demanda de acuerdo a los horarios de oficina. Temprano en la mañana aumentará y tarde en la noche puede tener una tendencia diferente y una baja demanda en horario nocturno de 10:00 p.m. a 4:00 a.m.
- **Tendencia diaria**: se demanda más agua en días festivos y fines de semana en comparación los días laborales, aunque también depende del sector (residencial / comercial).
- **Tendencia estacional**: la demanda de agua será mayor en días calurosos en comparación con días fríos.

Para realizar este objetivo principal, vamos a dividirlo en varios subojetivos específicos:

1. Construir un Data Lake que sea capaz de gestionar la ingesta, transformación, almacenamiento, acceso e integración de datos, de modo que pueda usarse para propósitos de análisis y descubrimiento de información, destinada a empresas abastecedoras de agua o que gestionen los procesos del ciclo integral del agua.
2. Explorar las distintas aproximaciones disponibles para inferir modelos de predicción a partir de los datos, así como un estudio práctico de las que mejor se ajusten al problema particular que se quiere solventar.
3. Usar la visualización para los dos propósitos requeridos en este trabajo, “explorar los datos” obteniendo información que es desconocida a priori y “explicar los datos” para enseñar los resultados obtenidos. Teniendo en cuenta los métodos de presentación y representación de los datos, como también considerando los principios de diseño.

## 1.4. Estructura de capítulos

El documento está estructurado en los siguientes capítulos:

- El Capítulo 2 presenta el marco teórico y el estado del arte. En el se describen algunos conceptos del entorno de un sistema de abastecimiento de agua, algunas técnicas de aprendizaje y se hace un recorrido a propuestas que han hecho frente a la predicción, gestión y monitorización de las redes de distribución de agua.
- El Capítulo 3 describe la metodología usada para este trabajo, donde se sigue el paradigma KDD, integrado con el proceso ELT y la arquitectura del Data Lake dentro del entorno Big Data.
- El Capítulo 4 describe todo el proceso de la construcción del Data Lake, desde el análisis de las fuentes, hasta la obtención del conjunto de datos procesados que se utilizarán como entrada para el modelo de aprendizaje.
- El Capítulo 5 es la etapa de aprendizaje, en la que se describe el proceso de modelado para extraer los patrones del consumo de agua en el sector residencial de estudio. Se entrenan 3 métodos de regresión disponibles en la biblioteca ML de Spark y se evalúan en los conjuntos de prueba con los parámetros óptimos obtenidos con validación cruzada.
- En el Capítulo 6 se visualizan los resultados de las predicciones de los modelos entrenados en la etapa de aprendizaje, desarrollada en el capítulo 5, usando la herramienta *Qlik*.
- El capítulo 7 se presentan las conclusiones y se propone el trabajo futuro.

El documento tiene, por último, un apéndice que contiene el código de algunos procesos realizados durante las etapas de este trabajo, por ejemplo la configuración de *Flume* que se usó para la fase de ingestión de datos, el código en *scala* para la fase de limpieza usando las biblioteca *Breeze*, etc.

# Capítulo 2

# Marco teórico y Estado del arte

*El conocimiento que no se está utilizando para obtener más conocimiento, ni siquiera permanece, se descompone y desaparece.*

JD Bernal, 'Science in History'

**Resumen:** En este capítulo se hace una descripción general de los conceptos entorno al sistema de abastecimiento de agua, sus redes de distribución y la problemática de las fugas, donde su monitorización y análisis son parte fundamental para la detección y control de las mismas. También se hace un recorrido de algunas propuestas que han hecho frente a la predicción, gestión y monitorización de las redes de distribución de agua, usando métodos de aprendizaje automático, pero no en todos los casos se puede identificar el uso de herramientas Big Data.

## 2.1. Redes de distribución de agua

El ciclo integral del agua es la expresión que define el recorrido hecho por el agua desde su captación en estado bruto en la naturaleza, hasta su disponibilidad potabilizada para los usuarios, ya sean residenciales, comerciales o industriales y, finalmente, el que realiza para reintegrarse convenientemente depurada a la naturaleza.

Su captación se hace en manantiales, pozos, embalses, etc., para luego ser sometida a exigentes controles por las estaciones de tratamiento de agua potable, desde donde se transporta hasta los depósitos de las distintas poblaciones y, mediante sistemas de distribución, es servida al usuario. Una vez consumida, el agua residual se canaliza a través de redes de alcantarillado hasta las estaciones depuradoras de agua residuales, que las reintegran con toda clase de garantías nuevamente a arroyos, ríos, mar y medios receptores en general.

Este trabajo se centra exclusivamente en la etapa de distribución. La línea de distribución se inicia, generalmente, en el tanque de agua tratada. Consta de estaciones de bombeo, tuberías principales, secundarias y terciarias, tanques de almacenamiento intermediarios, válvulas que permitan operar la red, y sectorizar el suministro, diversos tipos de medidores de volumen y derivaciones domiciliares.


**Figura 2.1:** Ciclo integral del agua [24]

Las redes de distribución de agua (en ingles: Water Distribution Network - WDN) son sistemas de múltiples entradas y múltiples salidas a gran escala, con perturbaciones inciertas (como la demanda de agua de los consumidores) e involucran componentes de naturaleza. Además, la operación de las WDN está impulsada por demandas variables en el tiempo e implica un consumo considerable de energía eléctrica y la explotación de recursos hídricos limitados. Por lo tanto, la gestión de estas redes debe llevarse a cabo de manera óptima con respecto al uso de los recursos disponibles y la infraestructura, a la vez que se satisfacen los altos niveles de servicio para el suministro de agua potable.

### 2.1.1. Distrito Hidrométrico

Un distrito hidrométrico (District Metered Area - DMA) se define como un área discreta de una red de distribución de agua. Por lo general, se crea al cerrar las válvulas de límite para que siga siendo flexible a las demandas cambiantes. Muchas empresas de servicios de agua operan sus redes de tuberías como un sistema abierto donde el agua se alimenta desde más de una planta de tratamiento de agua a una red de tuberías interconectadas.

### 2.1.2. Agua no registrada

El Agua No Registrada (ANR) es el indicador de eficiencia de las redes de abastecimiento de agua más extendido del mundo. Definido como la diferencia entre el volumen de agua suministrada al sistema y el volumen de agua registrada en los contadores de los usuarios, engloba los consumos autorizados no medidos, los consumos no autorizados (fraudes), los errores de medida y las pérdidas físicas en la red.

Cuando un sistema de suministro se divide en áreas más pequeñas y manejables, es decir en DMA, la empresa puede dirigir mejor las actividades de reducción del ANR, aislar los problemas de calidad del agua y administrar mejor la presión general del sistema para permitir el suministro de agua las 24 horas en toda la red.

En España, el valor medio de este indicador ronda actualmente el 25 %. Sin embargo, a nivel mundial se estima en un 40 % y, por ejemplo, en las ciudades de Latinoamérica la cifra oscila entre el 33 y el 50 %. Esto quiere decir que, de media en el mundo, por cada 100 litros suministrados a la red sólo 60 son registrados en los contadores de los usuarios. Por ello,

el control del ANR es una prioridad para las entidades operadoras de cualquier municipio o ciudad como medida de optimización de la gestión del servicio. Un incremento de este indicador supone un mayor uso del agua y, en consecuencia, un aumento del consumo energético. Es decir, a mayor ANR, menor eficiencia y sostenibilidad del sistema [42].

Un enfoque efectivo para la gestión de ANR, es donde el sistema en su conjunto se divide en una serie de DMA, para los cuales se puede calcular ANR de forma individual. Cada subsistema deberá aislarse hidráulicamente para que pueda calcular el volumen de agua perdida dentro del DMA.

### 2.1.3. Fugas

Son las pérdidas de agua que se producen durante el recorrido del fluido desde los elementos reguladores situados en cabecera de red hasta los consumidores finales. Los porcentajes relativos a pérdidas reales de agua ofrecidos por empresas gestoras de redes de abastecimiento españolas varían notablemente. Las pérdidas de agua que se pierde en fugas, atribuida en su mayor parte a la fase de distribución tiene un valor medio del 13 %, en función del gran esfuerzo que los operadores realizan para minimizarlas mediante sectorización de las redes [43].

Es aquí donde el enorme potencial de las técnicas de aprendizaje automático está significando que las empresas encargadas de gestionar este tipo de infraestructuras apuesten cada vez más por modelos basados en el análisis de datos en tiempo real. Estos modelos son un gran avance en la predicción del comportamiento de infraestructuras y la detección de eventos anómalos que, extrapolado al contexto de este trabajo, se puede aplicar a predecir escenarios de agua consumida y detectar eventos anómalos originados por aparición de nuevas roturas.

## 2.2. Técnicas de aprendizaje para series temporales

Uno de los objetivos de este trabajo es predecir el consumo en una red de distribución de agua, donde la salida debe ser un valor continuo para una fecha determinada. Los datos de entrada están ordenados en el tiempo, por lo que se trata de un problema de predicción de series temporales. Dentro del aprendizaje supervisado, lo mencionado anteriormente, enmarca este trabajo dentro de la regresión. Los métodos propuestos para el aprendizaje, son métodos no lineales basados en árboles de decisión. Entre ellos están el árbol de decisión, Random Forest y Gradient Boosted Tree, que se describen a continuación.

### 2.2.1. Árbol de Decisión

Los árboles de decisión o Decision Tree (DT) son un método de aprendizaje supervisado no paramétrico, utilizado para la clasificación y regresión. El objetivo es crear un modelo que prediga el valor de una variable objetivo mediante el aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos. Los árboles de decisión funcionan al evaluar una expresión que contiene una característica en cada nodo y seleccionar una rama al siguiente nodo en función de la respuesta.

El árbol de decisión es un algoritmo que realiza una partición binaria recursiva del espacio de características. El árbol predice la misma etiqueta para cada partición inferior (hoja). Cada partición se elige seleccionando la mejor división de un conjunto de divisiones posibles, con el fin de maximizar la ganancia de información en un nodo de árbol [10].

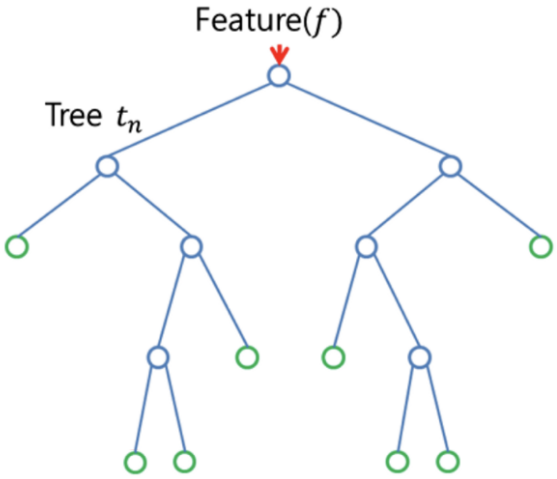


**Figura 2.2:** Representación de un árbol de decisión [8]

La ventajas que tienen los árboles de decisión es que son sencillos de entender, interpretar, realizan implícitamente la selección de características, y pueden manejar datos numéricos y categóricos. Los árboles de decisión requieren relativamente poco esfuerzo de los usuarios para la preparación de datos. Las relaciones no lineales entre parámetros no afectan el rendimiento del árbol.

Dentro de las desventajas que tienen los árboles de decisión es que pueden crear árboles demasiado complejos que no generalicen bien los datos, lo que llamamos sobreajuste. Pueden ser inestables porque pequeñas variaciones en los datos pueden dar como resultado la generación de un árbol completamente diferente (varianza). Se pueden crear árboles sesgados si dominan algunas clases.

La construcción recursiva del árbol se detiene cuando se alcanza la profundidad máxima, cuando no se encuentra un atributo que aumente la ganancia de información; o cuando ningún atributo obtenga nodos hijos que sean alcanzados por un número mínimo de instancias del conjunto de entrenamiento.

MLlib admite árboles de decisión para la regresión, utilizando características continuas y categóricas. La implementación divide los datos por filas, lo que permite un entrenamiento distribuido con millones de instancias.

#### 2.2.1.1. Impureza del nodo y ganancia de información

La impureza del nodo es una medida de la homogeneidad de las etiquetas en el nodo. MLlib proporciona dos medidas de impureza para la clasificación (Gini y entropía) y una medida de impureza para la regresión (varianza).

La varianza en problemas de tipo regresión está dada por:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$$

donde $y_i$ es el valor resultado (label) de la observación, $N$ es el número de observaciones y $\mu$ es la media dada por $\frac{1}{N}\sum_{i=1}^{N} y_i$

### 2.2.2. Random Forest

El modelo de Random Forest (RT) es un tipo de modelo aditivo que realiza predicciones combinando decisiones de una secuencia de modelos base. Más formalmente podemos escribir esta clase de modelos como:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + ...$$

donde el modelo final g es la suma de modelos básicos simples $f_i$. Aquí, cada modelo base es un árbol de decisión simple. Esta amplia técnica de uso de modelos múltiples para obtener un mejor rendimiento predictivo se denomina modelo ensemble. En Random Forest, todos los modelos base se construyen de forma independiente utilizando una submuestra diferente de los datos [9].

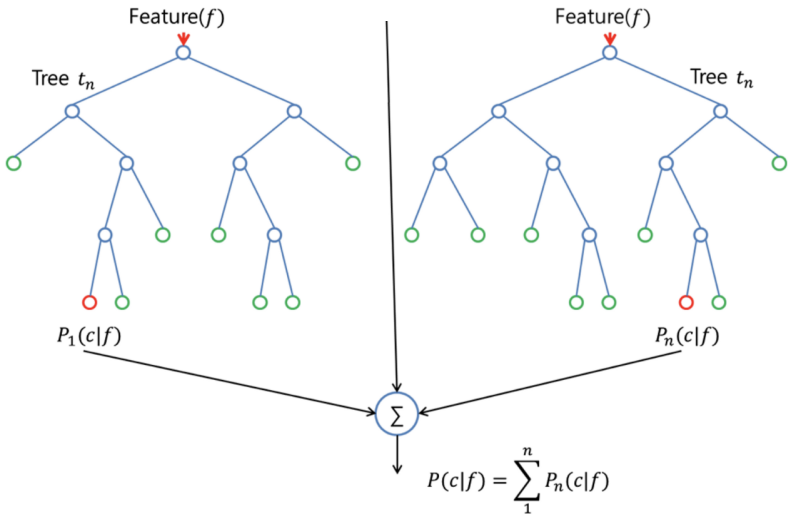


**Figura 2.3:** Representación de un Random Forest de 2 árboles [8]

En el caso de la clasificación, la predicción de cada árbol cuenta como un voto para la clase que corresponda, y por tanto, la etiqueta seleccionada como predicción será la clase que más votos reciba. Si se trata de un problema de regresión, la etiqueta será el promedio de las predicciones obtenidas con cada árbol.

Random Forest entrena un conjunto de árboles de decisión de forma paralela. Además aplica un submuestreo al conjunto de datos original en cada iteración, para obtener un conjunto de entrenamiento diferente. El entrenamiento del árbol de decisión se hace de la misma manera que para los árboles de decisión individuales. Los parámetros que pueden mejorar el rendimiento de Random Forest es la cantidad de árboles en el bosque y la profundidad del árbol. Aumentar la cantidad de árboles disminuirá la varianza en las predicciones, mejorando la precisión del modelo. El tiempo de entrenamiento aumenta linealmente con respecto a la cantidad de árboles.

### 2.2.3. Gradient Boosted Tree

Los árboles Gradient Boosted (GBT) son conjuntos de árboles de decisión. Los GBT entrenan iterativamente árboles de decisión para minimizar una función de pérdida. La idea general es calcular una secuencia de árboles muy simples, donde cada árbol sucesivo

se construye para los residuos de predicción del árbol anterior. Este método construirá árboles binarios, es decir, dividirá los datos en dos muestras en cada nodo dividido [11].

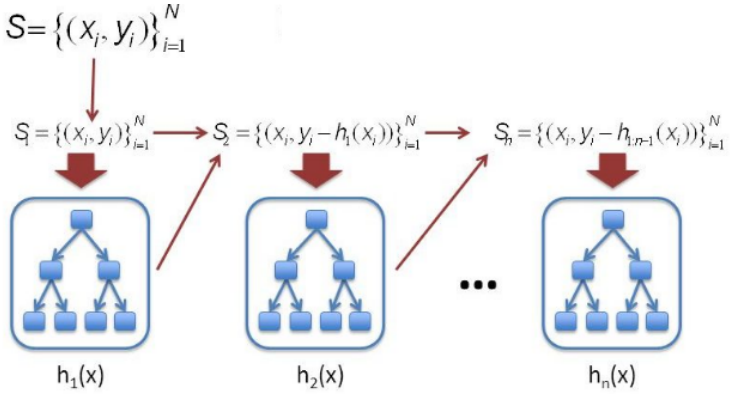


**Figura 2.4:** Representación de un Gradient Boosted Tree [12]

El aumento de gradiente entrena iterativamente una secuencia de árboles de decisión. En cada iteración, el algoritmo usa el conjunto actual para predecir la etiqueta de cada instancia de entrenamiento y luego compara la predicción con la etiqueta verdadera. El conjunto de datos se vuelve a etiquetar para poner más énfasis en las instancias de entrenamiento con predicciones pobres. Por lo tanto, en la siguiente iteración, el árbol de decisiones ayudará a corregir errores previos. El mecanismo específico para volver a etiquetar las instancias se define mediante una función de pérdida. Con cada iteración, los GBT reducen aún más esta función de pérdida en los datos de entrenamiento.

Al igual que los árboles de decisión, los GBT manejan características categóricas, se extienden a la configuración de clasificación multiclase, no requieren escalado de características y son capaces de capturar interacciones no lineales e interactivas.

La biblioteca *MLlib* admite GBT para la clasificación binaria y para la regresión, utilizando funciones continuas y categóricas. Implementa GBT utilizando la implementación del árbol de decisión existente. el parámetro principal en GBT que puede mejorar el rendimiento es en número de iteraciones. Esto establece el número de árboles en el conjunto. Cada iteración produce un árbol. Aumentar este número hace que el modelo sea más expresivo, lo que mejora la precisión de los datos de entrenamiento. Sin embargo, la precisión puede verse afectada si es demasiado grande.

Para regresión, GBT soporta 2 funciones de perdida que relacionamos a continuación, donde $N$ es el número de instancias, $y_i$ la etiqueta de la instancia, $x_i$ los atributos de la instancia y $F(x_i)$ es la etiqueta predicha:

1. Error cuadrático, también llamado pérdida L2. Es la pérdida predeterminada para tareas de regresión.

$$\sum_{i=1}^{N}(y_i - F(x_i))^2)$$

2. Error absoluto, también llamado pérdida L1. Puede ser más robusto a los valores atípicos que el error cuadrado.

$$\sum_{i=1}^{N}|y_i - F(x_i)|$$

## 2.3. Aproximaciones de Big Data para WDN

El manejo de los datos y la información es quizá uno de los temas de mayor importancia en la búsqueda de la innovación y el desarrollo de cualquier actividad. En la actualidad se recopila una cantidad inigualable de información de diferentes fuentes y formatos, de la cual, una parte se analiza y estudia. La gran mayoría de dicha información, hasta hace poco tiempo era descartada por no ofrecer de forma directa, información relevante en la toma de decisiones.

La revisión bibliográfica de los desarrollos y aplicaciones de la búsqueda de información y conocimiento, por medio de herramientas Big Data, empleadas en temas de abastecimiento de agua es muy escasa. Aunque el agua es un bien público, la información alrededor de ésta temática presenta un grado de confidencialidad debido, principalmente, a que las empresas de abastecimiento son, en su gran mayoría, privadas o mixtas. La información que manejan estas empresas, no llega a ser pública y no es fácil conseguir datos que sirvan para hacer este tipo de trabajos o investigaciones.

De igual forma, en las empresas del sector del agua y en general en la ingeniería clásica, aún existe un poco de recelo o incredulidad sobre la aplicación de nuevos paradigmas para el análisis y estudio de los datos en la gestión o toma de decisiones en sus redes y se prefiere el diseño y modelado clásico.

A continuación se mencionan algunos documentos relacionados con el estudio y predicciones en redes de abastecimiento de agua. Aunque mencionan el uso de algoritmos de aprendizaje automático, no podemos identificar si el proceso se realizó en entorno Big Data.

- **Water demand estimation and outlier detection from smart meter data using classification and Big Data methods** [34].

  Este trabajo tiene como objetivo predecir la demanda de agua y detectar comportamientos sospechosos, por ejemplo, una fuga, un fallo en el medidor o incluso un fraude, al extraer los patrones de consumo de agua. Lo hacen mediante una clasificación no supervisada de los patrones de demanda a partir de datos de medidores. Por tanto, puede pronosticar la demanda de un cliente en particular, un grupo o el distrito hidrométrico como un todo. Se desarrolla en un marco basado en tecnologías Big Data. El *framework* propuesto, está compuesto por tres módulos: almacenamiento, procesamiento y resultado. Usa el sistema de archivos distribuidos de *Hadoop* (HDFS), el motor de procesamiento *Apache Spark* y su biblioteca MLlib para el agrupamiento no supervisado. Finalmente, los resultados obtenidos de la etapa de procesamiento se guardan en la base de datos distribuida *Cassandra*. El enfoque de los autores en este trabajo es básicamente clasificación de patrones en entorno Big Data, donde agrupan los medidores de acuerdo a los niveles de consumo de agua de los habitantes. Este es el primer trabajo que he encontrado en la revisión bibliográfica que hace estimación de demanda en un entorno Big Data.

- **Big Data and Data Analytics applied to the Monitoring of Water Distribution Networks** [35]

  Este trabajo usa herramientas de Big Data para la monitorización de las redes de distribución de agua. En este estudio procesan tres conjuntos de datos de diferentes ciudades, planteado en dos etapas. En la primera los autores identifican grupos de sensores que comparten el mismo patrón de comportamiento, mediante agrupamiento. Esta primera etapa hace el mismo proceso descrito en el artículo [34], ya que este

trabajo hace parte de la continuación de esa investigación. En la segunda etapa, aplican tres modelos de aprendizaje (Regression tree, Classification tree y GBM), para encontrar la relación entre los patrones de demanda y las variables de la cuenta del medidor. En este punto podemos asegurar que nuestro enfoque es más general y propone estudiar el comportamiento del consumo de la ciudad o sector y no de cada consumidor como lo hacen los autores en este trabajo.

- **Predictive models for forecasting hourly urban water demand** [36]

  En este articulo los autores describen y comparan una serie de modelos predictivos para pronosticar la demanda de agua. Los modelos se obtienen utilizando datos de series de tiempo del consumo de agua en un área urbana, con datos cada hora. Lo que nos hace suponer que el desarrollo de este trabajo no fue considerado dentro del enfoque Big Data. Para su estudio, los autores tuvieron en cuenta, además de los valores de consumo de agua, información sobre variables climáticas: temperatura, velocidad del viento, lluvia y presión atmosférica. Aseguran que todos estos factores están relacionados con el comportamiento de la demanda de agua. Los modelos predictivos propuestos incluyen redes neuronales artificiales, regresión de búsqueda de proyección, regresión adaptativa multivariantes, random forest y regresión vectorial. Los análisis de este estudio se desarrollaron bajo un enfoque estadístico y se llevaron a cabo utilizando el entorno de lenguaje de programación R.

- **Application of HADOOP to Store and Process Big Data Gathered from an Urban Water Distribution System** [37]

  En este documento, se investiga la plataforma Apache Hadoop con respecto a una posible solución de base de datos basada en NoSQL. Presentan experimentos comparativos que evalúan el rendimiento de las bases de datos Hadoop y MySQL para encontrar la mejor solución a la problemática de la cantidad de información generada por varios tipos de sensores, en una WDN, funcionando en tiempo real entregando una gran cantidad de datos. Consideran muy importante el análisis Big Data para la gestión de WDN y la valoración de las bases de datos NoSQL para datos de agua, ya que se encuentra actualmente en las primeras etapas. No obstante los resultados revelaron que la base de datos MySQL supera a HADOOP en rendimiento, un resultado que es de esperarse, ya que el experimento se trata de ejecutar procesos muy sencillos de búsqueda de valores máximos y promedios dentro del conjunto de valores de flujo de agua. Esta claro que comparar el tiempo de ejecución no es una característica suficiente para determinar la mejor opción a elegir, entre una base de datos relacional y una NoSql para tratar temas entorno a las WDN. Aunque podemos considerar que es una explicación del porque las herramientas Big Data no son muy utilizadas en el sector del agua.

- **Hybrid regression model for near real-time urban water demand forecasting** [38]

  Este artículo presenta un modelo híbrido para la predicción de la demanda de agua por hora, para construir una predicción base. El método se basa en un modelo de regresión de vector de soporte fuera de línea. En este modelo, se construye un proceso de serie temporal de Fourier para mejorar la predicción base. Esta adición produce una herramienta capaz de eliminar muchos de los errores y gran parte del sesgo inherente a una estructura de regresión fija al responder a los nuevos datos de series de tiempo entrantes. El resultado de la predicción que se obtiene combinando estos

dos métodos tiene una precisión alta. Al igual que [36], el conjunto de datos obtenidos cada hora, supone una cantidad de información relativamente fácil de procesar con métodos tradicionales de moldeamiento matemático y estadístico. Aunque nuestro conjunto de datos no es muy grande, el análisis, procesamiento y aprendizaje esta diseñado para escalar.

- **Improving the Performance of Water Demand Forecasting Models by Using Weather Input**[39]

  En este trabajo se estudian tres modelos de pronóstico diferentes: un modelo heurístico adaptativo, un modelo de transferencia/ruido y un modelo de regresión lineal múltiple. El rendimiento de los modelos se estudiaron con y sin la utilización de la información meteorológica, con el fin de evaluar la posible mejora del rendimiento debido al uso de ésta información. Los autores han tenido en cuenta las condiciones climáticas, al igual que en el estudio [36], ya que aseguran que incluyendo dichas variables, en los modelos de previsión de la demanda de agua que utilizan los valores de demanda como único insumo, los errores de pronóstico son bastante grandes. En nuestro trabajo comprobamos que efectivamente los modelos de predicción que usamos son capaces de generar un pronóstico bastante preciso y esto es a causa de que la fuente de datos se centra en los valores de demanda. Este estudio está desarrollado principalmente con modelo matemáticos.

- **Adaptive water demand forecasting for near real-time management of smart water distribution systems**[41]

  Este documento presenta una metodología para realizar pronósticos adaptativos de la demanda de agua hasta 24 horas futuras con el objetivo de respaldar la gestión operativa casi en tiempo real de las WDN. Se basa exclusivamente en el análisis de series temporales de demanda de agua y hace uso de redes neuronales artificiales evolutivas. Los autores presentan 2 enfoques uno con múltiples modelos de redes, y otro usa un único modelo. Los resultados generan pronósticos precisos para ambos enfoques. Debido a que se basa exclusivamente en el análisis de señales de demanda de agua y como mencionamos anteriormente, cuando se utilizan los valores de demanda como único insumo la predicción es muy buena.

Dentro de esta revisión bibliográfica se observa que antes de 2015, las investigaciones, trabajos y documentos centrados en ingeniería del agua, eran ajenos al uso y aplicación en entorno Big Data. No obstante, a partir de esa fecha tampoco se ha producido un cambio muy sustancial y aún no encontramos suficientes documentos que lo propongan, teniendo en cuenta la gran cantidad de datos que se manejan y las oportunidades de mejora que ofrece. Por esta misma razón, el concepto de Data Lake supone un novedoso aporte a los estudios sobre esta materia.

Una de las ventajas de la utilización del paradigma KDD para generar reglas de predicción de demanda de agua, es que los modelos matemáticos solo tienen en cuenta la parte cuantitativa del problema, mientras que con el aprendizaje automático, a partir de los datos observados, se consideran variables tanto cuantitativas como cualitativas. Como valor agregado, en este trabajo estamos tomando dicha metodología como base, integrándola con el proceso ELT.

# Capítulo 3

# Metodología

*Cuando una técnica te dice "cómo" y una filosofía te dice "qué", una metodología contendrá elementos de "qué" y "cómo".*

Peter Checkland (Pensamiento sistémico)

**Resumen:** Este capítulo se inicia con una descripción del problema general que se quiere abordar con este trabajo, también se menciona la estructura física de los datos fuente con los que se trabajará en las diferentes etapas. Posteriormente se explica la metodología que se va a seguir, que es una integración entre el proceso de descubrimiento conocido como KDD y el proceso ELT. Finalmente se describe el esquema de la arquitectura del Data Lake (DL) en el entorno Big Data que se va a usar para desarrollar el proyecto.

## 3.1. Descripción del problema

Un sistema de abastecimiento de agua corresponde a una estructura compleja, en la cual se llevan a cabo procesos de producción, transporte, distribución de agua, entre otras. Por lo tanto el diseño de planes de mantenimiento y gestión, tanto para las condiciones actuales como las futuras demandas que afectan las presiones en la red de tuberías, así como la reducción de costos para las empresas, representa una parte integral dentro de la estrategia de la gestión de la red.

Para enfrentar esta necesidad, las empresas abastecedoras de agua han implementado variadas estrategias para la recolección de datos casi en tiempo real, donde los datos son obtenidos directamente desde los sensores. Antes de dichas implementaciones, las lecturas disponibles solían ser mensuales o bimensuales y se recopilaban in situ, lo que dificultaba el análisis adecuado del comportamiento del cliente y la evaluación para ampliar el estado de la infraestructura. Teniendo en cuenta el uso de estas tecnologías, y el hecho de que algunos sensores han estado recopilando datos durante varios años, es cierto que la cantidad de datos generados en el sistema supera los límites de cualquier herramienta de análisis estadístico convencional. Esto abre la posibilidad para la implementación de tecnologías Big Data.

Para la mayoría de problemas actuales con datos masivos es necesario el uso de una solución distribuida escalable porque las soluciones secuenciales no son capaces de abordar

tales magnitudes. Varias plataformas para el procesamiento a gran escala (como Spark o el ecosistema Hadoop) han intentado afrontar la problemática del Big Data en los últimos años [17]. Adicional a ello, las técnicas de aprendizaje automático han demostrado ser elementos muy útiles para el análisis, las cuales se pueden implementar en una plataforma de Big Data.

No obstante, los problemas de predicción de series temporales son un tipo difícil de problema de modelado predictivo. A diferencia del modelo predictivo de regresión, las series temporales también agregan la complejidad de una dependencia de secuencia entre las variables de entrada. Es posible abordar esta problemática con el modelo predictivo de regresión, pero para este caso no se ésta teniendo en cuenta la dependencia con el tiempo.

Para este trabajo contamos con los datos de una empresa municipal prestadora del servicio de agua potable, que sirve a aproximadamente 400.000 habitantes y cuenta con varias plantas de tratamiento y estaciones de bombeo principales, distribuidos a través de toda su red [1]. Los datos fuente pertenecen al nodo principal que abastece de agua a la zona residencial y están conformados por la información de 2 sensores, donde uno mide el caudal y el otro la presión. Las lecturas de los sensores se realiza en tiempo real y se almacenan en segundos, pero para su procesamiento con las tecnologías que maneja la empresa actualmente, sólo se toma la medida obtenida cada 10 minutos, es decir, los datos fuente contienen 144 observaciones para 1 día por cada sensor, desde la hora 00:00 hasta las 23:50.

Los datos suministrados están contenidos en dos carpetas independientes, una por cada sensor, las cuales contienen los datos diarios distribuidos en archivos .csv. El nombre de cada archivo es la fecha de medición del sensor.

## 3.2. Metodología

La metodología usada en este proyecto, que está representada en la Figura 3.1, es una integración entre el proceso de descubrimiento conocido como KDD y el proceso ELT. La extracción de conocimiento está principalmente relacionado con el proceso KDD, que se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información [15]. En el enfoque tradicional dicho repositorio es un Enterprise Data Warehouse (EDW), pero para este trabajo, que se realiza en entorno Big Data, se ajusta mucho mejor la construcción de un Data Lake, no solo por el volumen de datos, sino también por su flexibilidad, esquema, manejo de variedad de fuentes, entre otros. En el EDW sus modelos de datos están diseñados para que cambien lentamente por ser datos altamente estructurados y rígidos, lo que los hace demasiado frágiles para soportar el volumen y la variedad de grandes volúmenes de datos. El enfoque del Data Lake evita estos problemas [5]. En el Apartado 3.3 se amplia mucho más las características y bondades del Data Lake.

En el proceso ELT, los datos se extraen de las fuentes, pero en lugar de transformarlos, se cargan directamente a sistemas de almacenamiento, en nuestro caso Hadoop Distributed File System (HDFS). Este proceso se lleva a cabo de forma eficiente dado que no se requiere validar los datos respecto a ningún esquema. Cuando los datos son necesarios, se construye su esquema, se transforman y se entregan para su consumo final [20].

[1] Por temas de confidencialidad no se puede proporcionar información más específica del problema y su implementación. Por este mismo motivo tampoco se permite mostrar el esquema de la red de distribución de aguas.

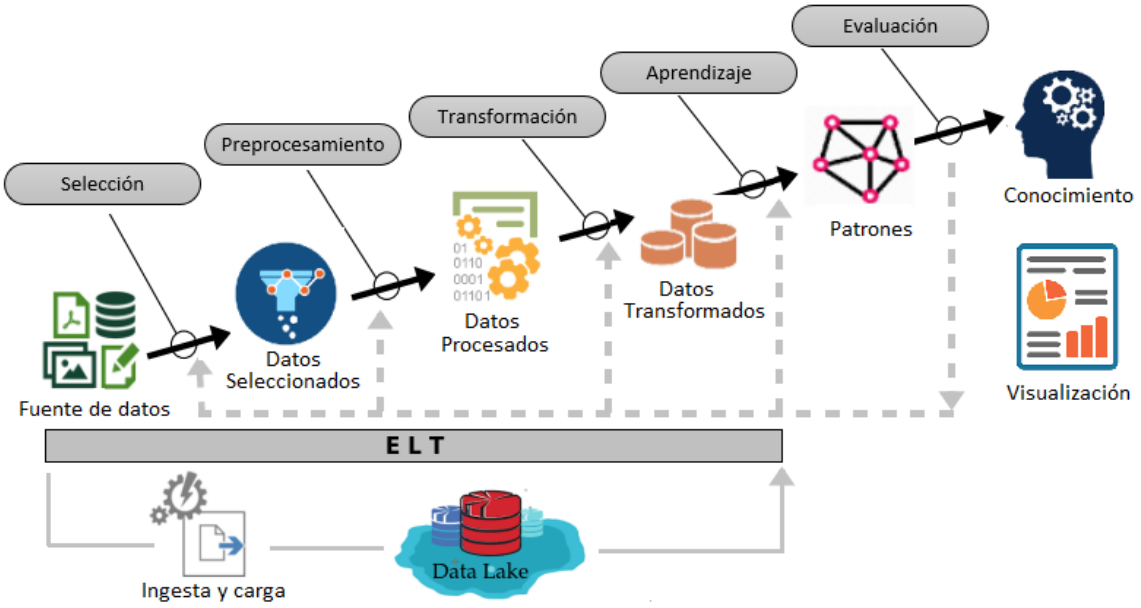


**Figura 3.1:** Metodología KDD + ELT (Fuente: elaboración propia)

1. **Análisis de las fuentes**. La fase de análisis consta de dos etapas principales, ambas etapas están relacionadas con la exploración de los datos: descubrimiento de los datos y detección de anomalías en los datos. Donde incluye el análisis de las fuentes, el modelo conceptual y el mapa lógico de datos.

2. **Ingestión y carga**. En esta etapa se hace la conexión y acceso a las fuentes de datos para almacenarlos en el Data Lake. Aquí hacemos la carga de los archivos que están en disco local y los almacenamos en HDFS en el directorio *raw data*.

3. **Selección de datos**. En esta etapa se determinan las fuentes de datos y el tipo de información a utilizar. Es la etapa donde los datos relevantes para el análisis son extraídos desde la o las fuentes de datos.

4. **Preprocesamiento**. Esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes de datos en una forma manejable, necesaria para las fases posteriores. En esta etapa se utilizan diversas estrategias para manejar datos faltantes o en blanco, datos inconsistentes o que están fuera de rango, obteniéndose al final una estructura de datos adecuada para su posterior transformación.

5. **Transformación**. Consiste en el tratamiento preliminar de los datos, transformación y generación de nuevas variables a partir de las ya existentes con una estructura de datos apropiada.

6. **Aprendizaje**. Es la fase de modelado propiamente dicha, en donde son aplicados métodos de inteligencia artificial, con el objetivo de extraer patrones desconocidos, válidos, nuevos, potencialmente útiles y comprensibles y que están contenidos u ocultos en los datos.

7. **Interpretación y Evaluación**. Se identifican los patrones obtenidos y que son realmente interesantes, basándose en algunas medidas y se realiza una evaluación de los resultados obtenidos.

8. **Presentación del conocimiento**. Visualización y representación del conocimiento para presentar el conocimiento extraído del usuario.

## 3.3. Data Lake

Mediante el uso de infraestructuras de Big Data, las empresas están comenzando a acumular volúmenes de datos crecientes para realizar análisis o simplemente almacenarlo para un uso indeterminado en el futuro. En este punto, el concepto de Data Lake puede ayudar a resolver el problema de almacenamiento, acceso e integración de datos.

Un Data Lake es un repositorio que contiene una gran cantidad de datos en bruto en su formato original (estructurado, no estructurado o semiestructurado), para su transformación o uso posterior. De hecho, las empresas vierten los datos y los recuperan cuando lo requieren. Únicamente en ese momento se procede a ordenarlos y a diseñar una estructura que haga más fácil su posterior análisis.

*"Se puede imaginar como un gran conjunto de datos para reunir todos los datos históricos acumulados y los nuevos datos (estructurados, no estructurados y semiestructurados más binarios de sensores, dispositivos, etc.) en tiempo casi real en un solo lugar, en el que los requisitos de esquema y datos no se definen hasta que se consultan los datos"* [6].

El Data Lake se asocia a menudo con el almacenamiento de objetos orientado a Hadoop. En este escenario, los datos de una organización se cargan primero en la plataforma Hadoop y, a continuación, se aplican las herramientas de análisis y de minería de datos a los datos que residen en los nodos clúster de Hadoop.

Algunas de las características del Data Lake son:

- Contiene todos los datos de las fuentes originales, sin rechazar ningún tipo de dato.
- Utiliza una arquitectura plana para almacenar los datos.
- Los datos en el Data Lake se vuelven accesibles tan pronto como se crean.
- Los datos se almacenan sin transformar o apenas transformados.
- Los datos se transforman y se aplica un esquema sólo para satisfacer las necesidades de análisis.
- Las estrategias de los Data Lake pueden combinar enfoques de bases de datos SQL y NoSQL.
- No requiere un esquema rígido o la manipulación de los datos, pero requiere mantener el orden de llegada de los datos.
- Son aptos para aprovechar grandes cantidades de datos de manera consistente, con algoritmos de gestión analítica en tiempo real.
- La obtención de nuevos datos en el lago puede ocurrir gradualmente y no afectará a los modelos existentes.
- Los datos usan un esquema dinámico creado al momento de la ejecución de la consulta.

- Con el Data Lake, los usuarios pueden tomar lo que para ellos es relevante y obviar el resto.
- No se requiere una clasificación de datos perfecta.

El Data Lake comienza con datos en bruto y madura a medida que fluyen más datos, a medida que los usuarios y las máquinas crean metadatos, y a medida que la adopción del usuario se amplía. La madurez de los datos es una consecuencia natural de la interacción y retroalimentación continua del usuario en la interacción de la capa de gestión de metadatos, que refina continuamente el Data Lake y mejora el descubrimiento.

### 3.3.1. Ventajas

A diferencia de la visión monolítica de un único modelo de datos para toda la empresa, un Data Lake flexibiliza la estandarización y posterga el modelado, lo que da como resultado un potencial casi ilimitado para el conocimiento operacional y el descubrimiento de datos [5].

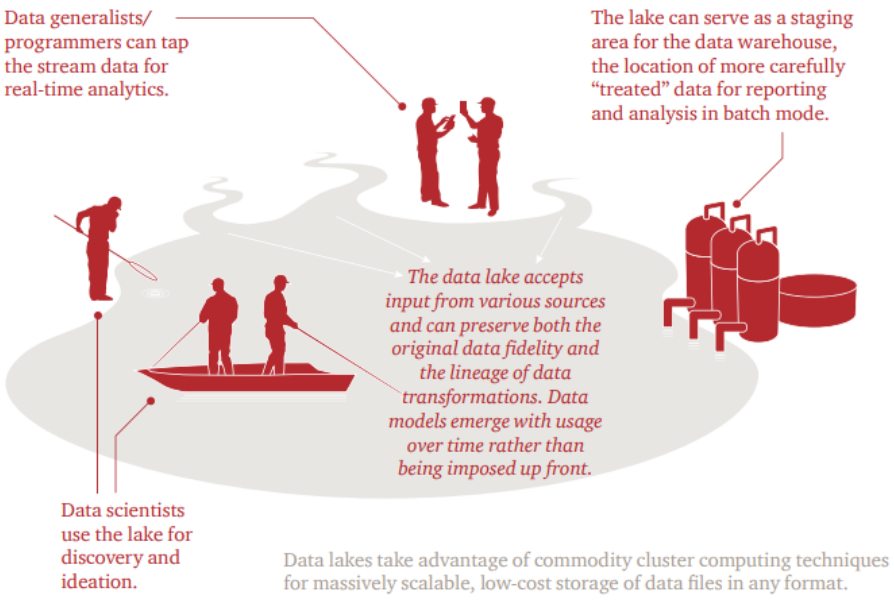


**Figura 3.2:** Representación de un Data Lake [5]

La principales ventajas que ofrece el Data Lake:

1. **Tamaño y bajo costo:** los Data Lake son gran tamaño. Pueden ser de un orden de magnitud menos costoso por configuración de terabyte para configurar y mantener que los EDW. Con Hadoop, los volúmenes de datos a escala de petabyte no son costosos ni complicados de construir y mantener.
2. **Fidelidad:** los Data Lake conservan los datos en su forma original y capturan los cambios en los datos y semántica contextual en todo el ciclo de vida de los datos. Este enfoque es especialmente útil para el cumplimiento y la auditoría interna. Si los datos han sufrido transformaciones, agregaciones y actualizaciones, la mayoría de las

organizaciones suelen tener dificultades para juntar datos cuando surge la necesidad y tienen pocas esperanzas de determinar la procedencia clara.

3. **Facilidad de acceso:** la accesibilidad es fácil en el Data Lake, como consecuencia de preservar los datos en su forma original. Ya sea estructurado, no estructurado o semiestructurado, los datos se cargan y almacenan tal como se van a transformar más adelante. Los usuarios de toda la empresa pueden ver la información de todas las áreas, no están limitados por silos organizativos o esquemas rígidos.

4. **Vinculación tardía:** Los datos están vinculados a un esquema dinámico creado al momento de la ejecución de la consulta. El principio de vinculación tardía traslada el modelado de datos de los administradores de bases de datos a equipos localizados como analistas de negocios y científicos de datos, que pueden ayudar a crear un contexto flexible y específico de dominio.

5. **Flexibilidad:** crecer y escalar sobre la marcha. Los Data Lake son altamente escalables y flexibles. Eso no necesita mucha elaboración. El sistema y los procesos se pueden escalar fácilmente para manejar cada vez más datos.

6. **Madurez:** Los análisis extraídos del Data Lake se vuelven cada vez más valiosos a medida que crecen los volúmenes de datos, la variedad de datos y la riqueza de metadatos.

7. **Inmediatez:** los Data Lake están diseñados para análisis Big Data y acciones en tiempo real basadas en análisis en tiempo real.

8. **Integración:** Una ventaja final, como describe la Figura 3.2 es que puede servir de antesala para el manejo y transformación de datos no estructurados para luego transmitirlos al EDW.

### 3.3.2. Contraste con EDW

El EDW es el primer paso para poder tomar decisiones en una empresa, por eso no hay que prescindir de él. Sin embargo, el Data Lake permite ampliar el horizonte respecto al nuevo enfoque que están adquiriendo los datos en la actualidad.

Con base en las características mencionadas del Data Lake y en sus ventajas descritas anteriormente, podemos extraer 5 diferenciadores clave con respecto al Data Warehouse:

1. Para desarrollar un EDW se deben conocer los procesos de negocio, y se debe dedicar un tiempo considerable en el análisis de los datos y creación de perfiles de datos. Todo ello para obtener un modelo de datos muy estructurado diseñado para generar informes. Para simplificar el modelo de datos y ahorrar espacio, se excluyen todos los datos que no se utilizan. Por su parte el Data Lake conserva todos los datos, no solo los datos que usa, sino también los que pueden usarse en el futuro. Este enfoque es posible porque el hardware usado para el Data Lake garantiza escalabilidad y la persistencia de los datos a bajo costo.

2. El EDW ignora en gran medida las fuentes de datos no tradicionales, como logs, datos de sensores, redes sociales, imágenes, entre otros. Data Lake admite todos los tipos de datos, independientemente de su forma o estructura. Se almacenan en su formato original y son transformados al momento de usarlos, este enfoque se conoce como “Schema on Read” frente al “Schema on Write” utilizado por el EDW.

3. El EDW es ideal para los usuarios operativos de las empresas, que son la gran mayoría, porque está bien estructurado, es fácil de usar y comprender, y está diseñado para responder a sus preguntas y generar los informes que usan con frecuencia. No obstante, existe un porcentaje menor de usuarios que hace un análisis más profundo sobre los datos y aunque usan la información del EDW, generalmente requieren complementar su estudio con otras fuentes que se encuentran fuera del almacén e incluso fuera de la organización. El enfoque del Data Lake apoya igualmente a todos estos usuarios. Los científicos de datos pueden ir al lago y trabajar con los conjuntos de datos grandes y variados que necesiten, mientras que otros usuarios hacen uso de vistas más estructuradas de los datos proporcionados para su uso.

4. Los Data Lake son altamente escalables y flexibles. No necesita mucha elaboración, por ello se adaptan a los cambios con facilidad. En cuanto al EDW puede adaptarse al cambio, pero debido a la complejidad del proceso de carga de datos, diseño y el trabajo realizado para facilitar el análisis y la generación de informes, estos cambios necesariamente consumirán algunos recursos de desarrollador y tomarán un tiempo.

5. La necesidad cada vez mayor de respuestas más rápidas, ha dado lugar al concepto de inteligencia empresarial de autoservicio. Dado que en el Data Lake, todos los datos están almacenados en su forma original y siempre accesibles para alguien que necesita usarlos, los usuarios tienen la capacidad de explorarlos y responder a sus necesidades a su ritmo. Pero es posible que usuarios operativos no quieran hacer ese trabajo y se conformen con usar los informes diseñados en su EDW. Por ello, la rapidez en la obtención de la información que el usuario requiere, depende de cada caso, si es información nueva será más rápido usando un Data Lake, pero si es información conocida y contenida dentro de la estructura del EDW, es más rápido este ultimo.

## 3.4. Arquitectura del Data Lake propuesto

El volumen de datos que generan entornos sensorizados es creciente y puede llegar a ser exponencial. Concretamente en nuestro caso, las redes de distribución de agua, manejan datos en bruto de distintos tipos de sensores, que recogen las medidas de presión y caudal. Estos datos en conjunto, almacenados cada segundo, puede llegar a ser inmanejable con tecnologías tradicionales. Es aquí donde el concepto de Data Lake entra a jugar un papel muy importante para facilitar su ingestion, procesamiento y análisis haciendo uso de las tecnológicas Big Data.

En este trabajo se propone una arquitectura de Data Lake que sea capaz de gestionar la gran cantidad de datos que puede generar los sensores de una WDN en tiempo real. La Figura 3.3 muestra el esquema de dicha arquitectura, que esta conformada por 6 los componentes y sus tecnologías relacionadas.

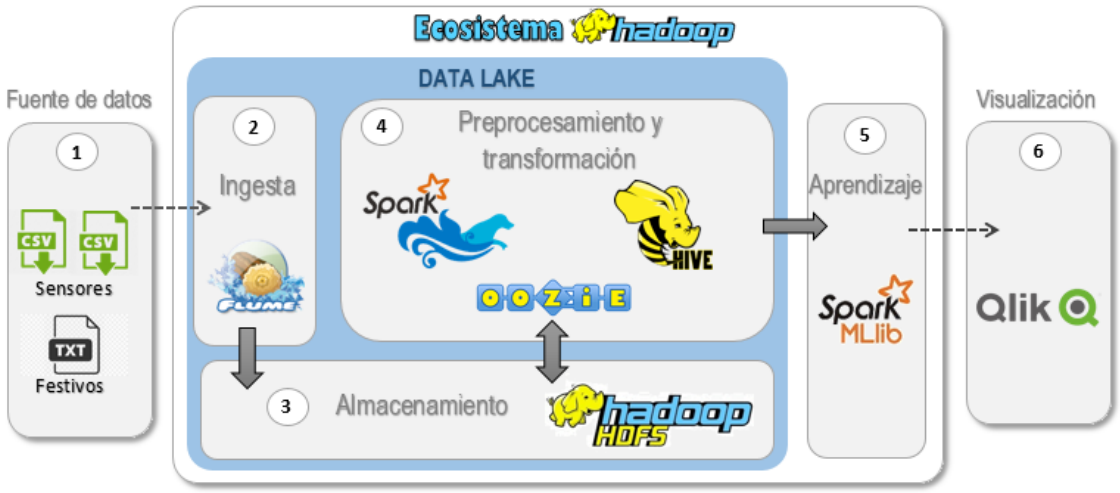


**Figura 3.3:** Arquitectura del Data Lake. (Fuente: elaboración propia)

A continuación se resume cada uno de los componentes que conforman la arquitectura del Data Lake:

1. **Fuente de datos:** Este componente es la entrada usada para alimentar nuestro Data lake. Los datos pertenecen a las observaciones de 2 tipos de sensores (caudal y presión) de una WDN y del calendario festivo de 2015. Para hacer la construcción del Data Lake iniciamos con el análisis de las fuentes que se describe en el Apartado 4.1.

2. **Ingesta:** Para las fuentes de datos de los sensores, usamos *Apache Flume*[2], un servicio distribuido, fiable, y altamente disponible para recopilar, agregar, y mover eficientemente grandes cantidades de datos. *Flume* permite ingerir archivos de datos contenidos en un directorio en el disco a una ruta en HDFS, con ayuda del tipo de fuente llamada *Spooldir*, donde almacenamos cada archivo en un directorio de forma independiente, que es justo lo que necesitamos para el desarrollo de este proyecto. *Spooldir* es una fuente que observa el directorio de entrada y analiza los eventos a partir de los nuevos archivos y a medida que aparezcan. En el apartado 4.2 se describe con más detalle la configuración.

3. **Almacenamiento:** Para el almacenamiento se diseñó una estructura que está compuesta por diferentes niveles de refinamiento para organizar, preservar y permitir la trazabilidad de los datos. Para ello usamos el Hadoop Distributed File System (HDFS) que es un sistema de archivos distribuido, escalable y portátil escrito en Java para el framework Hadoop. HDFS divide los archivos en bloques de un mismo tamaño y distribuidos entre los nodos que forman el clúster de datos para garantizar escalabilidad y la persistencia de los datos a bajo costo. El detalle de este componente se describe en el Apartado 4.3.

4. **Preprocesamiento y transformación** Para la limpieza de las series tenemos a *Breeze* [3], que es el conjunto principal de bibliotecas para *ScalaNLP*[3], que incluye

[2] `https://flume.apache.org/`
[3] `http://www.scalanlp.org/`

álgebra lineal, computación numérica y optimización. En este proyecto se usa la interpolación lineal para eliminar los datos nulos de las series. Afortunadamente, la versión de *Apache Spark* que trae Cloudera 5.10 es compatible y se pudo hacer uso de la biblioteca sin problemas.

La mayor parte de las transformaciones se hicieron con ayuda de *Apache Hive*[4] que nos facilita enormemente el trabajo con los datos, ya que su funcionamiento es sencillo, a través de querys HQL (HiveQL), donde se lanzan consultas que son traducidas a trabajos MapReduce. El desarrollo detallado de este componente de la arquitectura DL se describe en el Apartado 4.4.

5. **Aprendizaje:** Las tareas de predicciones de consumo, son realizadas usando *MLlib*[5], una biblioteca de aprendizaje automático de Apache Spark. Se realiza el proceso de modelado para extraer los patrones del consumo de agua en la WDN. Los datos se entrenan usando 3 algoritmos de regresión disponibles en la biblioteca ML de Spark, Decision Tree, Random Forest y Gradient Boosted Tree y se evalúan en los conjuntos de prueba con los parámetros óptimos obtenidos con validación cruzada. todo el desarrollo de esta etapa se describe en el Capítulo 5

6. **Visualización:** Finalmente, los resultados de los modelos obtenidos de la etapa de aprendizaje son visualizados mediante *Qlik*[6], una herramienta de Business Intelligence (BI o Inteligencia de negocio) que permite recolectar datos desde diferentes orígenes, modelarlos para facilitar su manejo y presentarlos de forma muy visual. Se diseñaron varias gráficas que permiten comparar los resultados obtenidos y discernir sobre las hipótesis planteadas en los objetivos de este trabajo. Los detalles de este componente se describen en el Capítulo 6.

[4] `https://hive.apache.org/`
[5] `https://spark.apache.org/mllib/`
[6] `https://www.qlik.com/es-es`

# Capítulo 4

# Construcción del Data Lake

*El Data Lake es un gran cuerpo de agua en su estado natural. Los datos fluyen desde una fuente para llenar el lago, y los usuarios tienen acceso al lago para examinar, tomar muestras o sumergirse.*

James Dixon (2010)

**Resumen:** Este capítulo describe todo el proceso de la construcción del Data Lake, siguiendo la arquitectura DL planteada en el Apartado 3.4. Desde el análisis de los datos en bruto, hasta la obtención del archivo con las características definidas con los datos refinados, que se usará en la etapa de aprendizaje descrita en el Capítulo 5. El proceso se inicia con el análisis de las fuentes, cuyo objetivo es el descubrimiento de los datos, ésta etapa incluye la arquitectura de datos, el modelo conceptual y el mapa lógico de datos refinados. Se detalla el proceso de ingesta de datos, que serán almacenados en la carpeta rawData de HDFS. Como también se define la estructura de almacenamiento en HDFS. Para finalizar y teniendo en cuenta las definiciones y diseño de los datos anteriormente descrito, se realizan todas las transformaciones necesarias para obtener los datos refinados de acuerdo al mapa lógico.

## 4.1. Arquitectura de datos

Ésta es la primera etapa para el descubrimiento de los datos y el primer componente de la construcción del DL, definido en la arquiectura DL (Figura 3.3). Es equivalente al modelado de información (diseño conceptual) en el desarrollo de un sistema software [20]. A continuación se describen las fuentes de datos y se muestra el modelo conceptual diseñado para este proyecto.

### 4.1.1. Fuentes de datos

Teniendo en cuenta que nuestro Data Lake contiene varias fuentes, vamos a detallar la información de cada una de forma individual:

- **Sensor de caudal:** Contiene la información del consumos medido por los sensores de la red de distribución de agua que vamos a estudiar. A partir de aquí nos vamos a referir al sensor de caudal como *sensorC*.

| Ind | Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|---|
| 0 | hour | String | 5 caracteres | Hora del día que registra el sensor. Es cada 10 minutos |
| 1 | forescast | float | | Predicción de consumo de agua $m^3/h$ |
| 2 | measured | float | | Consumo de agua real en $m^3/h$ |

**Tabla 4.1:** Detalles de los datos de consumo

El campo *forescast* que es el valor de la predicción de consumo, calculada por algoritmos de la empresa que nos facilitó las fuentes, no será incluido en los datos refinados ni se tendrá en cuenta para el estudio o comparaciones con nuestros resultados predictivos, por ser información confidencial.
La fuente en bruto del sensor de caudal contiene encabezado y muestra a continuación:

```
‘‘Category’’,‘‘demand forecast’’, ‘‘consumption’’
"00:00",1255.361343,1255.3611290000001
"00:10",976.8666650000001,976.8671572000001
```

- **Sensor de presión:** Contiene la información de la presión medida por sensores de la red de distribución de agua que vamos a estudiar. A partir de aquí nos vamos a referir al sensor de caudal como *sensorP*.

| Ind | Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|---|
| 0 | hour | String | 5 caracteres | Hora del día que registra el sensor. Es cada 10 minutos |
| 1 | measured | float | | Presión de agua, medida en "bar" |

**Tabla 4.2:** Detalles de los datos de presión

La fuente en bruto del sensor de presión contiene encabezado y se muestra a continuación:

```
‘‘Category’’,‘‘Measured pressure’’
"00:00",4.8995
"00:10",4.9334
```

- **Holiday:** Contiene las fechas que son días festivos del año 2015 para el país donde se encuentra la red de distribución de agua.

| Ind | Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|---|
| 0 | date | String | 10 caracteres | Fecha del calendario de 2015 |

**Tabla 4.3:** Información de días festivos de 2015

La fuente en bruto de holiday es un archivo plano con el listado de fechas como se muestra a continuación:

```
2015-01-01
2015-01-06
2015-03-03
```

### 4.1.2. Modelo conceptual

Evaluando los datos contenidos en cada una de las fuentes se identificó que todos eran relevantes para atender los requisitos, excepto la medida del consumo simulada, ya que es el dato resultado de la predicción descrita en el Capítulo 5. Adicional a los datos incluidos en cada fuente, para los consumos y presiones, se incluyó la fecha de las medidas de los sensores, que se pueden extraer del nombre de cada archivo.

A continuación se muestra el diagrama entidad relación generado a partir de los datos refinados obtenidos desde los datos fuente.


**Figura 4.1:** Entidad de los datos refinados

### 4.1.3. Datos refinados

El mapa lógico de datos es la base para la construcción de las actividades que conforman el proceso ELT. Contiene información fundamental para la obtención de metadatos y para la gestión de calidad [20].

La Tabla 4.4 muestra la transición de los datos desde las fuentes de origen, descritas en el Apartado 4.1, hasta los datos refinados. Todos los datos son leídos como tipo String desde la fuente y son convertidos a cada uno de los tipos correspondientes. Igualmente vemos que a partir de las fuentes se han extraído datos adicionales importantes, que se tienen en cuenta para generar el archivo de salida de datos refinados.

| **Origen** | | | **Transformación** | **Destino** | | |
|---|---|---|---|---|---|---|
| **Fuente** | **Atributo** | **Tipo** | | **Fuente** | **Atributo** | **Tipo** |
| sensorC | hour | String | Eliminar comillas, dividir el String en hora/minutos para obtener unicamente horas. | Consumption | hour | Int |
| | measured | String | Convertir en double | | demand | Double |
| | date | String | Convertir fecha + hora en timestamp | | time | BigInt |
| | | | A partir de los nombres de los archivos diarios obtener la fecha | | date | String |
| | | | Dividir el String de la fecha y obtener el mes | | month | Int |
| | | | Dividir el String de la fecha y obtener el día | | day | Int |
| | | | Asignar la estación del año correspondiente de acuerdo a la fecha | | season | Int |
| sensorP | measured | String | Convertir en double | | pressured | Double |
| holiday | date | String | De acuerdo a la fecha se identifican los fines de semana y los días laborales. ademas se integran con los festivos | | workingday | Int |

**Tabla 4.4:** Mapa lógico de datos que relaciona los datos en bruto con los datos refinados

Es importante aclarar, que el campo *date* relacionado dentro de los atributos de la fuente original para el sensorC, no viene incluido de forma explícita, pero se extrae del nombre del archivo, por ello lo hemos asociado a esta fuente. Además en la Tabla 4.4 se describen de forma muy general las transformaciones que se deben realizar a los datos en bruto para obtener el dato refinado, dichas transformaciones se explican en detalle en el Apartado 4.4.

## 4.2. Ingesta de Datos

El segundo componente de la construcción del DL, definido en la arquiectura DL (Figura 3.3) es la ingesta de datos. Como se mencionó en el Apartado 3.1, la información de los sensores, está contenida en dos carpetas independientes, una por cada sensor, las cuales contienen los datos distribuidos en archivos .csv, cuyo nombre es la fecha de medición del sensor.

Para la ingestión estás dos fuentes de datos, se hizo uso del servicio *Apache Flume*, que hace parte del ecosistema *Hadoop*. En la Figura 4.2 se muestra la arquitectura del agente que se configuró para tal fin.

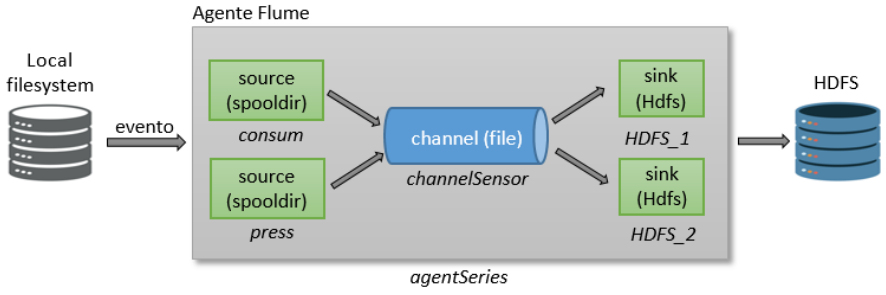


**Figura 4.2:** Arquitectura del agente flume. (Fuente: elaboración propia)

Las propiedades de configuración del agente "agentSeries", están registradas en el archivo spooldir.conf y se describen a continuación:

- **Propiedades de la fuente:** Teniendo en cuenta que nuestros datos se encuentran almacenados en el sistema de archivos local, se utilizó el tipo de fuente spooldir.

  Para este proyecto fue necesario usar dos *"source"*, debido a que las fuentes de datos se encuentran almacenadas, en disco, en rutas diferentes porque el nombre de los archivos csv de cada sensor es el mismo, se diferencian unicamente por el nombre de la carpeta que los contiene.

- **Propiedades del canal:** El tipo de canal que utilizamos es "file" que permite leer, escribir, mapear y manipular un archivo. En la configuración se declaran las rutas que se utilizaron para almacenar los archivos de registro y checkpoint, que deben estar creadas antes de iniciar la ingesta.

- **Propiedades del sumidero:** El tipo de sumidero que se definió en este proyecto fue HDFS y al igual que la configuración de las fuentes, para el sumidero también se requiere contar con dos diferentes para poder configurar dos rutas del directorio HDFS donde se descargaran los archivos correspondientes a cada sensor. Dentro

de está configuración se desactivó la funcionalidad de renovación porque queremos mantener los archivos originales y el formato de archivo es DataStream para que no comprima el archivo de salida. El esquema de almacenamiento en HDFS descrito en el Apartado 4.3.

El archivo *spooldir.conf* se guardó en el almacén de configuraciones de Flume (/etc/flume-ng/conf) y se muestra en el Anexo A.

Ejecución del agente: Para ejecutar el agente Flume debemos tener en cuenta la ubicación del archivo de configuración.

```
$ sudo flume-ng agent -n agentSeries -c /etc/flume-ng/conf -f spooldir.conf
```

## 4.3. Almacenamiento

El tercer componente de la construcción del DL, definido en la arquiectura DL (Figura 3.3) es el almacenamiento. Todos los datos de este proyecto fueron almacenados en HDFS, con el objetivo de ser accesibles a todas las herramientas utilizadas en el DL y que fueron descritas en el Apartado 3.4. Para el manejo de los datos de los sensores se crearon particiones dinámicas, por lo tanto dentro de cada transformación se generan subdirectorios por cada día del año. El esquema de almacenamiento HDFS se muestra en la Figura 4.3 donde se visualiza claramente la estructura de directorios.

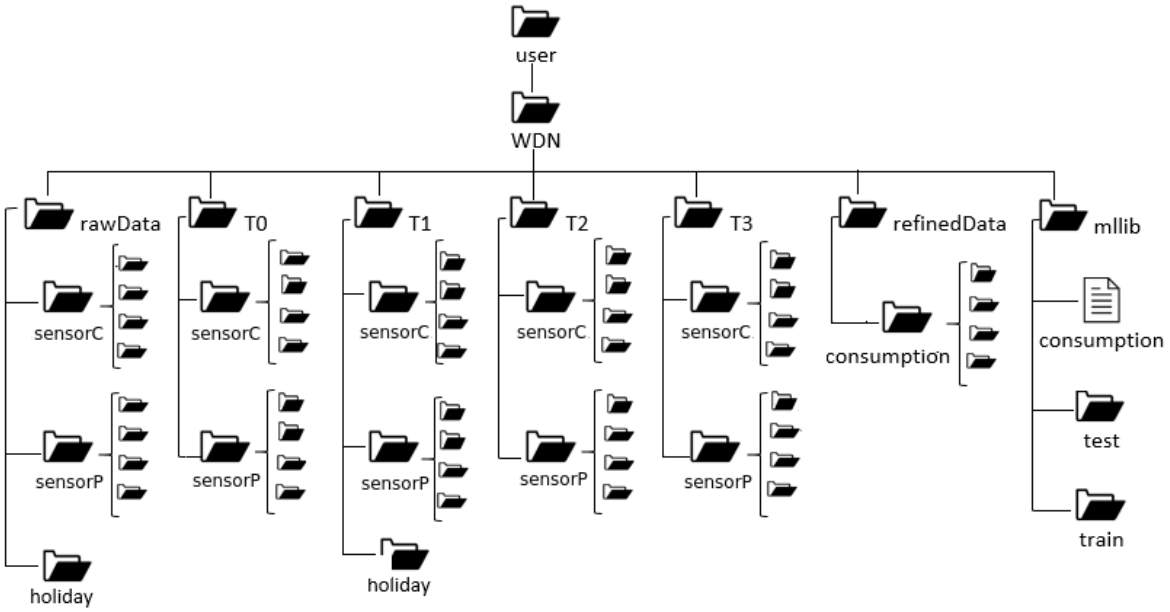


**Figura 4.3:** Esquema de Almacenamiento HDFS

- **rawData:** Esta carpeta contiene los datos en bruto del DL, es decir, las series temporales de los 2 sensores y el archivo del calendario festivo 2015. Las series temporales a su vez, contienen cada una un subdirectorio independiente para cada día de captura, con los ficheros *csv* descargados con *Flume*. La información almacenada en este directorio esta asociado a las tablas en *Hive*: load_s1, load_s2 y holiday.

- **T0:** Este directorio contiene la salida del proceso de interpolación lineal generado con Apache Spark. La salida es generada por cada sensor en rutas diferentes, por ello vemos en el esquema directorio independiente para cada sensor. Igualmente la salida generó un subdirectorio por cada día procesado.

- **T1**: Esta carpeta almacena la transformación 1 donde se extrae la fecha del nombre de cada directorio, dicha transformación se describen con detalle en la sección 4.4.2. Contiene las tablas *Hive* sensorC, sensorP y holiday descritos en la Figura 4.4.

- **T2**: Esta carpeta almacena la transformación 2 donde se incluye el día de la semana obtenido de la fecha, dicha transformación se describen con detalle en la sección 4.4.3. Contiene las tablas *Hive* sensorC_t2 y sensorP_t2 descritos en la Figura 4.4.

- **T3**: Esta carpeta almacena la transformación 3 donde se incluye el campo de los días festivos de la fuente de datos *holiday*, dicha transformación se describen con detalle en la sección 4.4.4. Contiene las tablas *Hive* sensorC_t3 y sensorP_t3 descritos en la Figura 4.4.

- **refinedData:** Este directorio almacena los datos ya refinados para su explotación, donde transformamos los datos de acuerdo a nuestro mapa lógico. Contiene la información de la tabla *Hive* consumption descrita en la Figura 4.4.

- **mllib**: En esta carpeta almacenamos un único archivo sin particiones, con toda la información necesaria para generar los modelos de aprendizaje. También se almacena allí los conjuntos de datos de entrenamiento y prueba.

## 4.4. Preprocesamiento y Transformación

El cuarto componente de la construcción del DL, definido en la arquiectura DL (Figura 3.3) es el preprocesamiento y la transformación de los datos. En este apartado se describe la forma como se realizó la preparación de los datos, y su transformación con base en el mapa lógico de datos definido. Para ello, ha sido necesario diseñar un workflow Oozie (Apéndice B), con varias etapas de procesamiento desplegadas en Spark y Hive. Además con antelación se crearon las 10 tablas que se necesitaron para las transformaciones en Hive (Apéndice D).

Como muestra la Figura 4.4 la limpieza y las transformaciones 1, 2 y 3 se realizan tanto para el sensor de caudal como para el de presión. Por ello, a continuación se describe en detalle dichas transformaciones para el sensorC, teniendo en cuenta que se realiza el mismo proceso para el sensorP.

Aunque la calidad de los datos de los sensores es buena, hay algunas lecturas que faltan, principalmente debido a problemas de comunicación. Con excepción de esta situación, en este trabajo, hemos supuesto que los datos recopilados son válidos y por lo tanto no se analizaron las redundancias, inconsistencias, ruido, y outliers. Para el tema de los valores nulos, se aplicó una interpolación lineal, ya que faltan unas pocas muestras y se consideró que con este método de interpolación es suficiente.

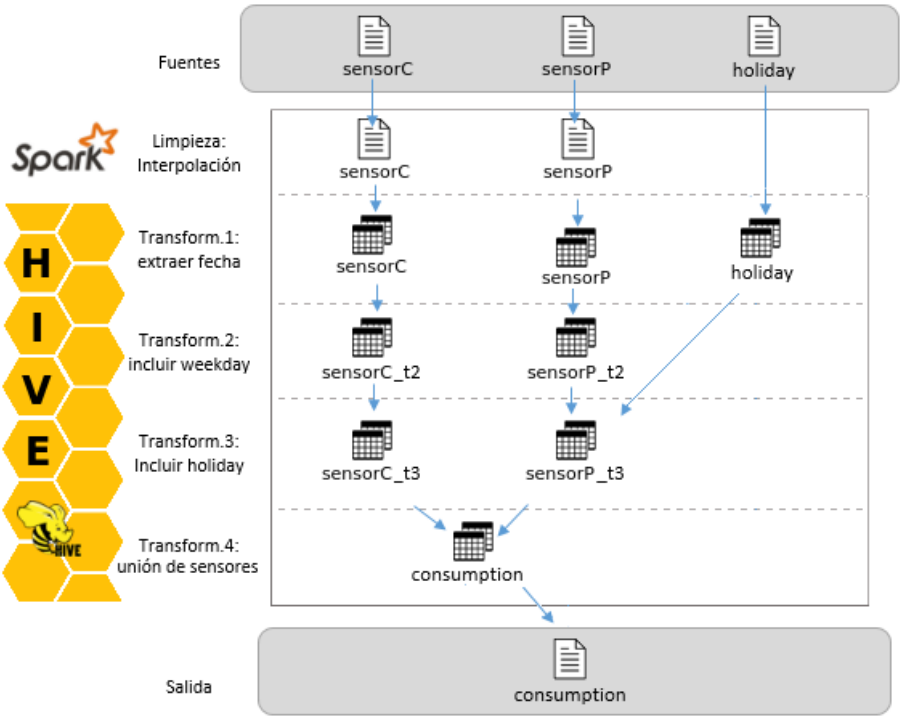


**Figura 4.4:** Esquema del preprocesamiento de datos. (Fuente: elaboración propia)

### 4.4.1. Limpieza: interpolación

La limpieza de datos es la primera fase del preprocesado de los datos de entrada. El proceso de interpolación se realizó a cada sensor de forma independiente, ya que la ruta y los atributos son diferentes. A causa de que los datos se encuentran almacenados en directorios diarios, la primera tarea fue leer todos los subdirectorios de cada ruta del sensor y guardarlos como una lista para que la función de interpolación pudiera obtener el vector correspondiente a cada día.

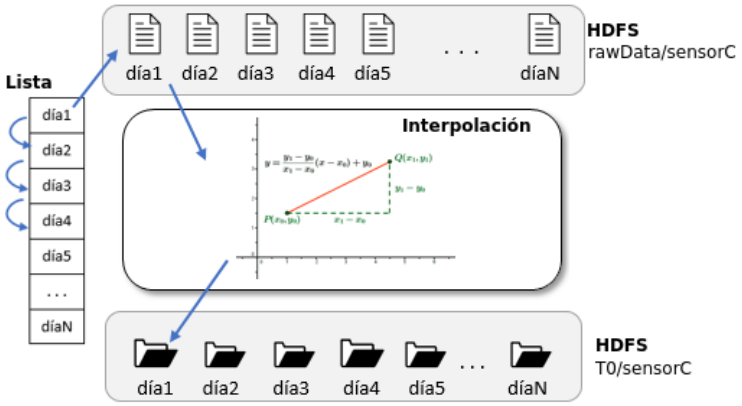


**Figura 4.5:** Representación del bucle para el proceso de limpieza de series. (Fuente: elaboración propia)

Como muestra la Figura 4.5, el proceso de interpolación es un bucle que recorre la lista de fechas diaria, generada con anterioridad. Después toma la serie temporal del día correspondiente (almacenada en HDFS en el directorio rawData), ejecuta la función de

interpolación y devuelve el resultado (serie completa) dentro de un directorio de salida que es llamado igual que el archivo fuente.

Para este proceso se hizo uso de la biblioteca Breeze/ScalaNPL. Breeze es el conjunto principal de bibliotecas para ScalaNLP, que incluye álgebra lineal, computación numérica y optimización. Permite un enfoque genérico, potente pero aún eficiente para el aprendizaje automático [3]. La función que se usamos para este proyecto es *breeze.interpolation* que sólo admite datos en una dimensión (interpolación univariante). Esta función dispone de interpolación lineal y cúbica, en nuestro caso se ajusta la interpolación lineal.

Los interpoladores 1D (univariado) obtienen las coordenadas de los nodos, para ello se requiere un vector para cada coordenada [4]. El interpolador de Breeze recibe 2 vectores $x$ e $y$. Para el vector $x$ se tomaron números consecutivos de 0 a 143 que corresponde a las 144 observaciones diarias (1 observación cada 10 minutos por 24 horas) y para el vector $y$ el valor medido del sensor que puede tener 144 observaciones o menos, en caso de que contenga nulos.

Es importante tener claro que los 2 vectores deben tener el mismo número de elementos, porque cada valor de $x$ corresponde a un valor en $y$. Para obtener los 2 vectores (DenseVector en Breeze) se siguieron los siguientes pasos:

1. Para crear el DenseVector $x$ se toman los valores medidos del sensor, que están ordenados por el tiempo de cada observación y se obtienen las posiciones de los valores no nulos, con ayuda de un indice. Como se mencionó anteriormente dichas posiciones van desde 0 a 143.

2. Para crear el DenseVector $y$ sencillamente se eliminan los valores nulos de la serie (si los hay).

Finalmente aplicamos el interpolador lineal con los 2 vectores obtenidos, y se solicitamos el resultado para un nuevo vector completo, con valores de 0 a 143, obteniendo como resultado un vector 144 observaciones. Este proceso se realiza por cada día del año y por cada sensor, la salida de la función, se almacena en nuestra ruta HDFS en un directorio por cada sensor y por cada día.

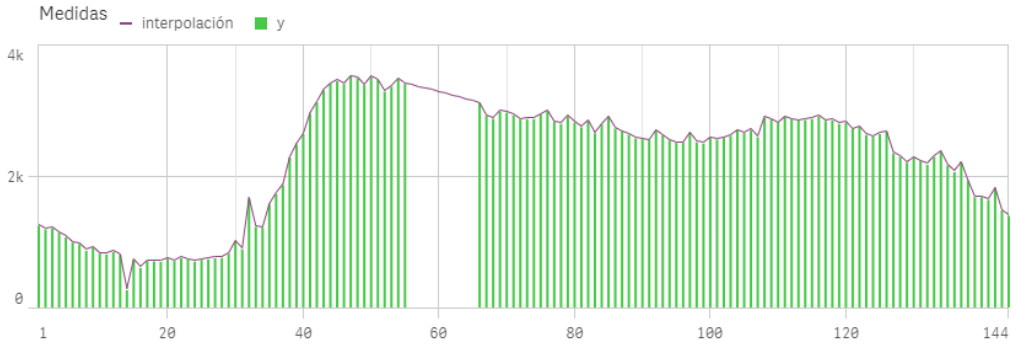


**Figura 4.6:** Gráfica del vector resultante de la interpolación lineal para el día 2015-02-03.

La Figura 4.6 muestra la gráfica del vector $y$ obtenido a partir de la fuente de datos del sensorC del día 3 de febrero (color verde), en comparación con el vector resultado de proceso de interpolación (color violeta). Aquí se aprecia claramente cómo el proceso de interpolación, completa la serie temporal de lo valores nulos.

El código *Scala* de este proceso se puede consultar en el Apéndice C. La salida de la interpolación genera un archivo por cada día. Cada observación es de tipo <clave,valor>, donde la clave es un String que corresponde al tiempo en hora:minutos y el valor esta compuesto por 2 Doubles correspondientes a la predicción y al valor medido. A continuación se muestra un ejemplo de los datos para el sensorC.

```
("00:00",(1255.361343,1255.3611290000001))
("00:10",(976.8666650000001,976.8671572000001))
("00:20",(1102.599583,1102.5996900000002))
```

### 4.4.2. Transformación 1: extraer fecha

Para cargar los datos de la salida de la interpolación a Hive fue necesario realizar una transformación, donde, además de eliminar los paréntesis, se extrajo la fecha del nombre de cada directorio, para ser incluida como una nueva columna y para usarla en las particiones de todas las tablas creadas en Hive.

En lenguaje HQL:

```
INSERT OVERWRITE TABLE sensorC PARTITION (date)
SELECT
  regexp_replace(hour,'\\(','') AS hour,
  regexp_replace(measured,'\\)','') AS measured,
  REGEXP_EXTRACT(INPUT__FILE__NAME, '.*/(.*)/(.*)', 1) AS date
FROM load_s1;
```

| Ind | Campo | Tipo | Ejemplo |
|---|---|---|---|
| 0 | hour | String | "00:00" |
| 1 | measured | Double | 1255.361129 |
| 2 | date | String | 2015-01-01 |

**Tabla 4.5:** Estructura de la tabla Hive para la transformación 1

En la Tabla 4.5 se muestra un ejemplo de la transformación realizada a partir de los datos generados después de la interpolación. Como se puede ver, quitamos los paréntesis que conformaban la estructura <clave,valor>, obteniendo el campo *hour* que corresponde al tiempo HH:MM y el campo measured que corresponde al valor medido del sensor. En esta transformación se obvio el valor de la predicción porque al ser el resultado que se pretende alcanzar, no se incluye dentro de los datos refinados. Finalmente se incluye una nueva columna "date" que se obtiene a partir del nombre de la carpeta que contiene la serie.

### 4.4.3. Transformación 2: incluir weekday

Se requiere manejar el tiempo en formato POSIX [1] porque es mucho más fácil de analizar y más preciso, ademas se va a usar como llave para unir la información de los dos sensores. Para ello, se creamos una nueva columna concatenando fecha y hora, para hacer la conversión en una transformación posterior. Para comprobar las hipótesis de tendencia horaria, diaria y estacional, se considera que es necesario tener la información en campos independientes:

[1] Sistema para la descripción de instantes de tiempo: se define como la cantidad de segundos transcurridos desde la medianoche UTC del 1 de enero de 1970, sin contar segundos intercalares.

- Se crearon 2 nuevas columnas (*month, day*) donde se divide la fecha para extraer mes y día.
- En el campo *hour* (HH:MM) se eliminan las comillas.
- Se crea una nueva columna (*weekday*) que indica el día de la semana correspondiente a la fecha, donde 1 es lunes y 7 es domingo.

En lenguaje HQL:

```
INSERT OVERWRITE TABLE sensorC_T2 PARTITION (date)
SELECT
  (concat(concat(date,' '),SUBSTR(hour, 2, LENGTH(hour)-2))) AS time,
  split(‘date‘, '-')[1] AS month,
  split(‘date‘, '-')[2] AS day,
  SUBSTR(hour, 2, LENGTH(hour)-2) AS hour,
  date_format(date, 'u') AS weekday,
  measured,
  date
FROM sensorC;
```

| Ind | Campo | Tipo | Ejemplo |
|---|---|---|---|
| 0 | time | String | 2015-01-01 00:00 |
| 1 | month | String | 01 |
| 2 | day | String | 01 |
| 3 | hour | String | 00:00 |
| 4 | weekday | String | 4 |
| 5 | measured | Double | 1255.361129 |
| 6 | date | String | 2015-01-01 |

**Tabla 4.6:** Estructura de la tabla Hive para la transformación 2

En la Tabla 4.6 se muestra un ejemplo la transformación de los datos realizada en ésta sección. Como se puede ver, tenemos el campo *time* que contiene la fecha y la hora, *month* correspondiente al mes que se extrajo del campo *date*, *day* correspondiente al día que se extrajo del campo *date*, *hour* que corresponde a la hora de la observación, aquí sencillamente eliminaron las comillas, *weekday* que corresponde al día jueves, *measured* y *date* datos que vienen igual desde la transformación anterior.

### 4.4.4. Transformación 3: incluir holiday

En esta transformación, se realizaron los siguientes ajustes:

- Se convirtió la fecha que está en formato ‘yyyy-MM-dd HH:mm’, a formato POSIX.
- Para *hour* se eliminaron los minutos, obteniendo unicamente las horas para convertirlas a un entero.
- Los datos *month* y *day* se convirtieron a un entero.
- Los días de la semana del campo *weekday* se transformaron en binario, donde 1 es fin de semana y 0 para días laborales. A partir de esta transformación el nombre del campo pasa a ser *workingday* que representa mejor el dato que contiene.
- Se incluyo una nueva columna binaria *holiday*, que compara las fechas registradas en el archivo holiday con las fechas de los sensores, cuando una fecha está contenida en holiday asigna 1 (día festivo) de lo contrario 0 (día laboral).

En Lenguaje HQL:

```
INSERT OVERWRITE TABLE sensorC_T3 PARTITION (date)
SELECT
  (UNIX_TIMESTAMP(time, 'yyyy-MM-dd HH:mm')) AS time,
  cast(month AS INT)AS month,
  cast(day AS INT)AS day,
  cast(split(hour, ':')[0] AS INT) AS hour,
  if(weekday > 5, 1, 0) AS workingday,
  if (holiday IS NULL,0,1) as holiday,
  measured,
  date
FROM sensorC_t2
LEFT OUTER JOIN holiday
ON (date = holiday);
```

| Ind | Campo | Tipo | Ejemplo |
|---|---|---|---|
| 0 | time | Bigint | 1420099200 |
| 1 | month | Int | 1 |
| 2 | day | Int | 1 |
| 3 | hour | Int | 0 |
| 4 | workingday | Int | 0 |
| 5 | holiday | Int | 1 |
| 6 | measured | Double | 1255.361129 |
| 7 | date | String | 2015-01-01 |

**Tabla 4.7:** Estructura de la tabla Hive para la transformación 3

Como vemos en la Tabla 4.7 el campo *time* (2015-01-01 00:00) fue convertido a formato POSIX, los campos *month* y *day* están como entero, para el campo *hour* (00:00) sólo se dejó la hora eliminando los minutos, el campo *weekday* (4) correspondiente a día jueves, fue convertido a *workingday* donde pasó a ser día laboral (0), en los campos *measured* y *date* los datos vienen igual que la transformación anterior.

### 4.4.5. Transformación 4: unión de sensores

Aquí se hace la unión de los 2 sensores, donde la información que contiene el sensorC es la demanda y la del sensorP es la presión de agua, combinados a través de la fecha y la hora que convertimos en POSIX en la transformación anterior.

Además unimos las columnas binarias *workingday* (0 para laboral, 1 para fin de semana) y *holiday* (1 para festivo) para identificar fácilmente los días laborales de los que no lo son.

Finalmente se incluyo la información de las estación del año, creando una columna que compara la fecha de la observación con el rango de fechas de cada estación y asigna el número correspondiente de 1 a 4:

1 Primavera, 2 Verano, 3 Otoño, 4 Invierno.

En lenguaje HQL:

```
INSERT OVERWRITE TABLE consumption PARTITION (date)
SELECT
s1.time,s1.month,s1.day,s1.hour,
if(s1.holiday=1, 1, if(s1.workingday=1, 1, 0)) AS workingday,
case
when concat(split(s1.date, '-')[0],'-01-01')<= s1.date and s1.date <= concat(split(
    s1.date, '-')[0],'-03-20') then 4
when concat(split(s1.date, '-')[0],'-03-21')<= s1.date and s1.date <= concat(split(
    s1.date, '-')[0],'-06-20') then 1
```

```
when concat(split(s1.date, '-')[0],'-06-21')<= s1.date and s1.date <= concat(split(
    s1.date, '-')[0],'-09-22') then 2
when concat(split(s1.date, '-')[0],'-09-23')<= s1.date and s1.date <= concat(split(
    s1.date, '-')[0],'-12-20') then 3
when concat(split(s1.date, '-')[0],'-12-21')<= s1.date and s1.date <= concat(split(
    s1.date, '-')[0],'-12-31') then 4
else 0 end as season,
s1.measured as demand,
s2.measured as pressure,
s1.date
FROM
sensorC_T3 s1
JOIN sensorP_T3 s2 ON (s1.time = s2.time);
```

| Ind | Campo | Tipo | Ejemplo |
|---|---|---|---|
| 0 | time | Bigint | 1420099200 |
| 1 | month | Int | 1 |
| 2 | day | Int | 1 |
| 3 | hour | Int | 0 |
| 4 | workingday | Int | 1 |
| 5 | season | Int | 4 |
| 6 | demand | Double | 1255.361129 |
| 7 | pressure | Double | 3.2663 |
| 8 | date | String | 2015-01-01 |

**Tabla 4.8:** Estructura de la tabla Hive para la transformación 4

En la Tabla 4.8 se muestra un ejemplo de la transformación de los datos realizada en esta sección. Como se puede ver, tenemos los campos *time*, *month*, *day* y *hour* que continúan igual que la transformación anterior. Para el campo *workingday* y *holiday*, en este ejemplo en concreto, se presentaba una incoherencia: el 1 de enero para el campo *workingday* era día laboral pero para el campo *holiday* era un día festivo, al integrar estas 2 columnas se soluciona el inconveniente ya que pasa de ser 0 (día laboral) a 1 y queda eliminado el campo *holiday*. Se incluye la columna *season* que es 4 correspondiente a invierno. En vista que los 2 sensores tienen el campo *measured*, en esta transformación se cambiaron los nombres del valor medido del sensor para diferenciar la demanda de la presión, por tanto tenemos los campos *demand* correspondiente al sensorC y *pressure* correspondiente al sensorP y el campo *date* que no tiene cambios.

### 4.4.6. Salida: de tabla a archivo plano

La información contenida en la Tabla 4.8, se encuentra almacenada en particiones de fecha, es decir, los 52.560 registros están divididos en 365 directorios que corresponden al número de días del año. En vista de la necesidad de tener un sólo archivo de lectura, para el proceso de aprendizaje, se realizó exportación a un archivo de texto que también se almacenó en HDFS.

En Lenguaje HQL:

```
INSERT OVERWRITE DIRECTORY '/user/WDN/mllib'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
SELECT * FROM consumption;
```

El archivo de salida tipo texto, contiene los mismos datos de la Tabla 4.8 y se visualizan de la siguiente forma:

```
1420099200;1;1;0;1;4;1255.3611290000001;3.2663333333333333;2015-01-01
1420099800;1;1;0;1;4;976.8671572000001;3.288933333333333;2015-01-01
1420100400;1;1;0;1;4;1102.5996900000002;3.2685333333333335;2015-01-01
```

Es importante tener en cuenta, que la salida de la transformación 5, está diseñada exclusivamente para el uso de los algoritmos ML y MLLib descritos en el Capítulo 5, si se requiere otro tipo de estructura, como por ejemplo, implementar una red neuronal LSTM que se propone en el trabajo futuro del Capítulo 7, se puede crear una nueva estructura a partir de los datos refinados almacenados en la carpeta refinedData.

# Capítulo 5

# Modelos de Aprendizaje

*"La predicción es muy difícil, especialmente si se trata del futuro"*

Nils Bohr (Premio Nobel de Física)

**Resumen:** A menudo es fácil encontrar un modelo que se adapte bien a los datos del pasado, pero otro asunto bastante importante es encontrar un modelo que identifique correctamente las características de los datos del pasado que se replicarán en futuro. Este capítulo describe el proceso de modelado para extraer los patrones del consumo de agua en la WDN. Los datos se entrenan usando 3 algoritmos de regresión disponibles en la biblioteca ML de Spark, Decision Tree, Random Forest y Gradient Boosted Tree y se evalúan en los conjuntos de prueba con los parámetros óptimos obtenidos con validación cruzada.

## 5.1. Introducción

El modelado predictivo es una representación de la realidad basada en un intento descriptivo de relacionar un conjunto de variables con otro. Predecir el consumo en una red de distribución de agua, donde la salida debe ser un valor continuo para una fecha determinada y los datos de entrada están ordenados en el tiempo, podemos asegurar que se trata de un problema de predicción de series temporales. Como se ha presentado en el Capítulo 2, los métodos propuestos que se utilizarán en este trabajo son los métodos no lineales basados en árboles de decisión. Compararemos los resultados de métodos que generan un único árbol (Decision Tree) o técnicas ensemble que generan muchos árboles, como son los métodos Gradient-Boosted Tree (GBT) y Random Forest. Los métodos basados en árboles se han propuesto principalmente porque sus modelos son fáciles de interpretar y las técnicas de ensemble normalmente mejoran los resultados obtenidos por un único árbol, además de obtener muy buenos resultados en numerosas aplicaciones reales.

Antes de iniciar con la evaluación de los modelos, se hará una exploración general de los datos.

## 5.2. Exploración dataset refinado

La estructura y descripción general de los atributos que contiene nuestro dataset, fueron explicados en la arquitectura de datos, Apartado 4.1. En esta sección se detallará como se distribuyen las observaciones del dataset obtenido, como resultado de las transformaciones, sus valores mínimos y máximos y la forma de relacionarse entre sí.

| Atributo | Tipo | Min | Max |
|---|---|---|---|
| **time** | timestamp | 1420099200 | 1451634600 |
| **month** | discreto | 1 | 12 |
| **day** | discreto | 1 | 31 |
| **hour** | discreto | 0 | 23 |
| **workingday** | binario | 0 | 1 |
| **season** | discreto | 1 | 4 |
| **demand** | real | 60.94 | 5137.20 |
| **pressure** | real | 2.02 | 3.70 |
| **date** | discreto | "2015-01-01" | "2015-12-31" |

**Tabla 5.1:** Rangos de cada atributo

Como se refleja en la Tabla 5.1, el conjunto de datos contiene atributos con valores discretos (month, hour, season), valores binarios (workingday) y valores reales (demand, pressure) y ninguno de los atributos contiene valores negativos.

A continuación se muestran varias imágenes que reflejan la distribución de las observaciones y las tendencias de consumo.

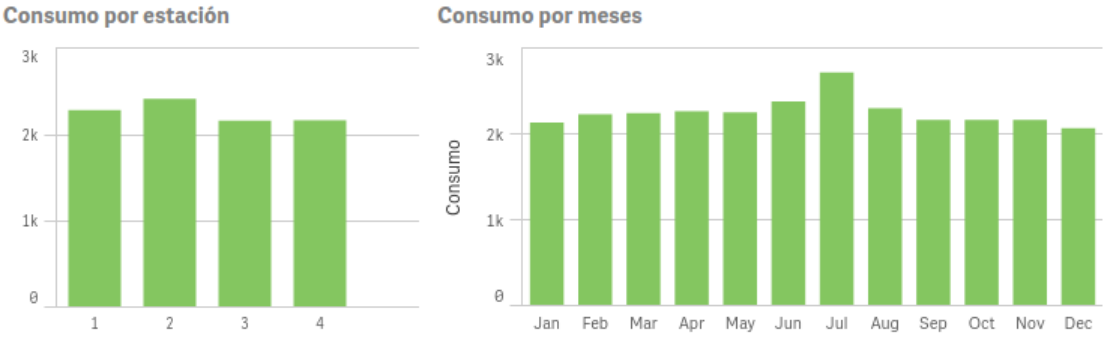


**Figura 5.1:** Tendencia de consumo por meses y por estaciones

Como se menciona en el Apartado 4.4.5, las estaciones del año están categorizadas de la 1 a la 4: 1 Primavera, 2 Verano, 3 Otoño, 4 Invierno. En la Figura 5.1 muestra mayor consumo en la época de verano, meses de junio, julio y agosto, aunque no podemos asegurar que la tendencia sea siempre así porque no se cuenta con datos históricos anuales.

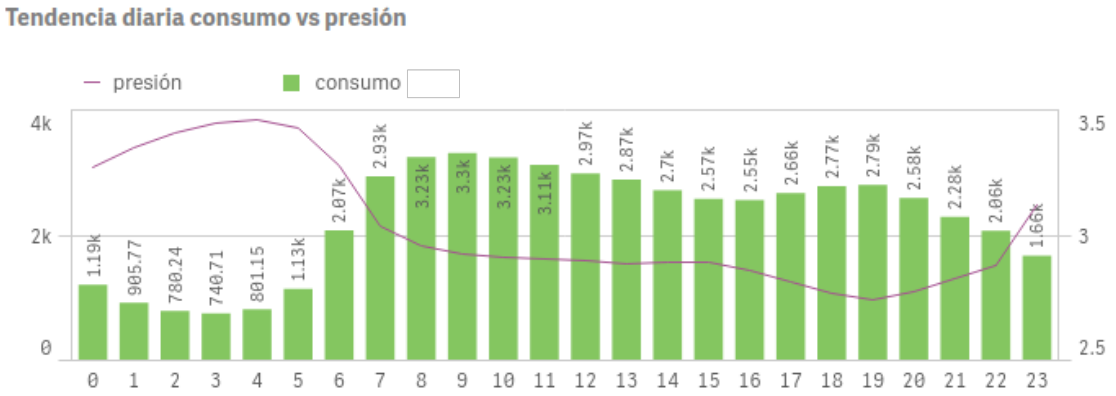


**Figura 5.2:** Relación de consumo con presión

La Figura 5.2 muestra la tendencia de consumo durante las 24 horas diarias, relacionada con los valores de presión, donde se refleja que son los 2 atributos inversamente proporcionales, es decir, a mayor consumo menor presión y a menor consumo mayor presión, por esta razón aseguramos que las 2 variables son dependientes.

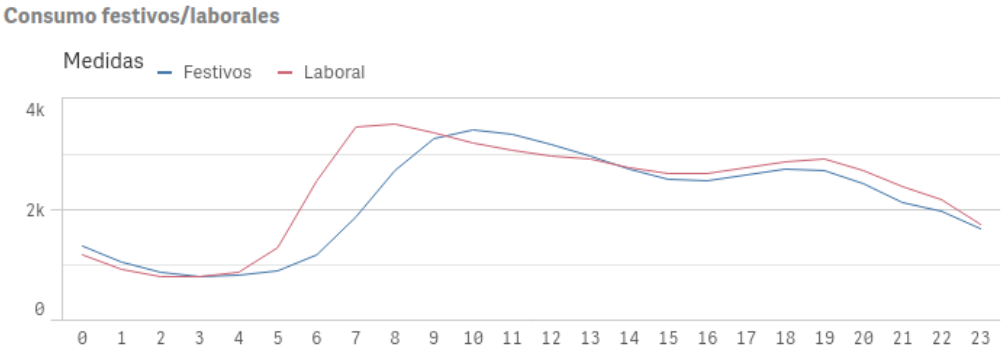


**Figura 5.3:** Tendencia de consumo en días laborales vs festivos

La comparación de consumo de los días laborales vs los días festivos (figura 5.3), nos demuestra que la tendencia de consumo es similar pero el horario de consumo varía al iniciar el día. En días laborales el consumo se empieza a incrementar a las 5:00, teniendo su máximo entre 7:00 y 8:00, mientras que en días festivos, el incremento del consumo inicia a las 6:00 con un consumo máximo a las 10:00. Al finalizar el día no muestra una diferencia significativa en el horario de días laborales vs festivos.

Finalmente en la figura 5.4, se muestra la tendencia del consumo en los 7 días de la semana, en la imagen se puede apreciar la estacionalidad que presenta la serie temporal, donde su frecuencia es diaria.

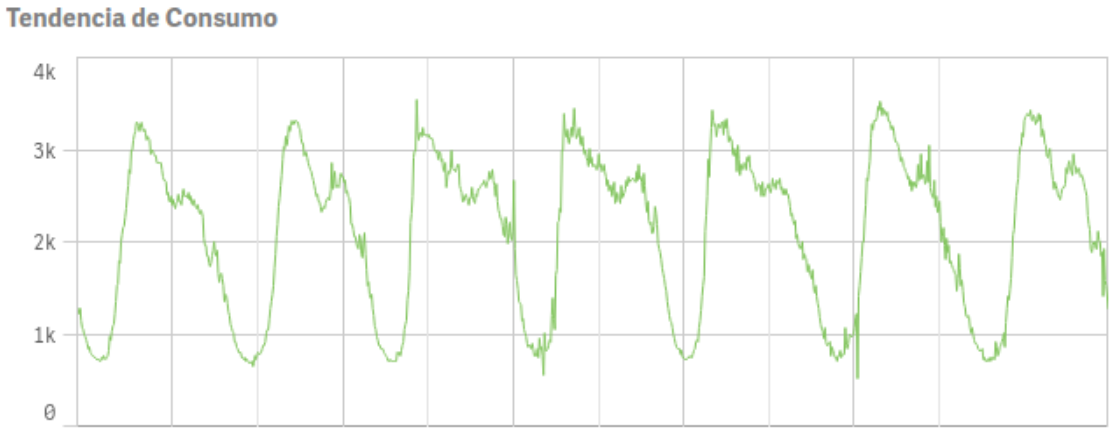


**Figura 5.4:** Tendencia de consumo semanal

## 5.3. Selección del modelo

Como se mencionó al iniciar este capítulo, vamos a evaluar 3 métodos de regresión disponibles en la biblioteca ML de Spark: Decision Tree, Random Forest y GBT. Se escogieron inicialmente estos 3 métodos, porque los árboles de decisión manejan características categóricas, no requieren escalado de características y son capaces de capturar interacciones no lineales e interactivas.

Antes de iniciar con ésta etapa, se dividió el dataset de forma aleatoria, para obtener el conjunto de entrenamiento y el de pruebas (80-20). Por lo tanto, en este apartado sólo se usó el conjunto de entrenamiento y el de pruebas se reservo para la Sección 5.4.

Para definir los parámetros de ajuste de estos métodos se usó la validación cruzada (Figura 5.5). Esta validación consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones o subconjuntos. La validación cruzada se utilizó para estimar la profundidad del árbol para Decision Tree, el número de árboles para Random Forest y el número de iteraciones (o número de árboles) para Gradient Boosted Tree, obteniendo los parámetros óptimos de cada modelo. Los datos de muestra se dividieron aleatoriamente en 10 subconjuntos, cada uno de ellos con datos de prueba y entrenamiento.

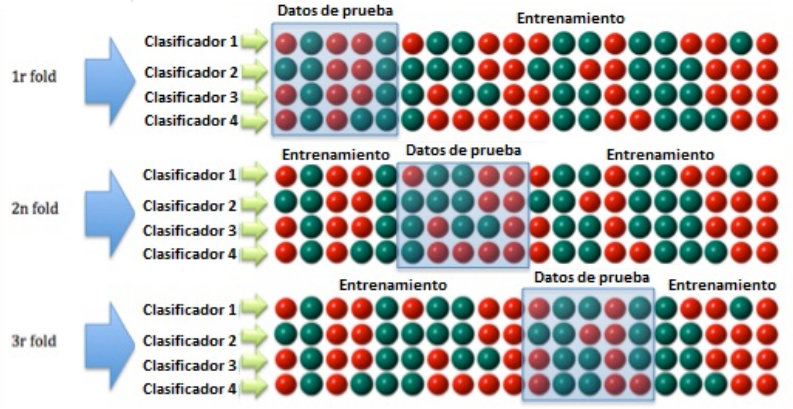


**Figura 5.5:** Representación de la validación cruzada (Wikipedia)

Hemos utilizado varias métricas de evaluación para comparar la precisión de las predicciones obtenidas por los diferentes métodos de predicción. A continuación describimos dichas métricas, donde $y_i$ es el valor real e $\hat{y}_i$ es el valor predicho.

1. **Mean Squared Error (MSE):** el error cuadrático medio mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima. Su definición es:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. **Root Mean Squared Error (RMSE):** tomando la raíz cuadrada del MSE produce el error de la raíz cuadrada de la media o la desviación de la raíz cuadrada media. Se puede interpretar como la desviación estándar de la varianza inexplicada, y tiene la propiedad útil de estar en las mismas unidades que la variable de respuesta. Los valores más bajos de RMSE indican un mejor ajuste. Esta definida como:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3. **Mean Absolute Error (MAE):** el error absoluto medio es una medida de la diferencia entre dos variables continuas (valor real vs. valor predicho). Se define como:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

4. **Coeficiente de determinación ($R^2$):** se define como la proporción de la varianza total de la variable explicada por la regresión, refleja la bondad del ajuste de un modelo a la variable que pretender explicar. El resultado del R Cuadrado oscila entre 0 y 1. Cuanto más cerca de 1 se sitúe su valor, mayor será el ajuste del modelo a la variable que estamos intentando explicar. Esta definida como:

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

La Tabla 5.2 muestra los resultados obtenidos al aplicar un árbol de regresión (Decision Tree) usando el método conocido en MLlib como DecisionTreeRegressor. Este método requiere especificar la profundidad máxima del árbol, maxDepth, que se ha establecido en profundidad 4 y 5. De esta forma, se han obtenido 20 modelos de predicción. La configuración óptima se ha obtenido con una profundidad de 5, obteniendo un $R^2$ promedio de 0,8525.

**Tabla 5.2:** Resultados del modelo Decision Tree

| k_fold | maxDepth | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 1 | 4 | 379,094 | 143712,111 | 240,664 | 0,838 |
| 1 | 5 | 366,070 | 134007,226 | 239,540 | 0,849 |
| 2 | 4 | 378,554 | 143302,760 | 246,760 | 0,835 |
| 2 | 5 | 361,437 | 130636,670 | 243,040 | 0,849 |
| 3 | 4 | 368,738 | 135967,640 | 242,187 | 0,843 |
| 3 | 5 | 353,821 | 125188,980 | 237,960 | 0,856 |
| 4 | 4 | 381,864 | 145820,492 | 247,857 | 0,835 |
| 4 | 5 | 365,213 | 133380,552 | 243,418 | 0,849 |
| 5 | 4 | 372,275 | 138588,996 | 242,315 | 0,843 |
| 5 | 5 | 357,104 | 127523,160 | 240,402 | 0,856 |
| 6 | 4 | 372,357 | 138649,693 | 241,886 | 0,843 |
| 6 | 5 | 357,007 | 127454,355 | 238,428 | 0,856 |

Sigue en la página siguiente.

| k_fold | maxDepth | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 7 | 4 | 373,569 | 139553,512 | 241,867 | 0,839 |
| 7 | 5 | 356,337 | 126975,924 | 236,747 | 0,853 |
| 8 | 4 | 385,057 | 148268,527 | 247,585 | 0,832 |
| 8 | 5 | 369,448 | 136491,850 | 245,066 | 0,845 |
| 9 | 4 | 368,768 | 135989,504 | 238,441 | 0,846 |
| 9 | 5 | 352,319 | 124128,328 | 235,528 | 0,859 |
| 10 | 4 | 376,459 | 141721,451 | 245,032 | 0,839 |
| 10 | 5 | 361,253 | 130503,630 | 243,625 | 0,851 |

La Figura 5.6 muestra la comparación del coeficiente de determinación para los 10 subconjuntos con el método Decisión Tree, con una profundidad del árbol de 4 y 5. Donde la diferencia entre los resultados entre las dos profundidades en mínima, aunque la profundidad 5 con un mejor resultado de predicción. Cabe mencionar que la escala del gráfico está en un rango de [0,6 a 0,9] para poder apreciar las diferencias, ya que son mínimas y con escala completa no se puede visualizan.

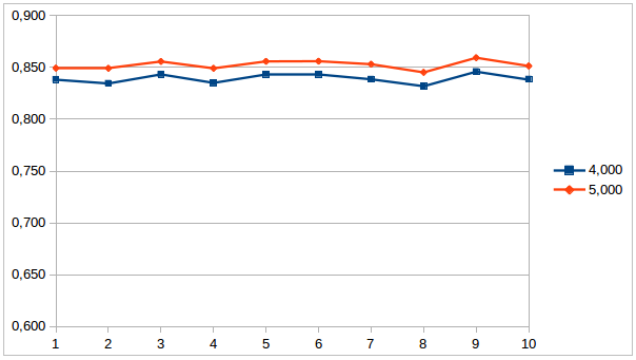


**Figura 5.6:** Gráfico del coeficiente de determinación para Decision Tree

La Tabla 5.3 muestra los resultados obtenidos al aplicar la técnica ensemble Random Forest, conocida como RandomForestRegressor en MLlib. Estos parámetros son el número de árboles a entrenar, habiéndose considerado 50, 60, 70, 80, 90 y 100 árboles; y la profundidad máxima del árbol, maxDepth, que se ha establecido en profundidad 5. Finalmente, se han obtenido 60 modelos, donde se puede observar que el mejor promedio de $R^2$ 0,8719 se obtiene con 60 árboles.

**Tabla 5.3:** Resultados del modelo Random Forest

| k-fold | NumTree | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 1 | 50 | 349,431 | 122101,866 | 255,431 | 0,863 |
| 1 | 60 | 345,710 | 119515,120 | 249,933 | 0,866 |
| 1 | 70 | 345,549 | 119404,133 | 248,621 | 0,866 |
| 1 | 80 | 349,714 | 122300,084 | 253,172 | 0,862 |
| 1 | 90 | 349,218 | 121953,557 | 251,521 | 0,863 |
| 1 | 100 | 349,131 | 121892,467 | 250,943 | 0,863 |
| 2 | 50 | 344,969 | 119003,872 | 264,540 | 0,863 |
| 2 | 60 | 341,323 | 116501,605 | 257,956 | 0,865 |
| 2 | 70 | 335,888 | 112820,908 | 250,972 | 0,870 |
| 2 | 80 | 340,671 | 116056,654 | 259,440 | 0,866 |
| 2 | 90 | 344,859 | 118927,948 | 260,243 | 0,863 |
| 2 | 100 | 340,016 | 115611,083 | 256,750 | 0,866 |
| 3 | 50 | 343,640 | 118088,363 | 261,365 | 0,864 |
| 3 | 60 | 333,591 | 111283,259 | 250,993 | 0,872 |
| 3 | 70 | 333,215 | 111032,253 | 250,863 | 0,872 |
| 3 | 80 | 336,294 | 113093,541 | 253,748 | 0,870 |
| 3 | 90 | 335,959 | 112868,622 | 252,911 | 0,870 |
| 3 | 100 | 334,969 | 112204,050 | 250,427 | 0,871 |

Sigue en la página siguiente.

| k-fold | NumTree | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 4 | 50 | 355,095 | 126092,398 | 266,867 | 0,857 |
| 4 | 60 | 347,248 | 120581,094 | 256,277 | 0,864 |
| 4 | 70 | 351,069 | 123249,230 | 262,763 | 0,861 |
| 4 | 80 | 354,266 | 125504,106 | 265,986 | 0,858 |
| 4 | 90 | 348,773 | 121642,685 | 258,533 | 0,862 |
| 4 | 100 | 348,020 | 121118,162 | 257,291 | 0,863 |
| 5 | 50 | 336,420 | 113178,687 | 260,024 | 0,872 |
| 5 | 60 | 328,008 | 107589,143 | 249,805 | 0,878 |
| 5 | 70 | 331,023 | 109576,137 | 253,978 | 0,876 |
| 5 | 80 | 340,862 | 116187,116 | 262,366 | 0,869 |
| 5 | 90 | 333,140 | 110982,244 | 253,615 | 0,875 |
| 5 | 100 | 330,359 | 109137,061 | 250,741 | 0,877 |
| 6 | 50 | 348,593 | 121516,983 | 262,545 | 0,863 |
| 6 | 60 | 334,946 | 112188,656 | 248,858 | 0,873 |
| 6 | 70 | 337,786 | 114099,425 | 251,788 | 0,871 |
| 6 | 80 | 345,021 | 119039,454 | 259,030 | 0,865 |
| 6 | 90 | 344,056 | 118374,565 | 256,432 | 0,866 |
| 6 | 100 | 338,418 | 114526,664 | 251,616 | 0,871 |
| 7 | 50 | 336,781 | 113421,289 | 253,554 | 0,869 |
| 7 | 60 | 332,809 | 110761,662 | 250,147 | 0,872 |
| 7 | 70 | 331,584 | 109947,858 | 249,166 | 0,873 |
| 7 | 80 | 338,066 | 114288,664 | 255,268 | 0,868 |
| 7 | 90 | 338,207 | 114383,686 | 254,681 | 0,868 |
| 7 | 100 | 337,120 | 113649,868 | 253,764 | 0,869 |
| 8 | 50 | 343,567 | 118038,516 | 263,584 | 0,866 |
| 8 | 60 | 332,418 | 110501,764 | 249,775 | 0,875 |
| 8 | 70 | 337,009 | 113574,964 | 256,059 | 0,871 |
| 8 | 80 | 342,824 | 117528,382 | 262,553 | 0,867 |
| 8 | 90 | 340,775 | 116127,462 | 259,309 | 0,868 |
| 8 | 100 | 337,690 | 114034,260 | 255,366 | 0,871 |
| 9 | 50 | 332,897 | 110820,458 | 252,254 | 0,874 |
| 9 | 60 | 328,534 | 107934,649 | 245,710 | 0,878 |
| 9 | 70 | 328,010 | 107590,283 | 244,846 | 0,878 |
| 9 | 80 | 337,215 | 113713,977 | 253,915 | 0,871 |
| 9 | 90 | 337,561 | 113947,238 | 252,996 | 0,871 |
| 9 | 100 | 333,423 | 111170,699 | 250,054 | 0,874 |
| 10 | 50 | 343,525 | 118009,720 | 264,219 | 0,866 |
| 10 | 60 | 331,870 | 110137,919 | 250,255 | 0,875 |
| 10 | 70 | 330,561 | 109270,893 | 249,458 | 0,876 |
| 10 | 80 | 339,620 | 115342,059 | 258,100 | 0,869 |
| 10 | 90 | 340,953 | 116248,949 | 257,288 | 0,868 |
| 10 | 100 | 335,756 | 112732,053 | 253,485 | 0,872 |

La Figura 5.7 muestra la evolución $R^2$ de acuerdo al número de árboles definidos para el entrenamiento del método Random Forest, para los 10 subconjuntos de datos.

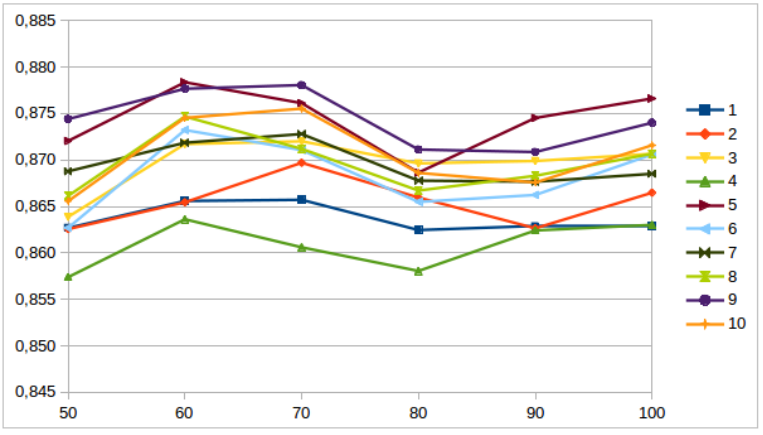


**Figura 5.7:** Gráfico del coeficiente de determinación para Random Forest

Es importante mencionar que la escala del gráfico está en un rango de [0,845 a 0,885] para poder apreciar las diferencias entre cada iteración, ya que son mínimas y con escala completa no se pueden visualizar. Se aprecia la diferencia de los resultados a medida que aumentamos el número de árboles. Como se mencionó anteriormente, el mejor resultado es el obtenido con 60 árboles y profundidad de 5, obteniendo un $R^2$ de 0,8719. Este resultado demuestra que no siempre, cuanto más árboles mejor es la predicción. Al parecer a partir de esta cantidad de árboles se genera sobreajuste en el modelo y genera peores predicciones.

Por último la Tabla 5.4 muestra los resultados obtenidos al aplicar la técnica ensemble GBT, conocida en MLlib como GradientBoostingRegressor. Al igual con RandomForest, los parámetros son el número de árboles a entrenar. Aquí consideramos 20, 50, 60, 70, 80, 90 y 100 árboles. Finalmente, se han obtenido 80 modelos, donde se puede observar que el mejor promedio de $R^2 = 0,9302$ se obtiene con 100 árboles. No obstante, como muestra la Figura 5.8 a partir de los 60 árboles, la mejora en la predicción es insignificante en contraste con el tiempo de entrenamiento empleado.

**Tabla 5.4:** Resultados del modelo Gradient Boost Tree

| k-fold | NumTree | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 1 | 20 | 280,245 | 78537,293 | 176,589 | 0,912 |
| 1 | 50 | 268,831 | 72270,223 | 162,946 | 0,919 |
| 1 | 60 | 267,909 | 71775,007 | 161,653 | 0,919 |
| 1 | 70 | 267,461 | 71535,301 | 161,115 | 0,920 |
| 1 | 80 | 267,331 | 71465,963 | 160,895 | 0,920 |
| 1 | 90 | 267,281 | 71438,926 | 160,771 | 0,920 |
| 1 | 100 | 267,113 | 71349,543 | 160,582 | 0,920 |
| 2 | 20 | 261,497 | 68380,840 | 175,390 | 0,921 |
| 2 | 50 | 244,409 | 59735,521 | 159,955 | 0,931 |
| 2 | 60 | 243,502 | 59293,001 | 159,169 | 0,932 |
| 2 | 70 | 242,843 | 58972,581 | 158,623 | 0,932 |
| 2 | 80 | 242,293 | 58705,656 | 158,236 | 0,932 |
| 2 | 90 | 241,910 | 58520,437 | 157,939 | 0,932 |
| 2 | 100 | 241,630 | 58385,213 | 157,739 | 0,933 |
| 3 | 20 | 259,204 | 67186,782 | 173,686 | 0,923 |
| 3 | 50 | 246,744 | 60882,412 | 160,301 | 0,930 |
| 3 | 60 | 246,139 | 60584,262 | 159,305 | 0,930 |
| 3 | 70 | 245,405 | 60223,557 | 158,668 | 0,931 |
| 3 | 80 | 245,018 | 60034,021 | 158,201 | 0,931 |
| 3 | 90 | 244,381 | 59722,023 | 157,860 | 0,931 |
| 3 | 100 | 244,278 | 59671,553 | 157,722 | 0,931 |
| 4 | 20 | 271,445 | 73682,304 | 175,726 | 0,917 |
| 4 | 50 | 259,698 | 67443,124 | 164,422 | 0,924 |
| 4 | 60 | 257,858 | 66490,822 | 163,241 | 0,925 |
| 4 | 70 | 256,772 | 65932,103 | 162,504 | 0,925 |
| 4 | 80 | 256,246 | 65661,952 | 162,110 | 0,926 |
| 4 | 90 | 255,843 | 65455,648 | 161,877 | 0,926 |
| 4 | 100 | 255,523 | 65292,243 | 161,642 | 0,926 |
| 5 | 20 | 257,386 | 66247,521 | 173,669 | 0,925 |
| 5 | 50 | 244,366 | 59714,789 | 160,184 | 0,932 |
| 5 | 60 | 243,757 | 59417,671 | 159,316 | 0,933 |
| 5 | 70 | 243,139 | 59116,530 | 158,579 | 0,933 |
| 5 | 80 | 242,817 | 58960,194 | 158,146 | 0,933 |
| 5 | 90 | 242,538 | 58824,861 | 157,941 | 0,933 |
| 5 | 100 | 242,423 | 58768,734 | 157,852 | 0,934 |
| 6 | 20 | 263,100 | 69221,561 | 176,089 | 0,922 |
| 6 | 50 | 249,051 | 62026,193 | 162,391 | 0,930 |
| 6 | 60 | 246,918 | 60968,630 | 160,587 | 0,931 |
| 6 | 70 | 246,256 | 60641,937 | 160,213 | 0,931 |
| 6 | 80 | 245,615 | 60326,875 | 159,798 | 0,932 |
| 6 | 90 | 245,288 | 60166,007 | 159,485 | 0,932 |
| 6 | 100 | 244,976 | 60012,997 | 159,284 | 0,932 |
| 7 | 20 | 267,320 | 71460,234 | 180,748 | 0,917 |

Sigue en la página siguiente.

| k-fold | NumTree | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|
| 7 | 50 | 244,109 | 59589,232 | 160,542 | 0,931 |
| 7 | 60 | 243,313 | 59201,348 | 159,635 | 0,932 |
| 7 | 70 | 242,970 | 59034,353 | 159,227 | 0,932 |
| 7 | 80 | 242,557 | 58834,008 | 158,909 | 0,932 |
| 7 | 90 | 242,374 | 58744,933 | 158,714 | 0,932 |
| 7 | 100 | 242,237 | 58678,575 | 158,481 | 0,932 |
| 8 | 20 | 271,446 | 73682,966 | 178,588 | 0,916 |
| 8 | 50 | 252,228 | 63619,130 | 163,023 | 0,928 |
| 8 | 60 | 251,799 | 63402,622 | 162,525 | 0,928 |
| 8 | 70 | 251,299 | 63151,429 | 161,863 | 0,928 |
| 8 | 80 | 251,007 | 63004,538 | 161,438 | 0,929 |
| 8 | 90 | 250,822 | 62911,704 | 161,240 | 0,929 |
| 8 | 100 | 250,715 | 62857,776 | 161,140 | 0,929 |
| 9 | 20 | 251,136 | 63069,272 | 168,171 | 0,929 |
| 9 | 50 | 239,883 | 57543,986 | 157,420 | 0,935 |
| 9 | 60 | 239,248 | 57239,525 | 156,832 | 0,935 |
| 9 | 70 | 238,893 | 57070,035 | 156,497 | 0,935 |
| 9 | 80 | 238,695 | 56975,339 | 156,230 | 0,935 |
| 9 | 90 | 238,437 | 56852,378 | 155,970 | 0,936 |
| 9 | 100 | 238,214 | 56745,937 | 155,654 | 0,936 |
| 10 | 20 | 256,875 | 65984,999 | 171,329 | 0,925 |
| 10 | 50 | 248,952 | 61977,126 | 162,109 | 0,929 |
| 10 | 60 | 248,053 | 61530,493 | 161,364 | 0,930 |
| 10 | 70 | 247,474 | 61243,431 | 160,898 | 0,930 |
| 10 | 80 | 246,951 | 60985,033 | 160,452 | 0,931 |
| 10 | 90 | 246,586 | 60804,418 | 160,113 | 0,931 |
| 10 | 100 | 246,478 | 60751,564 | 159,987 | 0,931 |

La Figura 5.8 muestra la evolución $R^2$ de acuerdo al número de árboles definidos para el entrenamiento del método GBT, para los 10 subconjuntos de datos. Donde se aprecia la diferencia de los resultado a medida que aumentamos el número de árboles. Como ya mencionamos, a partir de los 50 árboles no se aprecia una mejora significativa en los resultados de la predicción. Para este método el mejor promedio se obtiene con 100 árboles con un $R^2 = 0,9302$.

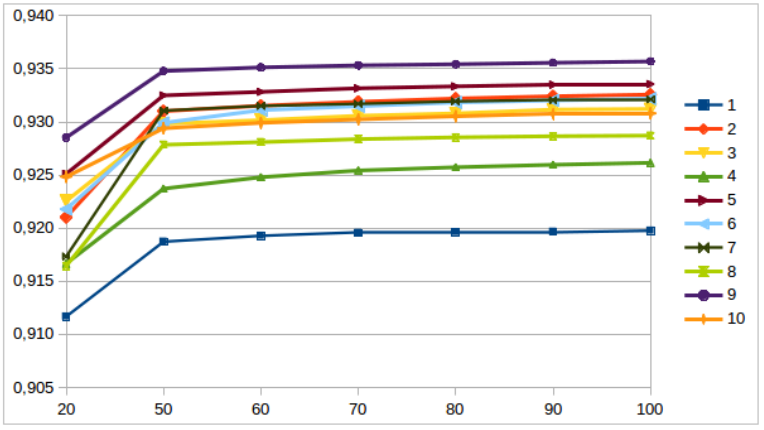


**Figura 5.8:** Gráfico del coeficiente de determinación para GBT

Para resumir los resultados de los parámetros óptimos obtenidos, a continuación se encuentra la Tabla 5.5.

| Método | maxDepth | NumTree | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|---|
| **Decision Tree** | 5 | - | 360,001 | 129629,067 | 240,375 | 0,8525 |
| **Random Forest** | 5 | 60 | 338,490 | 114607,637 | 253,044 | 0,8719 |
| **GBT** | - | 100 | 248,710 | 61921,351 | 161,836 | 0,9302 |

**Tabla 5.5:** Resultados de parámetros óptimos obtenidos con validación cruzada.

Para evaluar el rendimiento de los métodos, tanto en resultado como en eficiencia, analizamos el tiempo de ejecución del entrenamiento de los métodos, para identificar el más adecuado. La Tabla 5.6 resume los tiempos de generación del modelo de predicción, es decir los tiempos de entrenamiento (en segundos), para cada uno de los métodos, usando los parámetros óptimos obtenidos anteriormente.

| | R2 | Tiempo (seg) |
|---|---|---|
| **Decision Tree** | 0,8525 | 30 |
| **RandomForest** | 0,8719 | 55 |
| **GBT** | 0,9302 | 315 |

**Tabla 5.6:** Resultados de parámetros óptimos vs. tiempo de entrenamiento

A pesar de que la técnica de GBT ha obtenido el mejor resultado, no se puede asegurar que sea el método más adecuado, especialmente si se tiene en cuenta el tiempo necesario para generar el modelo con 100 árboles. Por ello se propone entrenar el modelo con 50 árboles, con el que se obtiene un $R^2 = 0,928$ y el tiempo necesario se reduce a 114 segundos. Esto nos permite asegurar que el rendimiento puede comenzar a disminuir para los GBT si la cantidad de árboles crece demasiado.

Finalmente podemos concluir que tanto Random Forests como GBT son algoritmos para entrenar conjuntos de árboles, lo que significa que se crea un clasificador para un gran número de árboles más pequeños. Ahora la diferencia fundamental radica en que los GBT entrenan un árbol a la vez, por lo que pueden llevar más tiempo entrenar que los Random Forests. Random Forests puede entrenar múltiples árboles en paralelo.

## 5.4. Evaluación del modelo

Después ajustar los modelos con el conjunto de entrenamiento, se hizo la evaluación con el conjunto de pruebas que inicialmente reservamos. Los resultados que se generaron se muestran en la tabla 5.7:

| | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|
| **Decision Tree** | 356,22 | 126896,61 | 240,08 | 0,858 |
| **Random Forest** | 265,61 | 70551,52 | 181,88 | 0,921 |
| **GBT** | 253,3 | 64165,41 | 167,42 | 0,928 |

**Tabla 5.7:** Resultado de evaluación de modelos

Como vemos, los resultado obtenidos han sido muy similares a los obtenidos en la selección del modelo con validación cruzada, por lo que podemos concluir que dicho modelo puede ser extrapolable, siempre que los conjuntos de datos sean similares a los utilizados para la elaboración del modelo.

A continuación vamos estudiar los errores máximos, mínimos y promedio que cometen los métodos analizados. Para ello agrupamos las predicciones por fechas para obtener el error diario.

En la Tabla 5.8 muestra los días donde el valor del error cuadrático medio ha sido máximo, mínimo y promedio por cada uno de los métodos. Para Decision Tree su mejor predicción la hace el 11 de noviembre para Random Forests también es el 11 de noviembre y para el método GBT su mejor predicción es para el 25 de noviembre.

| | Decision Tree | | Random Forests | | Gradient Boosted | |
|---|---|---|---|---|---|---|
| | **MSE** | **Fecha** | **MSE** | **Fecha** | **MSE** | **Fecha** |
| **error mínimo** | 14132,802 | 11/11/2015 | 7078,622 | 11/11/2015 | 8043,762 | 25/11/2015 |
| **error máximo** | 898280,44 | 16/07/2015 | 423684,952 | 07/08/2015 | 401121,941 | 07/08/2015 |
| **error medio** | 125683,203 | 11/04/2015 | 69788,112 | 09/09/2015 | 63385,34 | 20/06/2015 |

**Tabla 5.8:** Error mínimo, máximo y medio de los métodos en el conjunto test

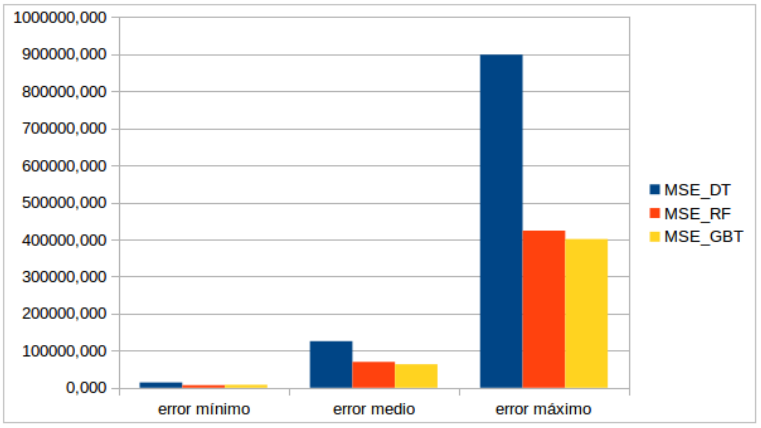


**Figura 5.9:** MSE diario del conjunto de datos test por método

La Figura 5.9 muestra la comparación los valores de MSE diarios de cada uno de los métodos que se relacionaron en la Tabla 5.8. A continuación se muestran gráficamente las mejores y peores predicciones diarias para cada uno de los métodos.

En la Figura 5.10 se puede ver el día con mejor predicción obtenida usando Decision Tree y MSE de 14132,802 que corresponde al 11 de noviembre de 2015, es un miércoles día laboral.

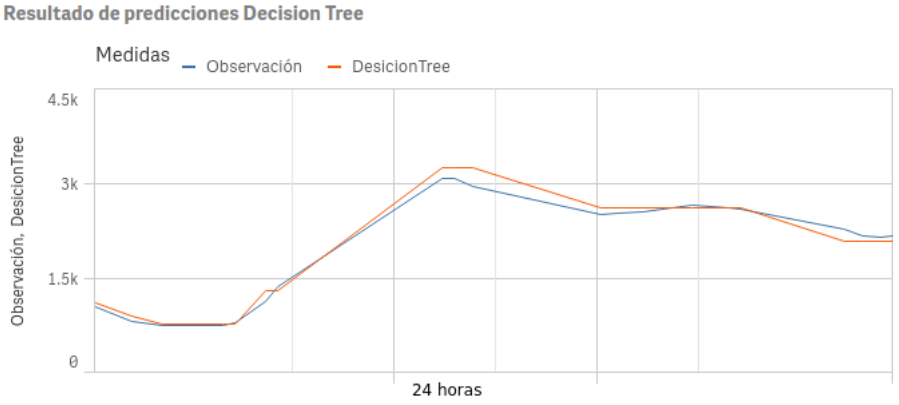


**Figura 5.10:** Predicción para el día 11/11/2015 por el método DT

En la Figura 5.11 se puede ver el día con peor predicción obtenida usando Decision Tree y MSE de 898280,440 que corresponde al 16 de julio de 2015, es un jueves día laboral.

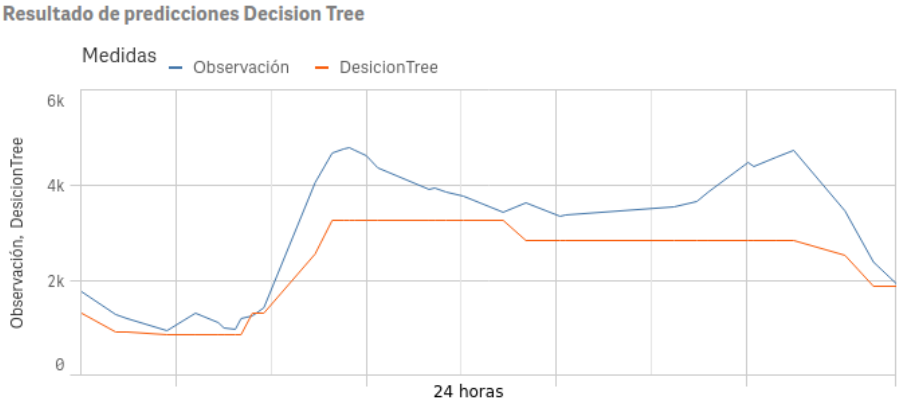


**Figura 5.11:** Predicción para el día 16/07/2015 por el método DT

En la Figura 5.12 se puede ver el día con mejor predicción obtenida usando Random Forest y MSE de 7078,622 que corresponde al 11 de noviembre de 2015, es un miércoles día laboral.

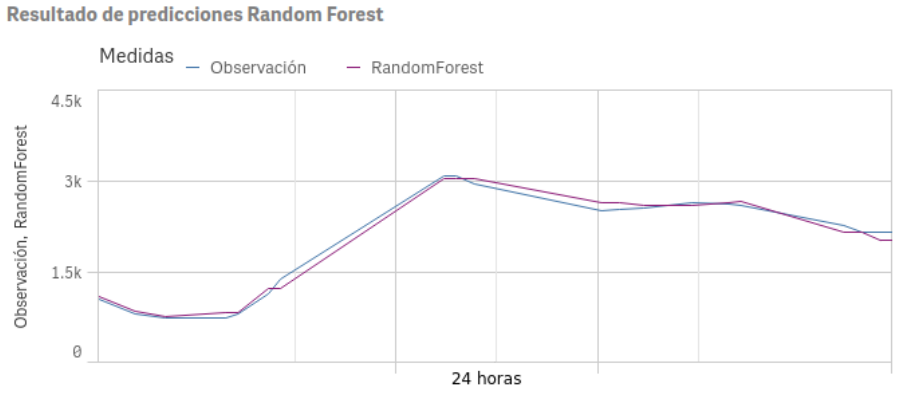


**Figura 5.12:** Predicción para el día 11/11/2015 por el método RF

En la Figura 5.13 se puede ver el día con peor predicción obtenida usando Random Forest y MSE de 423684,952 que corresponde al 7 de agosto de 2015, es un viernes día laboral, pero por ser verano y por estar cerca del fin de semana es posible que los hábitos de consumo de cambiaran en este día.

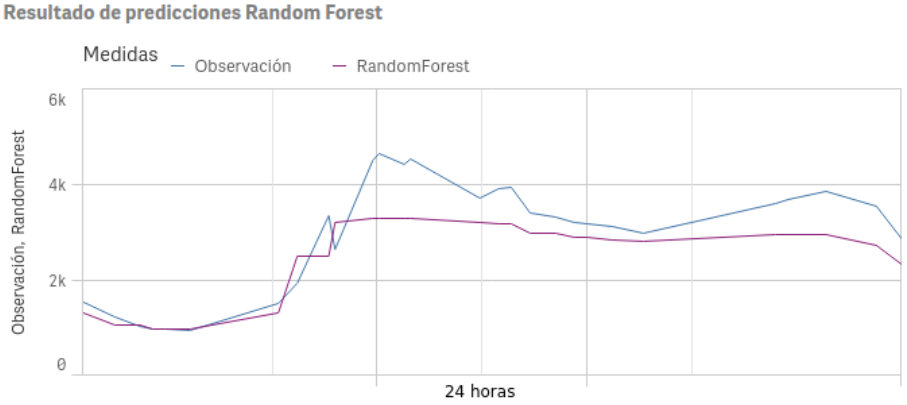


**Figura 5.13:** Predicción para el día 07/08/2015 por el método RF

En la Figura 5.14 se puede ver el día con mejor predicción obtenida usando Random Forest y MSE de 8043,762 que corresponde al 25 de noviembre de 2015, es un miércoles día laboral.

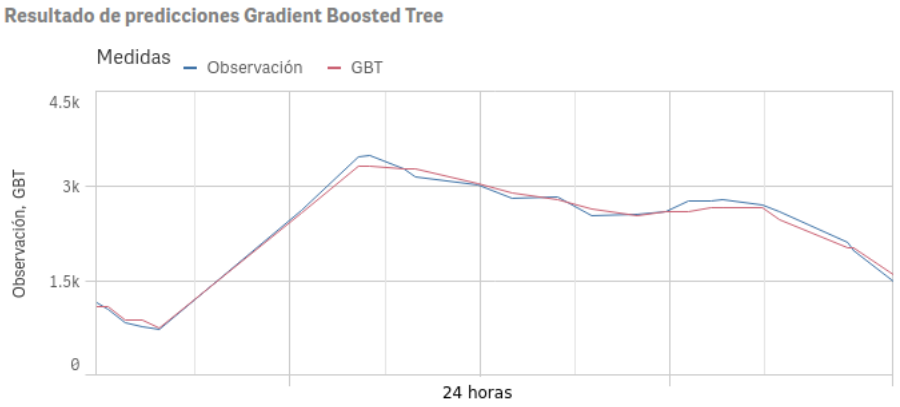


**Figura 5.14:** Predicción para el día 25/11/2015 por el método GBT

En la Figura 5.15 se puede ver el día con peor predicción obtenida usando Random Forest y MSE de 401121,941 que corresponde al 7 de agosto de 2015, es un viernes día laboral, pero por ser verano y por estar cerca del fin de semana es posible que los hábitos de consumo de cambiaran en este día.

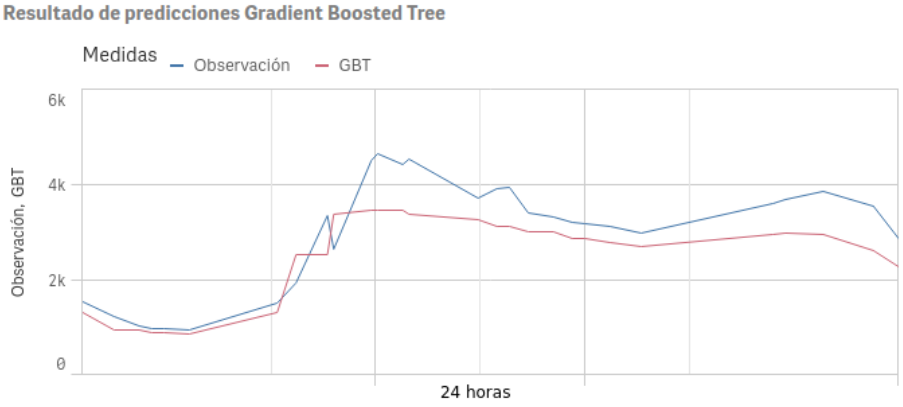


**Figura 5.15:** Predicción para el día 07/08/2015 por el método GBT

Podemos suponer con estos datos que noviembre es un mes muy estable en cuanto al consumo, ya que las 3 mejores predicciones ocurren en los días miércoles de noviembre. Las peores predicciones ocurren en verano, debido al desplazamiento de los consumidores por vacaciones y al cambio de comportamiento habitual por ser días de descanso y ocio.

Para complementar el estudio de las mejores y peores predicciones, a continuación generó la Tabla 5.9, que relaciona las predicciones de las 3 técnicas por meses ordenadas por error MSE y clasificadas en mejores y peores predicciones.

En la Tabla 5.9 se confirma lo mencionado anteriormente sobre las peores predicciones en días de verano. Aquí se observa que junio, julio y agosto tienen el error más alto en todos los métodos que el resto de meses. A diferencia de febrero, octubre y septiembre que cuentan con el mínimo error de predicción. Podemos suponer que la precisión de predicción para los meses de septiembre y octubre se debe a la temporada de otoño. También podemos

suponer que febrero y septiembre son meses de estabilización de comportamientos de los consumidores, por ser los meses que le siguen a los descansos de enero y agosto.

| | Decision Tree | | | Random Forests | | | Gradient Boosted | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Mes** | **MSE** | $\sigma$ | **Mes** | **MSE** | $\sigma$ | **Mes** | **MSE** | $\sigma$ |
| **Mejor** | **oct** | 69754,82 | 856,52 | feb | 32211,74 | 837,75 | feb | 27601,32 | 861,32 |
| | **feb** | 71407,57 | 836,74 | oct | 36472,27 | 829,21 | oct | 32858,21 | 851,55 |
| | **nov** | 74115,66 | 877,53 | sep | 41370,60 | 800,06 | sep | 35305,01 | 851,93 |
| | sep | 95383,16 | 882,21 | nov | 48937,16 | 847,09 | ene | 40599,05 | 848,66 |
| | may | 100604,98 | 907,36 | ene | 54805,65 | 785,37 | nov | 46642,67 | 875,72 |
| | mar | 102528,69 | 850,05 | may | 58754,92 | 879,19 | dic | 51439,38 | 840,28 |
| | ene | 102983,42 | 833,96 | dic | 59988,72 | 775,86 | may | 55600,01 | 928,89 |
| | abr | 104534,16 | 893,91 | mar | 62179,67 | 840,90 | mar | 57712,68 | 878,45 |
| | ago | 125369,84 | 879,15 | abr | 74382,814 | 872,57 | abr | 67478,52 | 910,92 |
| **Peor** | **jun** | 129051,55 | 885,95 | jun | 86965,39 | 881,61 | jun | 85671,98 | 915,95 |
| | **dic** | 134097,05 | 866,45 | ago | 100330,88 | 815,57 | ago | 91077,06 | 880,72 |
| | **jul** | 389598,98 | 877,45 | jul | 173088,45 | 992,60 | jul | 161341,77 | 1035,13 |

**Tabla 5.9:** Meses con mejor y peor predicción

En el capítulo de visualización (6), que está a continuación, se mostrará de manera visual algunos ejemplos de los resultados obtenidos.

# Capítulo 6

# Visualización

*"Ésta es mi parte favorita sobre análisis: tomar datos planos aburridos y darles vida a través de la visualización".*

John Tukey (matemático)

**Resumen:** En este capítulo se visualizan los resultados de las predicciones de los modelos entrenados en la etapa de aprendizaje, desarrollada en el Capítulo 5, usando la herramienta *Qlik*. En cada figura se muestra la comparación de los 3 modelos entrenados y la contrastación empírica de las hipótesis propuestas en el Apartado 1.3.

## 6.1. Introducción

La visualización de datos juega un papel muy importante durante todo el proceso de análisis. Desde la exploración de datos inicial, donde ayuda a exponer los patrones a través del tiempo y los patrones entre diferentes variables, hasta el desarrollo de modelos de predicción para informar sobre los resultados analíticos producidos por dichos modelos.

Las técnicas de visualización de datos y el software son herramientas clave para todo científico de datos. Para este trabajo se usó *Qlik*, que es una plataforma de Business Intelligence muy flexible, fácil de usar y con una versión gratuita muy completa.

## 6.2. Predicciones

Como se evidencia en la tabla 5.7, de los tres métodos de regresión evaluados, el mejor resultado se obtuvo con el Gradient-boosted tree entrenado con 50 árboles, con un coeficiente de determinación ($R^2$) de 0.928.

Para identificar mejor la diferencia entre los resultados obtenidos y mantener la tendencia de la serie, se importó a *Qlik*, tanto los resultados de las predicciones, como el dataset de prueba con fechas, timestamp y demás datos, con el objetivo de ordenar y visualizar dichos resultados.

La Figura 6.1 muestra la tendencia del consumo anual por hora. De forma general se podría decir que los 3 métodos de aprendizaje usados tienen un buen resultado de predicción y en gran medida simulan la tendencia real de consumo.

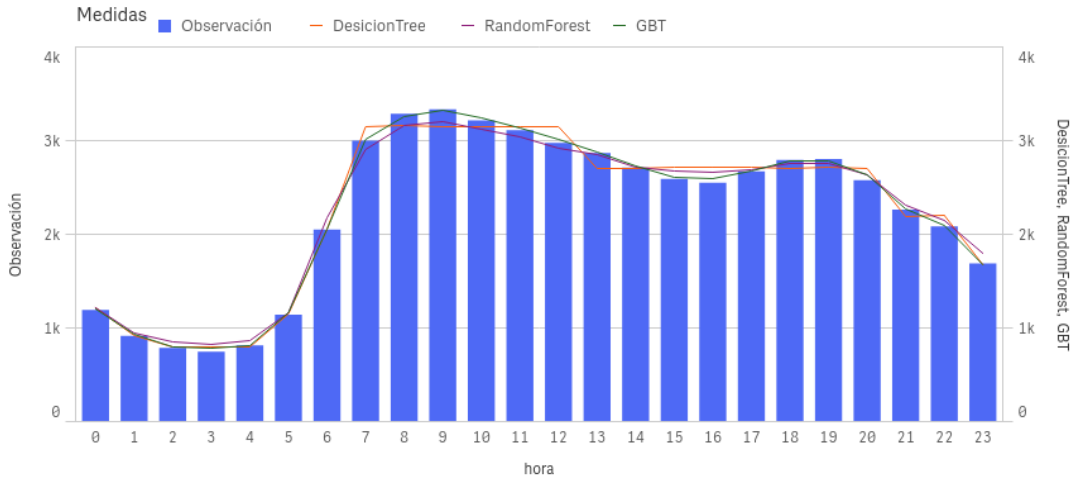


**Figura 6.1:** Comparación de resultados de los 3 métodos de regresión- Promedio anual

No obstante, se quiere ver el comportamiento de cada método en diferentes situaciones o eventos estacionales. A continuación se muestra las gráficas de los resultados de las predicciones, donde, ademas de comparar los 3 métodos de regresión entrenados, también se quiso comparar el comportamiento del modelo en días laborales, fines de semana, vacaciones y días especiales.

### 6.2.1. Predicción días laborales

Para ver el comportamiento de la tendencia de las predicciones en días laborales, se tomó como ejemplo inicialmente los días 14 y 15 de abril de 2015, que corresponden a martes y miércoles respectivamente. Consideramos el martes y miércoles como los días de la semana laborales, porque no se encuentran cerca de del fin de semana anterior ni posterior. Se escogió el mes de abril porque, como se muestra en la Tabla 5.9, forma parte de los meses cuya predicción no es la mejor ni la peor.

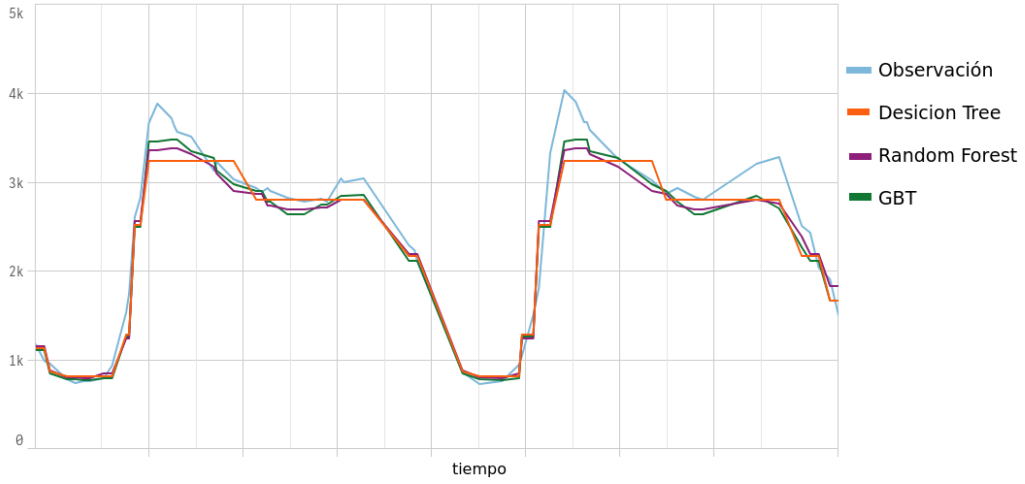


**Figura 6.2:** Comparación de resultados para días laborales de abril

Como se observa en la Figura 6.2, la muestra observada presenta un alto consumo a primera hora de la mañana, donde ninguno de los 3 métodos es capaz de predecir acertadamente el consumo real observado a esa hora de la mañana. Esto se debe a que en el mes

de abril el promedio máximo de consumo gira alrededor de los 3300 $m^3/h$y estos 2 días superan los 3500.

Según la Tabla 5.9 la predicción para el mes de octubre es una de las mejores. Esto queda demostrado en la Figura 6.3 donde se muestra la predicción para los días 21 y 22 de octubre, miércoles y jueves respectivamente, que son laborales. Es evidente que la técnica Decision Tree, no tiene una alta precisión, esto se debe a que, a diferencia de las otras 2 técnicas, ésta solo entrena un árbol y por supuesto, en estas condiciones, su rendimiento no es el mejor.

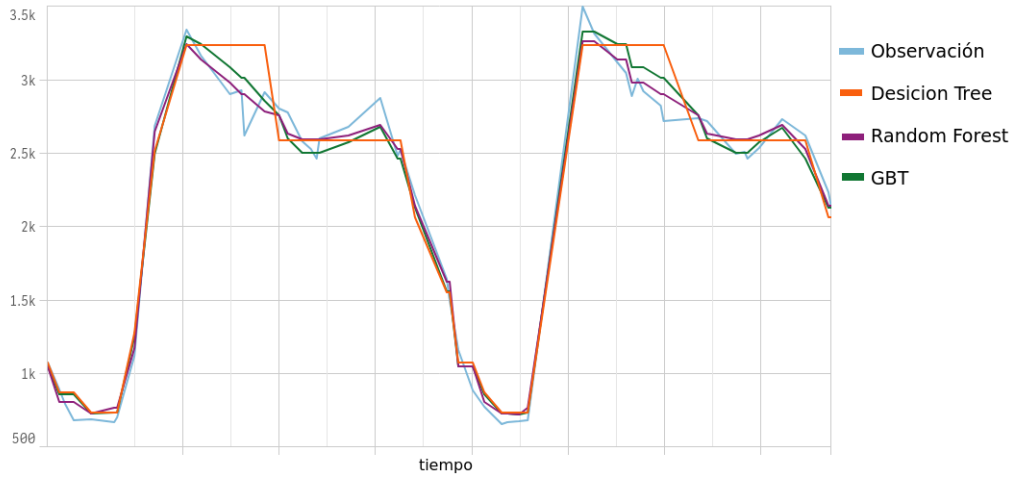


**Figura 6.3:** Comparación de resultados para días laborales de octubre

La predicciones para el mes de junio se encuentran entre las peores (Tabla 5.9). No obstante en la Figura 6.4 parecen ajustarse muy bien al consumo real, dentro de lo aceptable. Esta gráfica corresponde a los días 9 y 10 de junio, martes y miércoles respectivamente. Para el martes vemos un periodo de tiempo cuyas predicciones son superiores al consumo real, mientras que para el miércoles el consumo a primera hora de la mañana supera las predicciones.

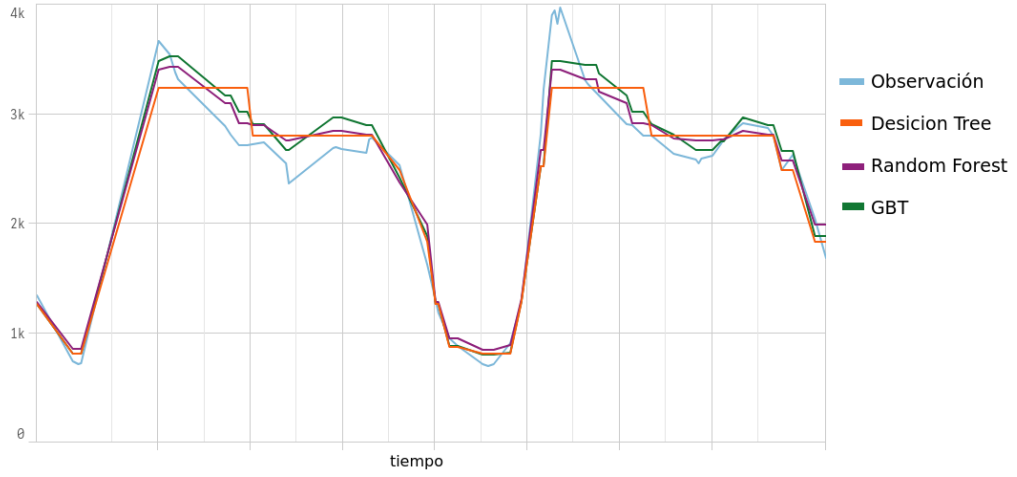


**Figura 6.4:** Comparación de resultados para días laborales de junio

### 6.2.2. Predicción fin de semana

Para visualizar el comportamiento de los resultados en los fines de semana, se tomaron varias observaciones, donde se incluyen meses de varias estaciones para comparar la precisión de la predicción por temporada. Iniciamos con mayo, mes primaveral, que tiene una predicción media, con los días 23 y 24, sábado y domingo respectivamente.

En la Figura 6.5, al igual que en la Figura 6.3 la predicción con el algoritmo Decision Tree es el que peor resultados ofrece, sobre todo en los periodos de máximo consumo. El sábado 23 ninguno de los 3 métodos alcanza a predecir el máximo consumo a primera hora de la mañana. Para el domingo, Gradient Boosted Tree se ajusta de una manera mucho más fiel a los datos reales que los otros dos métodos.

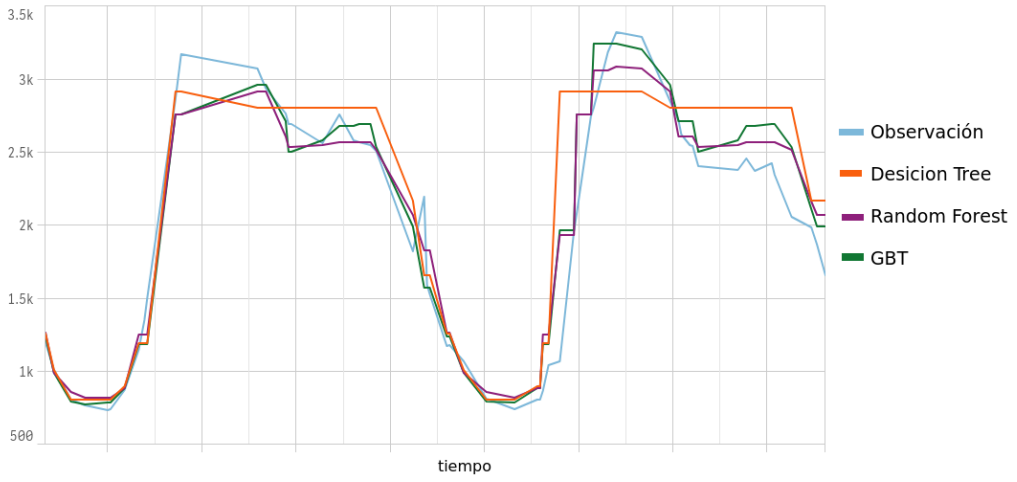


**Figura 6.5:** Comparación de resultados en fin de semana de mayo.

Al observar fines de semana de diferentes meses, el resultado varía, ya que son días de ocio donde las personas varían su comportamiento habitual. En algunos, la predicción es muy buena para sábado y domingo; en otros ejemplos la predicción es buena sólo para el sábado y en otros sólo para domingo. Ello nos da a entender que existen otras variables, diferentes a horario y fecha, que afectan el consumo en fines de semana.

Para la temporada de invierno tomamos datos del mes de noviembre (ver Figura 6.6), específicamente los días 7 y 8 que corresponden a fin de semana en medio de 2 festivos el 1 y 18 de noviembre. Este mes puede considerarse con buena precisión de predicción, como se observa en la gráfica. Los algoritmos Random Forests y Gradient Boosted Tree simulan muy bien la tendencia de consumo, mientras que Decision Tree, como hemos mencionado anteriormente, no llega a predecir los máximos consumos de agua. En este caso tanto sábado y domingo tienen un alto grado de precisión.

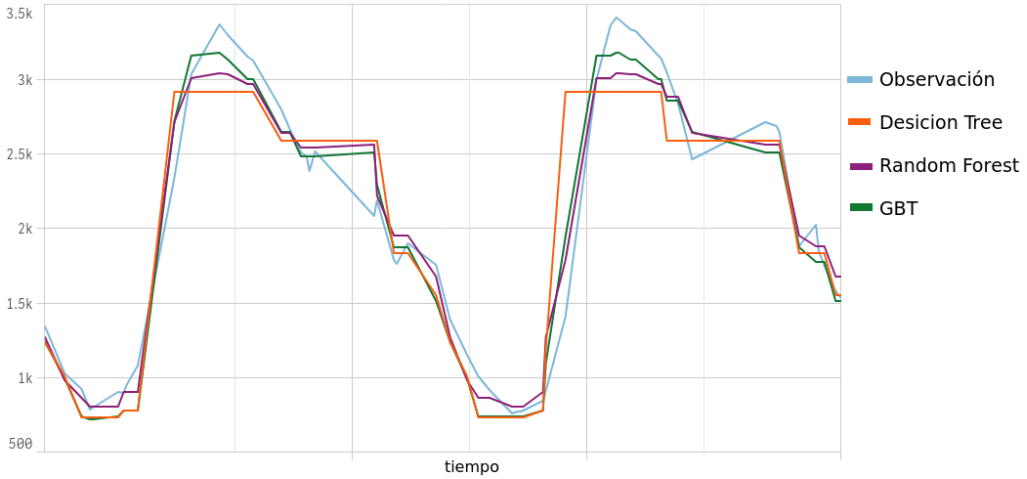


**Figura 6.6:** Comparación de resultados en fin de semana de invierno.

Finalmente para temporada de verano tenemos los días 18 y 19 de julio que son fin de semana. Según muestra la Tabla 5.9 se considera uno de los meses con peor predicción. No obstante, como se observa en la Figura 6.7, la predicción de los algoritmos Random Forests y Gradient Boosted se ajusta muy bien al consumo real, a diferencia del Decision Tree que esta muy por debajo del rendimiento de las otras técnicas.

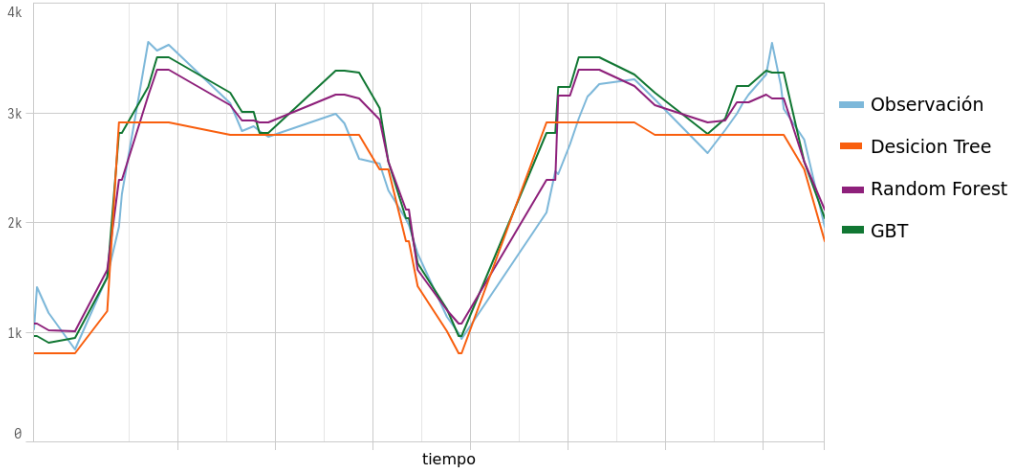


**Figura 6.7:** Comparación de resultados en fin de semana de verano.

### 6.2.3. Predicción fechas especiales

Como fechas especiales se tomaron 3 pares de días, que se consideran de alguna manera diferentes al resto de año.

La Figura 6.8 corresponde a los días 26 y 27 de diciembre 2015, que son sábado y domingo después de Navidad, estos 2 días se pueden considerar fin de semana o vacaciones. Las predicciones para éstas fechas, mostradas en la gráfica, son muy buenas, se percibe la similitud del valor observado con el resultado de la predicción Gradient Boosted.

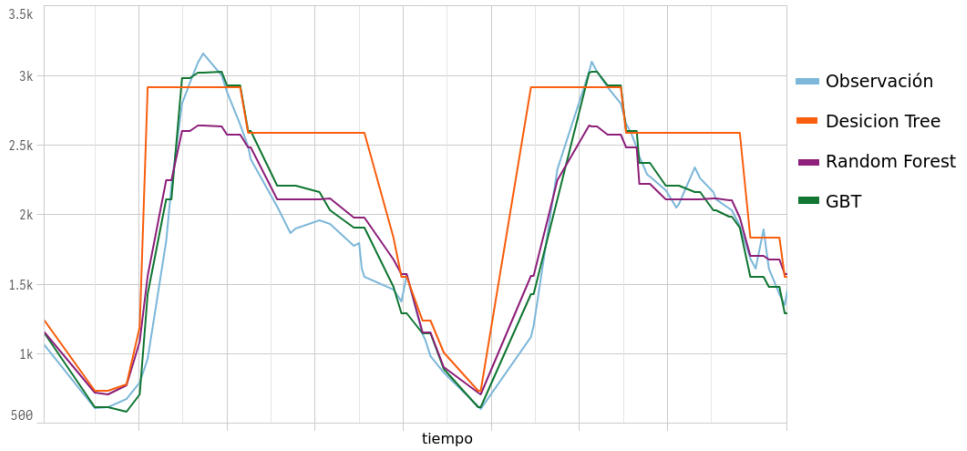


**Figura 6.8:** Comparación de resultados en navidad

Hasta el momento los algoritmos Random Forests y Gradient Boosted Tree tenían unos resultados de predicción muy parecidos y se acercaban mucho al consumo real, sin embargo en este caso, el método RF no lográ predecir el pico máximo de demanda. Este ejemplo nos confirma que para días diferentes o con eventos especiales, el método Gradient Boosted que realiza iteraciones secuenciales, logra mejorar la predicción debido a que en cada iteración, el árbol de decisiones ayudará a corregir errores previos. Como es de esperar, Decision Tree, en días especiales, tiene la peor predicción que hasta el momento hemos observado, el aprendizaje con un sólo árbol no es suficiente para estos casos.

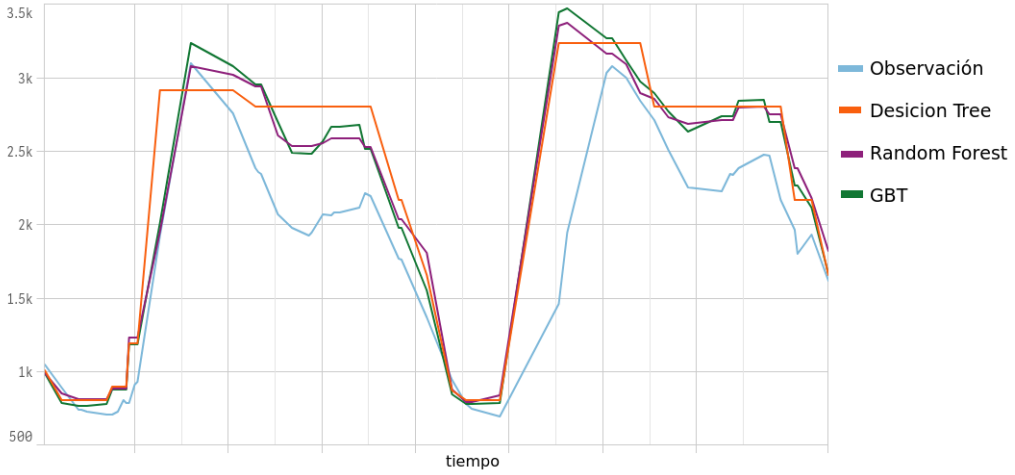


**Figura 6.9:** Comparación de resultados en Pascua

La Figura 6.9 corresponde a los días 5 y 6 de abril 2015, son domingo y lunes de Pascua, que para el país donde está ubicada la WDN, el lunes es festivo después de la Semana Santa. Como se observa en la gráfica, el consumo predicho, es superior al valor observado, lo que hace pensar, que este menor consumo se debe a una posible disminución en el número de habitantes de la zona, por tratarse de un periodo largo de descanso. Además de ello, la predicción de las 3 técnicas para el lunes 6 se ven desplazadas en el tiempo, es decir, se comportan como un día laboral donde el consumo de agua se inicia más temprano que en día de descanso. Este es un claro ejemplo donde la regresión no es capaz de identificar los

comportamientos en el consumo de agua de un día de fiesta, debido a que en el año tenemos pocos lunes festivos en comparación con los lunes normales y al entrenar el modelo no tiene suficiente información de este evento.

Finalmente en la Figura 6.10 muestran los días 2 y 3 de enero de 2015. Corresponden a viernes y sábado, el viernes es día laboral, no obstante, al encontrarse entre el festivo de fin de año y un fin de semana, la tendencia de consumo puede variar. El consumo real para el día 2 refleja una tendencia similar a un festivo, sin embargo la predicción de las 3 técnicas se ven desplazadas en el tiempo, es decir, se comportan como un día laboral donde el consumo de agua inicia más temprano que en día de descanso. Para el día 3, la predicción del modelo Gradient Boosted tiene un buen ajuste respecto a la observación. Las otras técnicas tienen una menor precisión en la predicción.

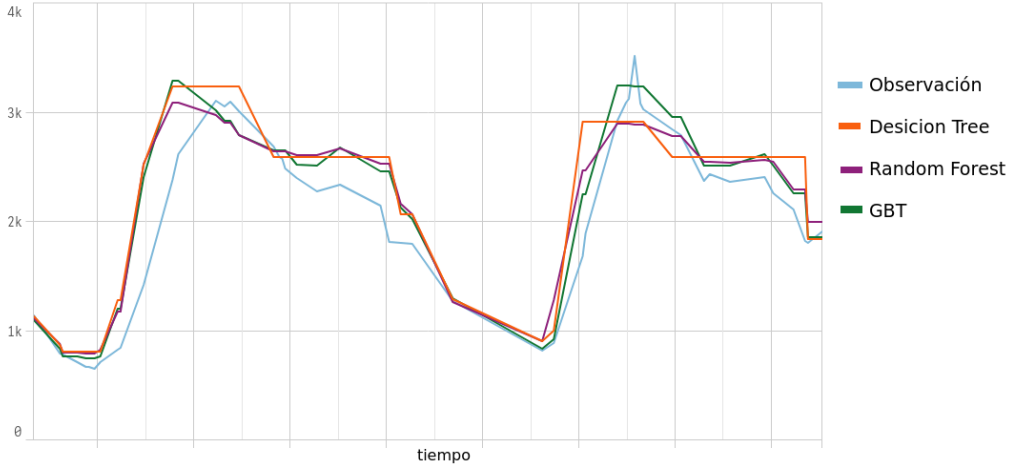


**Figura 6.10:** Comparación de resultados en año nuevo

# Capítulo 7

# Conclusiones y trabajo futuro

*Si el pasado no puede ser una regla para el futuro, toda la experiencia se vuelve inútil y no puede dar lugar a inferencias ni conclusiones.*

David Hume
An Enquiry Concerning Human Understanding

Disponer de herramientas que permitan el seguimiento del consumo de agua y que faciliten un análisis predictivo, no sólo ayuda a garantizar la distribución adecuada al usuario final, sino que también fomenta la sostenibilidad al reducir las pérdidas, optimizar los recursos para disminuir el gasto energético y económico en las empresas abastecedoras y ocasionar el menor impacto ambiental posible.

En este trabajo se ha desarrollado el procesamiento, análisis y la creación de modelos de aprendizaje automático para monitorizar la demanda en redes de agua utilizando datos recopilados de los sensores de caudal y presión. Se siguió la metodología KDD, junto con el proceso ELT para la construcción del Data Lake, en un entorno Big Data. Se generó la visualización de los resultados de la predicción producidos por los distintos modelos de aprendizaje, lo que permite hacer un análisis visual y comparación de dichos resultados.

El proceso ELT es clave para lograr que los datos extraídos asíncronamente de orígenes heterogéneos se integren finalmente en un entorno homogéneo, obteniendo de este proceso nuestro Data Lake. El Data Lake es el repositorio que contiene una gran cantidad de datos en bruto en su formato original (estructurado, no estructurado o semiestructurado), con una arquitectura plana para almacenar los datos y un esquema dinámico creado al momento de la ejecución de la consulta. El Data Lake se asocia a menudo con el almacenamiento de objetos orientado a Hadoop.

Es muy importante modelar y describir en forma detallada las fuentes de datos, para hacerse una idea de la problemática y comprender las propiedades de los datos que llegan al Data Lake y así lograr satisfacer los requisitos de información.

El uso del ecosistema Hadoop en este trabajo, garantiza que todo el análisis, procesamiento y métodos de aprendizaje aplicados para predecir el consumo de agua, sea escalable, de procesamiento rápido y con un almacenamiento de bajo costo.

Para el aprendizaje automático se seleccionaron los modelos de regresión Decision Tree, Random Forest y Gradient Boosted Tree porque los árboles de decisión manejan características categóricas, no requieren escalado de características y son capaces de capturar

interacciones no lineales. Se mostraron los resultados de la predicción, obteniendo mejor resultado con el algoritmo GBT usando 100 árboles. El segundo mejor resultado lo obtuvimos con Random Forests para 60 árboles. Como era de esperar, Decision Tree tiene la peor predicción por ser un método que entrena un sólo árbol.

Aunque GBT es el que mejor resultados ha ofrecido, los tiempos de entrenamiento tan han sido superiores a las otras técnicas. Por ello se propuso disminuir el número de árboles a la mitad, con lo que logramos disminuir el tiempo, en más de la mitad, con una precisión de predicción casi igual a la obtenida inicialmente con este algoritmo. A nivel general los resultados en las predicciones son aceptables para días "promedio", se ha observado que para festivos y para determinadas fechas, los algoritmos de predicción no funcionan muy bien. Razón por la cuál, se debe extender este estudio o considerar otras técnicas que funcionen bien en estas circunstancias, de ello se discutirá en el trabajo futuro.

Finalmente, basándonos en la revisión bibliográfica, se puede afirmar que el desarrollo aplicado a casos reales en sistemas de abastecimientos de agua de las técnicas expuestas, no ha sido ampliamente desarrollado; debido, en mi opinión, a 2 factores. Primero, a la dificultad para la consecución de la información y la confidencialidad de la misma por parte de los entes encargados del manejo y gestión de estos sistemas; y segundo por que aún existe un poco de recelo sobre la aplicación de nuevos paradigmas para el análisis y estudio de los datos en la gestión o toma de decisiones en sus redes y se prefiere el diseño y modelado clásico.

## 7.1. Trabajo futuro

A pesar de que los resultados obtenidos con los algoritmos de regresión de la biblioteca ML y MLlib de *Apache Spark* se consideran aceptables, se debe tener en cuenta que los modelos pueden ser extrapolados siempre y cuando los conjuntos de datos sean similares a los utilizados para la elaboración del modelo.

Se sabe que en la realidad, esta condición no se cumple, debido a la variedad de datos y series temporales que manejan las diferentes empresas de abastecimiento de agua y de las tendencias de consumo de la población, que en gran medida depende del tipo y tamaño de la ciudad. Para el caso de estudio de este trabajo, se trataba de una población pequeña, de comportamientos de consumo definidos y poco cambiantes. Como se menciono anteriormente, se debe tener en cuenta que los modelos de previsión de la demanda de agua que utilizan unicamente el valor de la demanda, son capaces de generar un pronóstico bastante preciso, pero si se requiere incluir variables climatologías, demográficas, etc, los resultados de la predicción empeoran.

Es aquí donde se ve la necesidad de implementar un algoritmo que cubra estas debilidades y que pueda extrapolarse con buenos resultados de predicción, a pesar de que la variación de la tendencia de consumo tenga mayor fluctuación y que la calidad de los datos de los sensores sea menor a la usada para este trabajo, para ello se propone la implementación de algoritmos de regresión usando redes neuronales recurrentes (RNN). Las RNN son un poderoso conjunto de algoritmos de redes neuronales artificiales, especialmente útiles para procesar datos secuenciales, como datos de sonido, series de tiempo (sensores) o lenguaje natural escrito. Este tipo de red incluye un bucle de retroalimentación, por lo que la salida del paso n-1 se retroalimenta a la red para afectar el resultado de la etapa n, y así sucesivamente para cada paso posterior [27].

Dentro del contexto de este trabajo fue probado el concepto de una red neuronal recurrente, específicamente una red tipo Long Short-Term Memory (LSTMs), diseñada para manejar dependencia de la secuencia y problemas de predicción de series de tiempo. La red LSTM fue configurada pero por falta de tiempo solo fue posible probar una implementación en 2 dimensiones que no da unos buenos resultados porque no es correcta y faltaría corregir el problema de carga del dataset en 3 dimensiones y rehacer los datos de entrada. En el futuro se implementará y configurará adecuadamente dicha red neuronal y se contrastará su resultado con los obtenidos en este trabajo.

# Bibliografía


[1] Scolnicov, H.; Horowitz, G.(2010) Water network monitoring: a new approach to managing and sustaining water distribution infrastructure.

[2] García, S.; Ramírez, S.; Luengo J. (2016). Big Data: Preprocesamiento y calidad de datos. Novática. 237, 17-23. http://sci2s.ugr.es/sites/default/files/ficherosPublicaciones/2133_Nv237-Digital-sramirez.pdf

[3] Berkeley NLP Group. ScalaNLP Scientific Computing, Machine Learning y Natural Language Processing. [Última consulta: 26-junio-2018]. Disponible en: http://www.scalanlp.org/

[4] Berkeley NLP Group. Interpolation. [Última consulta: 26-junio-2018]. Disponible en: https://github.com/scalanlp/breeze/wiki/Interpolation

[5] Stein, B.; Morrison, A. (2014). The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast. [Última consulta: 9-julio-2018]. Disponible en: https://www.pwc.com/us/en/technology-forecast/2014/cloud-computing/assets/pdf/pwc-technology-forecast-data-lakes.pdf

[6] Miloslavskaya, N.; Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. Procedia Computer Science. 88. 300-305. https://doi.org/10.1016/j.procs.2016.07.439

[7] Campbell, C. (2015). Las cinco principales diferencias entre Data Lake y Data Warehouse. [Última consulta: 9-julio-2018]. Bluegranite. https://www.blue-granite.com/blog/bid/402596/top-five-differences-between-data-lakes-and-data-warehouses

[8] Donges, N. (2018). The Random Forest Algorithm. [Última consulta: 13-julio-2018]. Disponible en: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

[9] Zhang, C.; Yunqian M. (2012). Ensemble Machine Learning: Methods and Applications. Springer.

[10] Russell, S.; Norvig P. (2010). Artificial Intelligence: A Modern Approach. Prentice Hall.

[11] Xia, T.; Lai, T.; Saunders, M.; Shih, M.C. (2014). Gradient Boosting Machine for High-dimensional Additive Models. Stanford University. Institute for Computational and Mathematical Engineering.

[12] Friedman, J. (2001). Greedy Function Approximation: A Gradient Boosting Machine. [Última consulta: 13-julio-2018]. Disponible en: https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[13] Serra,J.(2015) What is a Data Lake. [Última consulta: 26-junio-2018]. Disponible en: http://www.jamesserra.com/archive/2015/04/what-is-a-data-lake

[14] Karim R. (2018). Sacala Machine Learning Project: Build real-world machine learning and deep learning project with Scala. Packt.

[15] Han, J.; Kamber M. (2001). Data Mining: Concepts ans Techniques. Morgan Kaufmann Publishers, USA.

[16] Zubcoff, J; Pardillo, J. (2009) A UML profile for the conceptual modelling of data-mining with time-series in data warehouses. Information and Software Technology, 51(6), 977-992. doi.org/10.1016/j.infsof.2008.09.006.

[17] Fernández, A.; del Río, S; López, V. (2014). Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 4(5): pp. 380409.

[18] Galicia, A.; Torres, J.; Martínez, F. (2017). Scalable Forecasting Techniques Applied to Big Electricity Time Series. International Work-Conference on Artificial Neural Networks. IWANN 2017. 165-175. DOI:10.1007/978-3-319-59147-6_15.

[19] Apache Software Foundation. Machine Learning Library (MLlib) for Spark. [Última consulta: 11-junio-2018]. Disponible en: http://spark.apache.org/docs/latest/mllib-guide.html.

[20] Martínez, M. A. (2017-2018). Aterrizando en la Landing Zone. Material para la asignatura Almacenamiento Escalable. Máster en Ingeniería Informática. Universidad de Valladolid.

[21] Díaz, F. (2017-2018). Apache Flume. Material para la asignatura Almacenamiento Escalable. Máster en Ingeniería Informática. Universidad de Valladolid.

[22] Alonso, C.; Pulido, B. (2016-2017). Metodologías análisis big data. Técnicas escalables de análisis de datos. Máster en Ingeniería Informática. Universidad de Valladolid.

[23] Vivaracho, C. (2017-2018) Principios básicos de visualización. Material para la asignatura Inteligencia de Negocios. Máster en Ingeniería Informática. Universidad de Valladolid.

[24] Sánchez, A. (2016). Urban Innovation Project Manager (SUEZ). Congreso Smart Urban

[25] Goodfellow, I; Bengio, Y. (2016). Neural Networks and Deep Learning. MIT Press, USA. http://neuralnetworksanddeeplearning.com

[26] Brownlee,J. (2017). Time Series Forecasting with the Long Short-Term Memory Network in Python. [Última consulta: 26-junio-2018]. Disponible en: https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/

[27] Britz, D. (2015). Recurrent Neural Networks Tutorial, Part 1 Introduction to RNNs. WILDML, Artificial Intelligence, Deep Learning, and NLP.[Última consulta: 27-junio-2018]. Disponible en: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[28] LeCun, Y; Bengio, Y. (2015). Deep learning. Nature. 521, 436-444. dx.doi.org/10.1038/nature14539

[29] Olah C. (2015). Understanding LSTM Networks. [Última consulta: 20-junio-2018]. Disponible en: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[30] Open-Source Community. Introduction to Core Deeplearning4j Concepts. [Última consulta: 24-junio-2018]. Disponible en: https://deeplearning4j.org/core-concepts

[31] Open-Source Community. Recurrent Networks and LSTMs. [Última consulta: 27-junio-2018]. Disponible en: https://deeplearning4j.org/recurrentnetwork

[32] Bartolín, HJ. (2013). Confección de modelos de redes de distribución de agua desde un Sig y desarrollo de herramientas de apoyo a la toma de decisiones. Tesis doctoral no publicada. Universidad Politécnica de Valéncia. doi:10.4995/Thesis/10251/33152.

[33] Galicia, A. (2017). Técnicas escalables para la predicción de series temporales de gran dimensión. Trabajo de fin de máster. Universidad Pablo de Olavide Sevilla. http://eps.upo.es/martinez/TFM/TFM_Galicia_2017.pdf

[34] Garcia, D.; Gonzalez, D.; Quevedo, J. (2015). Water demand estimation and outlier detection from smart meter data using classification and Big Data methods. Center of research in supervision, safety and automatic control ($CS^2AC$). Universidad Politécnica de Cataluña. https://upcommons.upc.edu/handle/2117/26473

[35] Balbontín, M. (2015). Big Data and Data Analytics applied to the Monitoring of Water Distribution Networks. Proyecto final de máster oficial. Universidad Politécnica de Cataluña. https://upcommons.upc.edu/handle/2117/80831

[36] Herrera M.; Torgo L.; Izquierdo J. (2010). Predictive models for forecasting hourly urban water demand. Journal of Hydrology. 387 (1-2), 141-150. doi.org/10.1016/j.jhydrol.2010.04.005

[37] Jach, T.; Magiera, E.; Froelich, W. (2015). Application of HADOOP to Store and Process Big Data Gathered from an Urban Water Distribution System. Procedia Engineering. 119, 1375-1380. doi.org/10.1016/j.proeng.2015.08.988

[38] Brentan, B.; Luvizotto, E.; Herrera, M. (2017). Hybrid regression model for near real-time urban water demand forecasting. Journal of Computational and Applied Mathematics. 309. 532-541. doi.org/10.1016/j.cam.2016.02.009

[39] Bakker, M.; Duist, H.; Schangen K. (2014). Improving the Performance of Water Demand Forecasting Models by Using Weather Inpu. Procedia Engineering.70, 93-102. doi.org/10.1016/j.proeng.2014.02.012

[40] Kumar, A.; Grosso, J.; Sopasakis P. (2014). Water demand forecasting for the optimal operation of large-scale drinking water networks: The Barcelona Case Study.IFAC Proceedings Volumes. 47(3). 10457-10462. doi.org/10.3182/20140824-6-ZA-1003.01343

[41] Romano, M.; Kapelan, Z. (2014). Adaptive water demand forecasting for near real-time management of smart water distribution systems. Environmental Modelling & Software. 60, 265-276. doi.org/10.1016/j.envsoft.2014.06.016

[42] Asociación Española de Abastecimiento de Agua y Saneamiento AEAS. (2014). Control del agua no registrada.

[43] Instituto IMDEA Agua. (2017). El sector del abastecimiento y saneamiento urbano en España. Fundación Canal.

[44] Kingdom B., Liemberger R., Marin P. (2006)The Challenge of Reducing Non-Revenue Water (NRW) in Developing Countries - How the Private Sector Can Help: A Look at Performance-Based Service Contracting, The World Bank, Washington DC, Paper 8.

[45] Mazzolani, G.; Berardi, L. ; Laucelli, D.; Martino, R. ; Simone, A.; Giustolisi O. (2016). A methodology to estimate leakages in water distribution networks based on inlet flow data analysis. Procedia Engineering. 162, 411-418.

[46] Gerald, G.; Navas, R. (2016).Leak detection in virtual DMA combining machine learning network monitoring and model based analysis. Amsterdam International water week. https://www.rbs-wave.de/wp-content/uploads/AIWW-Zusammen.pdf

# Apéndice A

# Flume

```
# lista de fuentes, sumideros y canales del agente (agentSeries)
agentSeries.sources = consum press
agentSeries.channels = channelSensor
agentSeries.sinks = HDFS_1 HDFS_2

# propiedades de la fuente consum
agentSeries.sources.consum.type = spooldir
agentSeries.sources.consum.spoolDir = /home/cloudera/data/consumption
agentSeries.sources.consum.basenameHeader = true
agentSeries.sources.consum.fileHeader = false
agentSeries.sources.consum.fileSuffix = .COMPLETED
agentSeries.sources.consum.channels = channelSensor

# propiedades de la fuente press
agentSeries.sources.press.type = spooldir
agentSeries.sources.press.spoolDir = /home/cloudera/data/pressure
agentSeries.sources.press.basenameHeader = true
agentSeries.sources.press.fileHeader = false
agentSeries.sources.press.fileSuffix = .COMPLETED
agentSeries.sources.press.channels = channelSensor

# propiedades del canal
agentSeries.channels.channelSensor.type = file
agentSeries.channels.channelSensor.checkpointDir = /var/lib/flume-ng/file-channel/
    checkpoint
agentSeries.channels.channelSensor.dataDirs = /var/lib/flume-ng/file-channel/data
agentSeries.channels.channelSensor.checkpointInterval = 30000
agentSeries.channels.channelSensor.capacity = 1000000
agentSeries.channels.channelSensor.transactionCapacity = 10000

# propiedades del sumidero 1
agentSeries.sinks.HDFS_1.type = hdfs
agentSeries.sinks.HDFS_1.channel = channelSensor
agentSeries.sinks.HDFS_1.hdfs.path = hdfs://quickstart.cloudera:8020/user/WDN/
    rawData/consumption/%{basename}
agentSeries.sinks.HDFS_1.hdfs.filePrefix = consum
agentSeries.sinks.HDFS_1.hdfs.fileSuffix = .csv
agentSeries.sinks.HDFS_1.hdfs.useLocalTimeStamp = true
agentSeries.sinks.HDFS_1.hdfs.fileType = DataStream
agentSeries.sinks.HDFS_1.hdfs.batchSize = 10

# Desactivar rollover para mantener los archivos originales
agentSeries.sinks.HDFS_1.hdfs.rollSize = 0
agentSeries.sinks.HDFS_1.hdfs.rollInterval = 0
agentSeries.sinks.HDFS_1.hdfs.rollCount = 0
agentSeries.sinks.HDFS_1.hdfs.idleTimeout = 0

# propiedades del sumidero 2
agentSeries.sinks.HDFS_2.type = hdfs
agentSeries.sinks.HDFS_2.channel = channelSensor
```

```
agentSeries.sinks.HDFS_2.hdfs.path = hdfs://quickstart.cloudera:8020/user/WDN/
    rawData/pressure/%{basename}
agentSeries.sinks.HDFS_2.hdfs.filePrefix = press
agentSeries.sinks.HDFS_2.hdfs.fileSuffix = .csv
agentSeries.sinks.HDFS_2.hdfs.useLocalTimeStamp = true
agentSeries.sinks.HDFS_2.hdfs.fileType = DataStream
agentSeries.sinks.HDFS_2.hdfs.batchSize = 10

# Desactivar rollover para mantener los archivos originales
agentSeries.sinks.HDFS_2.hdfs.rollSize = 0
agentSeries.sinks.HDFS_2.hdfs.rollInterval = 0
agentSeries.sinks.HDFS_2.hdfs.rollCount = 0
agentSeries.sinks.HDFS_2.hdfs.idleTimeout = 0
```

# Apéndice B

# Oozie

```
<workflow-app name="WDN" xmlns="uri:oozie:workflow:0.5">
<start to="spark-2d8a"/>
<kill name="Kill">
  <message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</
      message>
</kill>
<action name="spark-2d8a">
  <spark xmlns="uri:oozie:spark-action:0.2">
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>${nameNode}</name-node>
    <master>yarn</master>
    <mode>client</mode>
    <name>MySpark</name>
    <jar>interpolated</jar>
    <file>/user/WDN/oozie/interpolate.scala#interpolate.scala</file>
  </spark>
  <ok to=" T1extraerFecha"/>
  <error to="Kill"/>
</action>
<action name=" T1extraerFecha" cred="hive2">
  <hive2 xmlns="uri:oozie:hive2-action:0.1">
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>${nameNode}</name-node>
    <configuration>
      <property>
        <name>hive.exec.dynamic.partition</name>
        <value>true</value>
      </property>
      <property>
        <name>hive.exec.dynamic.partition.mode</name>
        <value>nonstrict</value>
      </property>
      <property>
        <name>hive.input.dir.recursive</name>
        <value>true</value>
      </property>
      <property>
        <name>hive.mapred.supports.subdirectories</name>
        <value>true</value>
      </property>
    </configuration>
    <jdbc-url>jdbc:hive2://quickstart.cloudera:10000/default</jdbc-url>
    <script>${wf:appPath()}/ T1extraerFecha.hql</script>
  </hive2>
  <ok to="T2incluirWeekday"/>
  <error to="Kill"/>
</action>
<action name="T2incluirWeekday" cred="hive2">
  <hive2 xmlns="uri:oozie:hive2-action:0.1">
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>${nameNode}</name-node>
```

```
51      <jdbc-url>jdbc:hive2://quickstart.cloudera:10000/default</jdbc-url>
52  <script>${wf:appPath()}/T2incluirWeekday.hql</script>
53    </hive2>
54    <ok to="T3incluirHoliday"/>
55    <error to="Kill"/>
56  </action>
57  <action name="T3incluirHoliday" cred="hive2">
58    <hive2 xmlns="uri:oozie:hive2-action:0.1">
59      <job-tracker>${jobTracker}</job-tracker>
60      <name-node>${nameNode}</name-node>
61      <jdbc-url>jdbc:hive2://quickstart.cloudera:10000/default</jdbc-url>
62      <script>${wf:appPath()}/T3incluirHoliday.hql</script>
63    </hive2>
64    <ok to="T4unionSensores"/>
65    <error to="Kill"/>
66  </action>
67  <action name="T4unionSensores" cred="hive2">
68    <hive2 xmlns="uri:oozie:hive2-action:0.1">
69      <job-tracker>${jobTracker}</job-tracker>
70      <name-node>${nameNode}</name-node>
71      <jdbc-url>jdbc:hive2://quickstart.cloudera:10000/default</jdbc-url>
72      <script>${wf:appPath()}/T4unionSensores.hql</script>
73    </hive2>
74    <ok to="T5crearPlano"/>
75    <error to="Kill"/>
76  </action>
77  <action name="T5crearPlano" cred="hive2">
78    <hive2 xmlns="uri:oozie:hive2-action:0.1">
79      <job-tracker>${jobTracker}</job-tracker>
80      <name-node>${nameNode}</name-node>
81      <jdbc-url>jdbc:hive2://quickstart.cloudera:10000/default</jdbc-url>
82      <script>${wf:appPath()}/T5crearPlano.hql</script>
83    </hive2>
84    <ok to="End"/>
85    <error to="Kill"/>
86  </action>
87  <end name="End"/>
88  </workflow-app>
```

# Apéndice C

# Breeze

```
import org.apache.spark.rdd.RDD
import breeze.interpolation._
import breeze.linalg._
import java.io._


// Directorio para los datos
val PATH = "/user/WDN/rawData/senso1/"
val DATA_SERIE = "/*"
val ATTR_HOUR = 0
val ATTR_FORESCAST = 1
val ATTR_MEASURED = 2


// Creamos la lista de archivos que contiene el Directorio
def getListOfFiles(path: String):List[File] = {
val d = new File(path)
if (d.exists && d.isDirectory) {
d.listFiles.filter(_.isFile).toList
} else {
List[File]()
}
}
/* Función que lee cada archivo y los transforma a RDD */
def readData(dir: String): (RDD[Array[String]]) = {
// Leemos datos y creamos RDD de String
var data_serie:RDD[String] = sc.textFile(PATH + dir + DATA_SERIE)

// Quitamos el encabezado
var serie_filtered:RDD[String] = data_serie.filter(line => !line.contains("Category
    "))

//Dividimos por comas
var data:RDD[Array[String]] =  serie_filtered.map(line => line.split (","))
println(PATH + dir + DATA_SERIE)
(data)
}

def interpolation(data: RDD[Array[String]],dir: String): Unit = {
// Extraemos cada atributo para obtener la serie
var serie_hour:RDD[String]=data.map{a => a(ATTR_HOUR)}
var serie_measured:RDD[String]=data.map{a => a(ATTR_MEASURED)}

// Quitamos los valores nulos de la serie (si los hay) y creamos el DenseVector "y"
var vector:RDD[Double] = serie_measured.filter(n => !n.contains("null")).map{ x =>
    x.toDouble }
var y = new DenseVector(vector.collect)

// Para crear el DenseVector "x" obtenemos las posiciones de los valores no nulos,
    con ayuda de index
```

```
48 var index:RDD[Double] = serie_measured.zipWithIndex.filter((ind: (String, Long)) =>
       ind._1 != "null").map{case (k,v) => (v).toDouble}
49 var x = new DenseVector(index.collect)
50
51 // Usamos la funcion "LinearInterpolator" de la biblioteca BREEZE
52 var interpolated = LinearInterpolator(x, y)
53
54 // Un Time series completo tiene 144 valores, por eso solicitamos la interpolación
       para un DenseVector 0-143
55 var ts = DenseVector.range(0,144,1)
56 var out = interpolated(ts)
57
58 // creamos un Map con el valor inicial hour y el resultado de la interpolación
59 var serie_out = (serie_hour.toArray zip out.toArray).toMap
60 sc.parallelize(serie_out.toSeq).saveAsTextFile("hdfs://quickstart.cloudera:8019/
       user/WDN/TO/"+dir)
61
62 }
63
64 /************************************************
65 ****************** LECTURA DATOS *****************
66 ************************************************/
67 val Folder = getListOfFiles(PATH)
68
69 for (i <- 0.until(Folder.length)) {
70   var dir = Folder(i)
71   data = readData(dir)
72
73   data.map{a =>
74   if(a(ATTR_MEASURED).contains("null")){
75     interpolation(data,dir)
76     println("contiene null")
77   }
78   else{
79     println("NO contiene null")
80     var hour=a(ATTR_HOUR)
81     var measured = a(ATTR_MEASURED).toDouble
82     var serie_out = (hour.toArray zip measured.toArray).toMap
83     sc.parallelize(serie_out.toSeq).saveAsTextFile("hdfs://quickstart.cloudera
         :8019/user/WDN/TO/"+dir)
84   }
85   }
86 }
```

# Apéndice D

# Creación de Tablas Hive

```
CREATE TABLE load_s1
(
hour        STRING,
measured    STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/WDN/T0/sensor1'

CREATE TABLE load_s2
(
hour        STRING,
measured    STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/WDN/T0/sensor2'

CREATE EXTERNAL TABLE sensor1
(
hour        STRING,
measured     DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T1/sensor1'

CREATE EXTERNAL TABLE sensor2
(
hour        STRING,
measured     DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T1/sensor2'

CREATE EXTERNAL TABLE sensor1_T2
(
time      STRING,
month     STRING,
day       STRING,
hour      STRING,
weekday   STRING,
measured  DOUBLE
)
```

```
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T2/sensor1';

CREATE EXTERNAL TABLE sensor2_T2
(
time       STRING,
month      STRING,
day        STRING,
hour       STRING,
weekday    STRING,
measured   DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T2/sensor2';

CREATE EXTERNAL TABLE sensor1_T3
(
time       BIGINT,
month      INT,
day        INT,
hour       INT,
workingday  INT,
holiday    INT,
measured   DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T3/sensor1';

CREATE EXTERNAL TABLE sensor2_T3
(
time       BIGINT,
month      INT,
day        INT,
hour       INT,
workingday  INT,
holiday    INT,
measured   DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/T3/sensor2';

CREATE EXTERNAL TABLE sensors
(
time       BIGINT,
month      INT,
day        INT,
hour       INT,
workingday INT,
demand   DOUBLE,
pressure   DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/refinedData/sensors';

CREATE EXTERNAL TABLE holiday
(
holiday       STRING
)
LOCATION '/user/WDN/rawData/holiday';

```

```
CREATE EXTERNAL TABLE consumption
(
time            BIGINT,
month           INT,
hour            INT,
day             INT,
workingday      INT,
season          INT,
demand          DOUBLE,
pressure        DOUBLE
)
PARTITIONED BY (date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/WDN/refinedData/consumption'
```