Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Evaluación de plataformas de datos para Big Data y Open Data

Evaluation of data platforms for Big Data and Open Data

Mario Alfonso Clavijo Mojica

La Laguna, 24 de mayo de 2023

D. **José Luis Roda García**, con N.I.F. 43.356.123-L profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Carlos Domínguez García**, con N.I.F. 42.238.865-D gestor de datos en Cajasiete, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Evaluación de plataformas de datos para Big Data y Open Data"

ha sido realizada bajo su dirección por D. **Mario Alfonso Clavijo Mojica**,

con N.I.F. 10.258.511-M.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 24 de mayo de 2023

Agradecimientos

Agradezco a las personas que me han acompañado durante este viaje.
También quiero dar un gran reconocimiento a todos aquellos profesores que han logrado transmitir sus conocimientos.

Licencia

Resumen

El propósito de este trabajo ha sido la evaluación de herramientas para el tratamiento adecuado de los datos masivos que están a disposición de cualquier persona. De esta manera no solo se está dando un valor añadido a la unificación de herramientas que se encuentran en el ámbito del "Big Data", sino que también se está potenciando el valor de los datos abiertos ("Open Data").

Se han evaluado diferentes herramientas entre las que destacan Apache Hadoop, Spark, Hive y Airflow, y se ha desarrollado un ejemplo práctico usando estas herramientas y datos abiertos de balances de la entidad Cajasiete. Posteriormente se ha desarrollado un proceso de "machine learning" para predecir futuros valores de estos datos y se ha creado un "dashboard" que sirve a la entidad para visualizar fácilmente los datos. En definitiva, se ha configurado toda una plataforma de datos en la que se puede llevar a cabo cualquier proceso de Big Data.

Palabras clave: Big Data, Open Data, Machine Learning, Plataforma de Datos.

Abstract

The purpose of this work has been the evaluation of tools for the adequate treatment of massive data that are available to anyone. In this way, not only is added value being given to the unification of tools that are in the field of Big Data, but the value of Open Data is also being promoted.

Different tools have been evaluated, among which Apache Hadoop, Spark, Hive and Airflow stand out, and a practical example has been developed using these tools and open data from the Cajasiete entity's balance sheets. Subsequently, a machine learning process has been developed to predict future values of this data and a dashboard has been created that serves the entity to easily visualize the data. In short, an entire data platform has been configured in which any Big Data process can be carried out.

Keywords: Big Data, Open Data, Machine Learning, Data Platform.

Índice general

Capítulo 1 Introducción 1
1.1 Motivación 1
1.2 Objetivos 1
1.3 Antecedentes 2
Capítulo 2 Fundamentos teóricos 3
2.1 Big Data 3
2.2 Open Data 3
2.3 Análisis exploratorio 4
2.4 Data Warehouse 5
2.5 Workflow 5
2.6 Aprendizaje automático 5
2.6.1 Modelo de series temporales 6
2.6.2 Modelo ARIMA 6
Capítulo 3 Tecnologías 9
3.1 Docker 9
3.2 Apache Hadoop 10
3.2.1 Hadoop clúster 10
3.2.2 Hadoop distributed file system (HDFS) 10
3.2.3 MapReduce 11
3.3 Apache Hive 12
3.3.1 Data Flow en Hive 12
3.4 Apache Spark 13
3.4.1 Características de la arquitectura 13
3.5 Apache Hue 14
3.6 Apache Airflow 14
3.7 JupyterLab 15
Capítulo 4 Desarrollo 16
4.1 Pruebas de concepto de las tecnologías 16
4.1.1 Entendiendo Apache Hadoop 16
4.1.2 Entendiendo Apache Hive y Apache Spark 17
4.1.3 Entendiendo Apache Airflow y Apache Hue 19
4.1.4 Entendiendo JupyterLab 21
4.2 Caso específico 23
4.2.1 Extracción, transformación y almacenamiento de los datos 23
4.2.2 Análisis exploratorio de los datos 26
4.2.3 Modelo ARIMA 28
4.2.4 Visualización de los datos 33

Capítulo 5 Conclusiones y líneas futuras 35
5.1 Conclusiones 35
5.2 Líneas futuras 35
Capítulo 6 Summary and Conclusions 36
6.1 Conclusions 36
6.2 Future work 36
Capítulo 7 Presupuesto 37
7.1 Costes de hardware 37
7.2 Costes de recursos humanos 37
7.3 Costes totales 37
Capítulo 8 Apéndice A 38
8.1 Repositorio GitHub 38

Índice de Figuras

Figura 3.1: Arquitectura HDFS de Hadoop 11
Figura 3.2: Ejemplo del proceso MapReduce en Hadoop 11
Figura 3.3: Flujo de trabajo al ejecutar una consulta en Hive 12

Figura 4.1: Docker Desktop con los servicios de Apache Hadoop 17
Figura 4.2: Resultado de la ejecución del archivo jar en Hadoop 17
Figura 4.3: Contenido de un fichero hql para la creación de una tabla Hive 18
Figura 4.4: Información sobre Apache Spark 18
Figura 4.5: Resultados al ejecutar comandos haciendo uso de Spark 19
Figura 4.6: Función Python para la extracción de tweets 20
Figura 4.7: Función Python para la creación y carga de datos en Hive 20
Figura 4.8: Ejemplo de creación de una Spark Session con el servicio Hive 21
Figura 4.9: Bloques de código en JupyterLab para la extracción y limpieza de los datos 22
Figura 4.10: Carga de la información en la base de datos Hive 22
Figura 4.11: Función Python para la extracción de los balances del 2018 y 2019 23
Figura 4.12: Función Python para la limpieza de los datos 24
Figura 4.13: Función Python para la conexión y creación de tablas en Hive 24
Figura 4.14: Función Python para la conexión e inserción de los datos en Hive 25
Figura 4.15: DAG del caso específico en Apache Airflow 25
Figura 4.16: Interfaz de usuario web de Apache Hue 25
Figura 4.17: Metadatos de los balances del 2018 y 2019 26
Figura 4.18: Gráfico de barras para el total pasivo por entidad 27
Figura 4.19: Código Python para la aplicación del IQR 27
Figura 4.20: Resultado tras aplicar el proceso IQR 28
Figura 4.21: Gráfico de los datos por trimestre 28
Figura 4.22: Valores sin aplicar ninguna diferencia 29
Figura 4.23: Valores con segunda diferencia 29
Figura 4.24: Gráfico de los valores reales y las predicciones con el modelo ARIMA manual 30
Figura 4.25: Gráfico con los valores reales, modelo ARIMA manual y automático 31
Figura 4.26: Gráfica con los valores reales y las predicciones mediante el uso de conjuntos 31
Figura 4.27: Gráfico con las predicciones para años posteriores 32
Figura 4.28: Página principal del Power BI que muestra información general 33
Figura 4.29: Página del Power BI que muestra el ranking de las entidades 34

Índice de tablas

Tabla 4.1: Valores de la eficiencia de los modelos 32
Tabla 7.1: Costes de hardware 37
Tabla 7.2: Costes de recursos humanos 37
Tabla 7.3: Costes totales 37

Capítulo 1
Introducción

1.1 Motivación

Actualmente con el uso constante del internet de forma general y por consiguiente el aumento inminente de aplicaciones web, redes sociales, etc. hace que la generación masiva de datos por estos medios ya sea en el sector público o privado, sean muy complicados de mantener como se solía hacer. Hoy en día, los retos que nacen de estos datos masivos van más allá que solo almacenarlos, debemos ser capaces de potenciar todos esos datos procesándolos rápidamente y por supuesto teniendo en cuenta el factor veracidad. Es por ello por lo que los proyectos Big Data requieren arquitecturas e infraestructuras diseñadas para dar solución a estos retos.

El fin último del Big Data [1] no es almacenar una gran cantidad de información, sino establecer una estrategia para saber qué se hace con estos datos. La información puede ser tomada desde cualquier fuente, es por ello, que se debe nombrar la utilidad del Open Data [2] que en términos muy amplios busca poner a disposición de los ciudadanos conjuntos de datos en formatos reutilizables y sin costes.

1.2 Objetivos

El presente trabajo tiene tres objetivos importantes:

- Estudio de las diferentes alternativas de arquitecturas de datos, para posteriormente diseñar una arquitectura Big Data en contenedores.
- Desarrollar un workflow que nos permita la transformación y la ingesta de Open Data.
- Demostrar con un ejemplo práctico la utilidad de estas herramientas.

1.3 Antecedentes

El campo del Big Data ha experimentado un exponencial desarrollo en las últimas décadas. Como consecuencia, el Big Data ha adquirido un papel fundamental en diferentes áreas, incluyendo la ciencia, la industria, el gobierno y la investigación académica. Esto es así debido a la necesidad de tomar decisiones basadas en evidencias.

Sin ir más lejos, en el ámbito de la salud y el medioambiente, se han utilizado técnicas de Big Data para predecir los niveles de polución atmosférica en Tenerife [3].

Por otro lado, también podemos ver plataformas con especial fijación en el Big Data como, por ejemplo:

1. Apache Hadoop: es una herramienta Big Data considerada como el framework estándar para el almacenamiento de grandes volúmenes de datos, aunque también se emplea para el análisis y procesamiento.
2. Apache Spark: es otro framework que permite el procesamiento de datos estáticos o en tiempo real. Su gran característica es la disminución del tiempo de procesamiento.
3. Docker: es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas del sistema, código, etc.

Ante este panorama, en el presente trabajo de fin de grado se propone explorar técnicas y enfoques para el procesamiento y análisis de Big Data, con el objetivo de superar las limitaciones actuales y contribuir al avance en este campo.

Capítulo 2
Fundamentos teóricos

2.1 Big Data

Podemos definir el Big Data como un conjunto de datos extremadamente grandes y complejos que sobrepasan la capacidad de las herramientas convencionales de procesamiento y análisis de datos. Estos conjuntos de datos suelen caracterizarse por su volumen, velocidad y variedad, también conocido como las 3V del Big Data. El **volumen** se refiere a la cantidad de datos generados y almacenados, la **velocidad** hace referencia a la velocidad a la que los datos se generan y se deben procesar en tiempo real, y la **variedad** se enfoca en la diversidad de formatos y tipos de datos, como texto, vídeos, imágenes, redes sociales, sensores, entre muchos otros.

El Big Data se caracteriza por su potencial para producir información valiosa y conocimientos significativos a través del análisis y la correlación de grandes volúmenes de datos. Está claro que estamos acostumbrados a saber lo que pasó, pero en la actualidad nos interesa más conocer lo que pasará. Es por ello, que actualmente también debemos hablar del término "Big Analytics" [4], debido a que nos permite la aplicación de procesos matemáticos complejos con técnicas de "machine learning", como, por ejemplo, regresión lineal, redes neuronales, etc.

2.2 Open Data

Al momento de hablar del Big Data es importante reconocer la existencia del término Open Data. Este término se refiere a la idea de que ciertos datos deberían estar disponibles de forma abierta y accesible para que cualquier persona pueda utilizarlos. En definitiva, se trata de datos que son de dominio público o que se comparten bajo licencias abierta que permiten su libre uso.

El Open Data, promueve la transparencia, la colaboración y la participación ciudadana al poner a disposición información que puede ser utilizada para diversos fines, como la

investigación, la toma de decisiones informadas y el desarrollo de aplicaciones y servicios innovadores [5].

2.3 Análisis exploratorio

En el contexto del Big Data y el Open Data, el análisis exploratorio de los datos (EDA) desempeña un papel importante para descubrir información inicial de los datos para posteriormente poder tomar acciones de limpieza, etc., y así obtener patrones, identificar tendencia y extraer información valiosa de conjuntos de datos masivos y abiertos. A medida que la cantidad y diversidad de los datos siguen aumentando de manera exponencial, el EDA se convierte en una herramienta esencial para comprender y potenciar los datos [6].

El análisis exploratorio de los datos en el ámbito del Big Data implica el examen minucioso de los conjuntos de datos para descubrir información oculta y generar conocimientos adicionales. Al explorar los datos, podemos llegar a encontrar anomalías y relaciones causales, es por ello por lo que debemos hacer **detección de Outliers** [7].

- El objetivo de realizar este proceso de detección de Outliers es debido a que puede afectar considerablemente a los resultados. Los outliers pueden significar o apuntar a varias cosas:
    - Errores: por ejemplo, si estamos cargando una serie de datos relativos a las edades de un grupo de personas y hay un registro que supera el intervalo normal, esto posiblemente significaría un error.
    - Límites: por otro lado, pueden existir valores que se escapan del grupo medio de valores que nos interesan.
    - Puntos de interés: hace referencia a la existencia de valores anómalos los cuales debemos detectar como parte de nuestro objetivo del modelo que estemos realizando.

En el contexto del Open Data, el EDA obtiene una importancia aún mayor, ya que estos conjuntos de datos se caracterizan por su diversidad y complejidad. Además, al tratarse de datos abiertos muchas veces puede intervenir el factor humano al momento de generar esos datos, por lo que es importante hacer una limpieza.

En definitiva, el EDA desempeña un papel crucial tanto en el ámbito del Big Data como en el del Open Data. Permitiéndonos descubrir conocimientos ocultos, revelar relaciones y tendencias significativas.

2.4 Data Warehouse

Un Data Warehouse, o un almacén de datos, es una manera de solventar la necesidad del almacenamiento y gestión de datos [8].

El objetivo principal de un Data Warehouse es brindar un entorno centralizado y estructurado para consolidar datos de múltiples sistemas y fuentes heterogéneas. Además de esto, nos permite promover el análisis de datos y la generación de consultas complejas. Algunas herramientas son Snowflake, Oracle Database, etc.

En resumen, un Data Warehouse es una infraestructura esencial en la gestión y análisis de datos empresariales. Proporciona una plataforma centralizada y estructurada para almacenar, organizar y analizar grandes conjuntos de datos.

2.5 Workflow

En el ámbito del Big Data, los workflows, o flujos de trabajo, desempeñan un papel fundamental para la gestión eficiente y efectiva de los procesos de análisis de datos. Los workflows nos permiten automatizar y orquestar tareas y operaciones complejas en un entorno de Big Data, lo que contribuye a maximizar la productividad, optimizar los recursos y garantizar la calidad de los resultados que esperamos [9].

Un workflow en el contexto del Big Data se refiere a una secuencia lógica de pasos y actividades interrelacionadas que se van a ejecutar ya sea de manera secuencial o teniendo en cuenta la ejecución de diferentes actividades previamente.

Como dijimos anteriormente, la importancia de los workflows radica en varios aspectos. En primer lugar, los workflows permiten la automatización de tareas repetitivas y complejas, lo que ahorra tiempo y reduce errores. Y, en segundo lugar, nos permite programar el lanzamiento de estas tareas según nuestra conveniencia.

2.6 Aprendizaje automático

El machine learning permite obtener un gran aprovechamiento en el análisis del Big Data, por lo que mediante este aprendizaje automático podemos potenciar el valor de los datos que tenemos [10].

Debido a que el ML puede procesar y analizar grandes volúmenes de datos de manera eficiente, esto permite descubrir patrones, tendencia y relaciones ocultas que serían difíciles de encontrar utilizando metodologías tradicionales. Por otro lado, ML permite automatizar tareas complejas de análisis y toma de decisiones en el contexto del Big Data. Al entrenar

modelos, se pueden aprender a identificar patrones complejos y tomar decisiones basadas en esos patrones.

Cuando hablamos de Big Data debemos pensar en mejora continua y adaptabilidad debido a que los datos son dinámicos y cambian con el tiempo. En definitiva, el machine learning es importante en Big Data porque permite procesar y analizar datos, automatizar procesos complejos y adaptarse de manera continua a medida que se obtienen nuevos datos. Estas capacidades contribuyen a potenciar el Big Data y obtener información valiosa.

Algunos de los modelos más utilizados son:

- Regresión lineal: en este caso se busca predecir un valor numérico continuo, su objetivo es encontrar la mejor línea recta que se ajuste a lo datos y pueda utilizarse para poder predecir.
- Árboles de decisión: como su nombre indica, se utiliza la estructura de un árbol para la toma de decisiones, esto se hace debido a que se tiene que representar relaciones complejas existentes.
- Bosques aleatorios: es una técnica que utiliza un conjunto de árboles aleatorios.
- Support vector machine (SVM): se basa en la búsqueda de un hiperplano que mejor se separe de las diferentes clases o que se ajuste a los datos en caso de regresión.
- Artificial neural network (ANN): estos modelos están basados en la estructura y funcionamiento del cerebro humano. Consiste en capas de nodos interconectados, donde cada nodo realiza una operación matemática.
- Extreme learning machine (ELM): son modelos de aprendizaje automático.

2.6.1 Modelo de series temporales

El análisis de series temporales es una técnica utilizada en el campo del ML para analizar y predecir datos que varían o cambian a lo largo del tiempo. Este modelo se aplica en una amplia gama de áreas, como la economía, las finanzas, el clima y muchas otras [11].

Una serie temporal hace referencia a una secuencia de observaciones grabadas en intervalos de tiempo regulares. Estas observaciones pueden ser valores numéricos, como los activos de una empresa. El objetivo del análisis es descubrir patrones, tendencia y estacionalidad de los datos, y utilizar esta información para predecir.

Se puede decir, que el análisis de series temporales es importante poque permite comprender las relaciones de los datos a través del tiempo, identificar patrones y realizar predicciones precisas.

2.6.2 Modelo ARIMA

El modelo **ARIMA** (Autoregressive Integrated Moving Average) es un modelo ampliamente utilizado en el análisis de series temporales para describir y predecir datos

secuenciales que pueden mostrar patrones autorregresivos, estacionalidad y tendencias [12]. En definitiva, se busca obtener los hiperparámetros (p, d, q) para el modelo.

El nombre "ARIMA" se deriva de sus tres componentes principales: Autoregressive (AR), Integrated (I) y Moving Average (MA). Cada uno de estos componentes captura diferentes aspectos del comportamiento de la serie temporal.

Componente autoregresivo (AR):

El componente autorregresivo se refiere a la relación entre una observación actual y las observaciones pasadas. En un modelo AR, se asume que el valor actual de la serie temporal se puede explicar mediante una combinación lineal de sus valores anteriores, multiplicados por coeficientes llamados parámetros autorregresivos. La "p" en ARIMA representa el orden del componente AR, es decir, cuántas observaciones pasadas se deben tener en cuenta.

Componente de media móvil (MA):

El componente de media móvil se utiliza para capturar la influencia de los errores residuales pasados en la predicción actual. En un modelo MA, se asume que el valor actual de la serie temporal se puede explicar mediante una combinación lineal de los errores residuales pasados, multiplicados por coeficientes llamados parámetros de media móvil. La "q" en ARIMA representa el orden del componente MA, es decir, cuántos errores residuales pasados se deben considerar.

Componente integrado (I):

El componente integrado se refiere a la diferenciación de la serie temporal para hacerla estacionaria. Si una serie temporal muestra tendencias o cambios de nivel en el tiempo, no se ajusta bien a un modelo AR o MA directamente. En cambio, se aplica una operación de diferenciación para eliminar estas tendencias. La "d" en ARIMA representa el número de veces que se realiza la diferenciación para hacer que la serie temporal sea estacionaria.

El modelo ARIMA se define entonces por los valores de p, d y q, que representan el orden de los componentes AR, I y MA, respectivamente. La selección adecuada de estos valores se puede determinar mediante técnicas como la función de autocorrelación (ACF) y la función de autocorrelación parcial (PACF), así como mediante pruebas estadísticas y análisis visual de los datos.

Una vez que se ha ajustado el modelo ARIMA a los datos, se puede utilizar para hacer predicciones futuras. El modelo tiene en cuenta la dependencia temporal, la estacionalidad y los errores residuales pasados para generar pronósticos.

En resumen, el modelo ARIMA es una herramienta poderosa para el análisis y la predicción de series temporales. Combina componentes autoregresivos, de media móvil y de integración para capturar diferentes aspectos del comportamiento de la serie temporal y proporcionar pronósticos precisos.

Cuando trabajamos con modelos ARIMA, es importante evaluar su rendimiento utilizando diferentes métricas de error. Para ello, es crucial nombrar el MAE, el MAPE y el RMSE.

- MAE (Mean Absolute Error): este valor representa el promedio de las diferencias absolutas entre las predicciones del modelo y los valores reales. Es una medida de la magnitud promedio de los errores del modelo, sin tener en cuenta su dirección. Cuanto menor sea el valor del MAE, mejor será el ajuste del modelo.
    - Un valor de MAE de 0 indica un ajuste perfecto del modelo, lo cual es un poco común en la práctica, pero como dijimos anteriormente, cuanto menor sea su valor, mejor.
- MAPE (Mean Absolute Percentage Error): el MAPE es similar al MAE, pero expresa el error como un porcentaje relativo al valor real. Calcula el promedio de las diferencias absolutas porcentuales entre las predicciones y los valores reales.
- RMSE (Root Mean Square Error): el RMSE representa la raíz cuadrada del promedio de los errores al cuadrado entre las predicciones y los valores reales. El RMSE penaliza los errores más grandes de manera más significativa que el MAE, ya que se basa en la raíz cuadrada.

Capítulo 3
Tecnologías

En este capítulo se explicarán cada una de las tecnologías que se han utilizado para la creación de un entorno Big Data.

3.1 Docker

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute [13].

Hemos elegido utilizar Docker en lugar de una máquina virtual debido a las ventajas que nos da.

- Eficiencia de recursos: Docker utiliza una arquitectura de contenedores ligeros que comparten el mismo sistema operativo. A diferencia de las VM, que requieren un sistema operativo para cada instancia.
- Arranque rápido y escalabilidad: los contenedores de Docker se inician rápidamente debido a lo que comentamos anteriormente. Además, en cuanto a escalabilidad, nosotros podemos ir agregando contenedores que en un principio no teníamos contemplados.
- Portabilidad: este aspecto es importante, debido a que nosotros podemos utilizar los contenedores en cualquier entorno que tenga Docker instalado, independientemente del sistema operativo.

Si bien es cierto que una VM también es una opción válida, Docker nos proporciona una flexibilidad increíble.

Cuando hablamos de Docker debemos tener en cuanta algunos aspectos claves, como por ejemplo que es una imagen, los puertos, etc.

- Imagen: Docker utiliza imágenes como base para crear contenedores. Una imagen tiene especificado todo lo necesario para que Docker genere el contenedor, desde el sistema operativo hasta las librerías o dependencias que necesita.
- Contenedores: un contenedor es una instancia en ejecución de una imagen. Lo bueno es que Docker nos permite comunicarnos entre contenedores haciendo que la

escalabilidad sea aún mayor. Otra cosa que podemos destacar es que, si algún contenedor necesita un puerto, nosotros podemos asignárselo, es decir, podemos mapear un puerto del contenedor a uno del host.

- Dockerfile: es un archivo de texto que contiene las instrucciones para crear una imagen, o también para modificarla.

3.2 Apache Hadoop

De manera general Hadoop es un framework open source que se utiliza para almacenar datos y ejecutar aplicaciones en clústeres de hardware básico. Como se comentó anteriormente, el Big Data hace referencia a un gran conjunto de datos que posteriormente serán analizados. Es aquí donde entra la importancia de **Hadoop** [14].

Hadoop presenta una serie de características que lo hacen relevante cuando se menciona la palabra Big Data, estas características son:

1. Tiene la capacidad de almacenar y procesar grandes conjuntos de datos, que como dijimos anteriormente no presentan la misma estructura.
2. Proporciona un poder de procesamiento muy elevado.
3. Además de persistir los datos tiene un mecanismo de tolerancia a fallos.
4. Nos permite tener flexibilidad a la hora de almacenar los datos.
5. Al ser de código abierto el framework tiene un bajo coste.
6. Podemos implantar tantos nodos como queramos para aumentar el procesamiento de los datos, por lo que nos permite escalar rápidamente.

3.2.1 Hadoop clúster

Podemos decir que Hadoop clúster es un subconjunto de clústeres de computadores. Esto hace que pueda almacenar y analizar grandes cantidades de datos ya sean estructurados o no estructurados. Además, nos permite hacer uso del procesamiento en paralelo, balancear las cargas y también establecer tolerancia a fallos. Hay que destacar que el componente "NameNode" actúa como el punto central de control, mientras que los "DataNode" son los nodos esclavos de la arquitectura.

Las funciones de Hadoop clúster son iguales a las de un clúster de computadores. Es decir, un clúster de computadores es una colección de computadores interconectados que operan como uno solo.

3.2.2 Hadoop distributed file system (HDFS)

HDFS es una unidad de almacenamiento que ayuda a distribuir la información o los datos entre los computadores disponibles y que además se guardan en bloques.

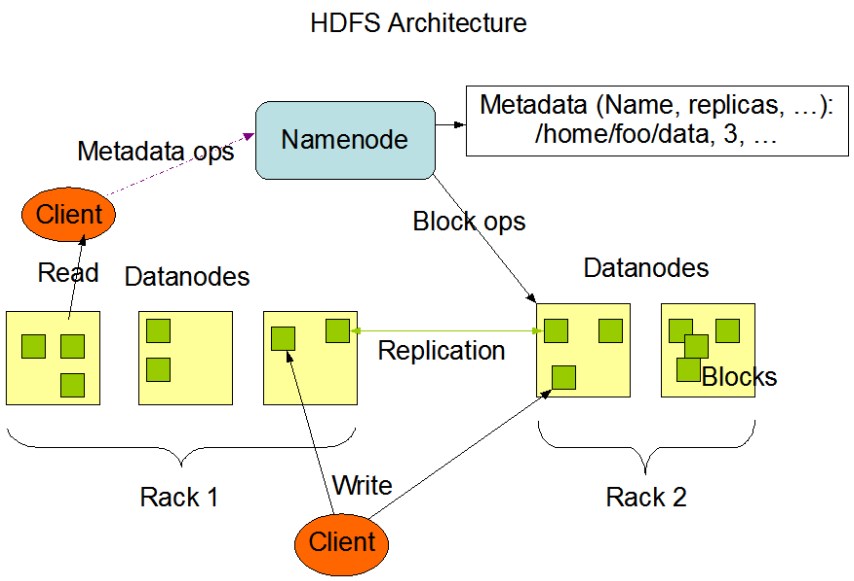

Figura 3.1: Arquitectura HDFS de Hadoop

3.2.3 MapReduce

El objetivo del MapReduce es dividir los datos en pedazos para posteriormente procesar cada uno de ellos de manera separada en nodos distintos. Después de esto, los "pequeños" resultados son combinados para seguidamente producir una salida completa.

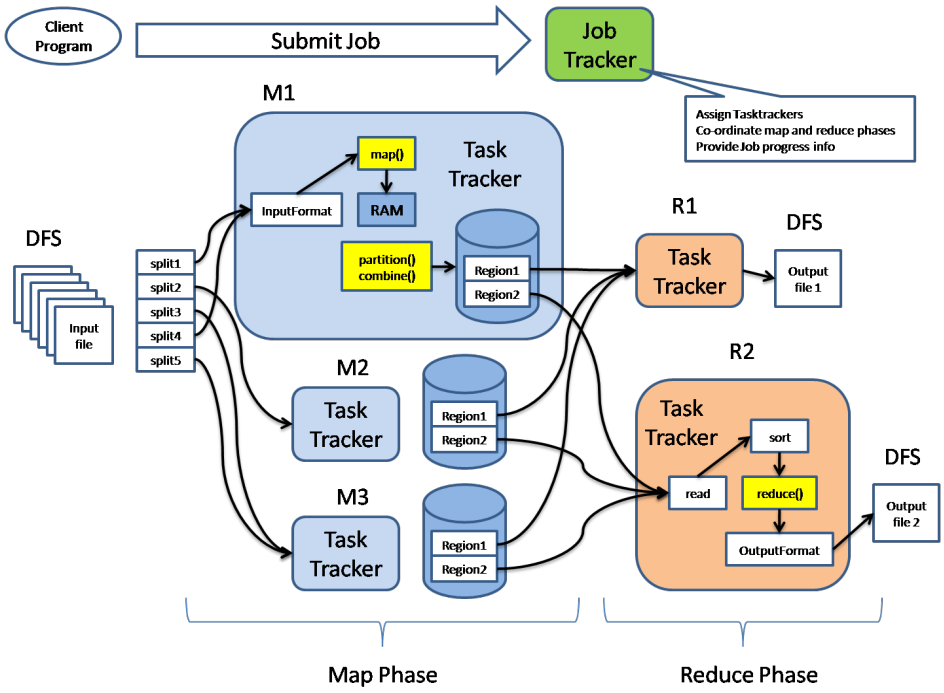

Figura 3.2: Ejemplo del proceso MapReduce en Hadoop

Apache Hadoop es un framework que como podemos observar ha quitado las limitaciones que suponían anteriormente el procesar toda la información que una empresa necesitaba. Hoy en día, Hadoop ha abierto nuevas posibilidades para la generación de nueva información y el análisis de esta. Además, si combinamos este framework junto a Docker nos da la posibilidad de manejar y desplegar, de una manera sencilla, aplicaciones complejas de Big Data.

Sin ir más lejos, la aplicación **Spotify** utiliza apache Hadoop en contenedores Docker para procesar grandes volúmenes de datos procedentes de los usuarios y **realizar análisis en tiempo real** [15].

3.3 Apache Hive

Apache Hive es una herramienta de almacenamiento de datos construida sobre Hadoop que permite a los usuarios consultar y analizar grandes conjuntos de datos almacenados en el sistema de archivos distribuidos de Hadoop (HDFS) utilizando un lenguaje similar a SQL llamado HiveQL. Proporciona una interfaz similar a SQL para consultar y administrar datos estructurados y semiestructurados almacenados en HDFS. Hive se usa normalmente para procesamiento por lotes, almacenamiento de datos y consultas ad-hoc [16].

3.3.1 Data Flow en Hive

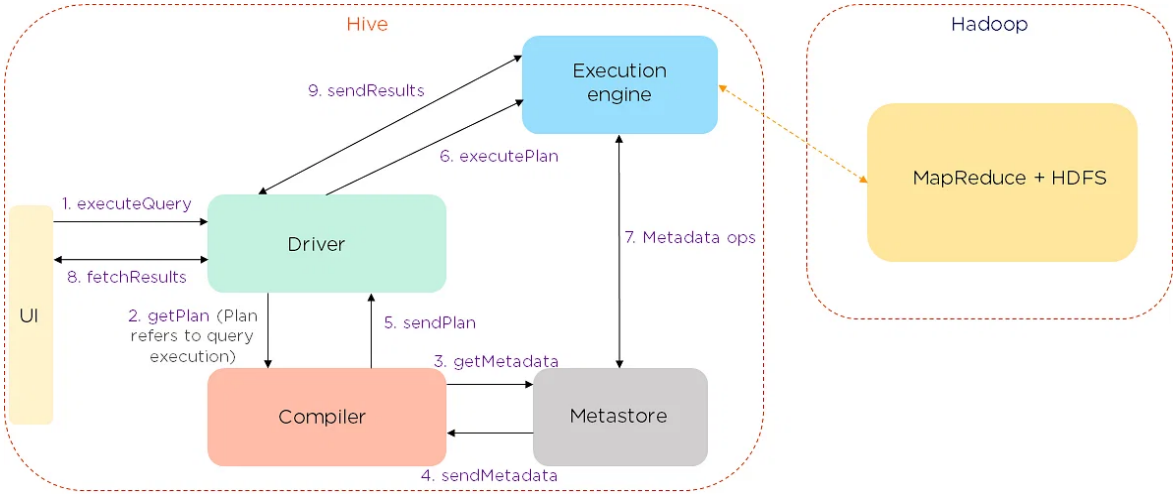

Figura 3.3: Flujo de trabajo al ejecutar una consulta en Hive

Los datos fluyen en la siguiente secuencia:

1. Ejecutamos una consulta, que va al controlador
2. Luego, el controlador pregunta por el plan, que se refiere a la ejecución de la consulta.
3. Después de esto, el compilador obtiene los metadatos del metastore.
4. El metastore responde con los metadatos.
5. El compilador recopila esta información y envía el plan de regreso al controlador.
6. Ahora, el controlador envía el plan de ejecución al motor de ejecución.
7. El motor de ejecución actúa como un puente entre Hive y Hadoop para procesar la consulta.
8. Además de esto, el motor de ejecución también se comunica bidireccionalmente con el metastore para realizar varias operaciones, como crear y soltar tablas.
9. Finalmente, tenemos una comunicación bidireccional para obtener y enviar resultados al cliente.

3.4 Apache Spark

Apache Spark, por otro lado, es un motor distribuido que permite a los usuarios procesar conjuntos de datos a gran escala de manera rápida y eficiente. Spark proporciona un marco de propósito más general y puede realizar una gama más amplia de operaciones que Hive. Se puede utilizar para el procesamiento por lotes, el procesamiento de secuencias en tiempo real, el aprendizaje automático, el procesamiento de gráficos y más [17].

3.4.1 Características de la arquitectura

La arquitectura de Apache Spark está diseñada para aprovechar al máximo los recursos de un clúster de computadoras distribuidas y permitir el procesamiento de grandes volúmenes de datos de manera eficiente. Algunas de las características más importantes de la arquitectura de Spark son:

1. **Modo de procesamiento en memoria:** Spark procesa datos en la memoria RAM en lugar del disco duro, lo que acelera significativamente el tiempo de procesamiento.
2. **Arquitectura maestro-esclavo:** Spark utiliza una arquitectura maestro-esclavo, donde el maestro coordina el trabajo y los esclavos realizan el procesamiento de datos.
3. **Resilient Distributed Datasets (RDD):** Spark almacena los datos en RDD, que son estructuras de datos inmutables distribuidas en el clúster de computadoras. Esto permite que Spark procese datos de manera distribuida y en paralelo.

4. **Transformaciones y acciones:** Spark proporciona transformaciones y acciones para manipular los datos almacenados en RDD. Las transformaciones son operaciones que transforman un RDD en otro, mientras que las acciones devuelven un resultado o un conjunto de resultados.
5. **Spark SQL:** Spark SQL es un módulo de Spark que permite el procesamiento de datos estructurados utilizando SQL.
6. **Bibliotecas integradas:** Spark viene con varias bibliotecas integradas para realizar tareas de procesamiento de datos, como procesamiento de gráficos y aprendizaje automático.

En definitiva, la arquitectura de Spark está diseñada para aprovechar al máximo los recursos de un clúster de computadoras distribuidas y permitir el procesamiento de grandes volúmenes de datos de manera eficiente. La arquitectura maestro-esclavo, el uso de RDD, las transformaciones y acciones, Spark SQL y las bibliotecas integradas son algunas de las características más importantes de la arquitectura de Spark.

3.5 Apache Hue

Hue (Hadoop User Experience) es una interfaz web de código abierto que se utiliza para interactuar con Hadoop y sus componentes, incluyendo Apache Spark, Hive, Pig, HBase y otros. Proporciona una interfaz de usuario gráfica para realizar tareas de administración de clústeres de Hadoop, como monitoreo, programación y ejecución de trabajos, y también ofrece una interfaz para interactuar con los datos almacenados en el clúster [18].

3.6 Apache Airflow

Apache Ariflow es una plataforma que sirve para programar, coordinar y supervisar flujos de trabajo (workflows) de datos [19].

En el contexto del Big Data, donde los datos se generan en volúmenes masivos y se procesan a alta velocidad, es crucial contar con un sistema que pueda gestionar y orquestar de manera eficiente todas las tareas y procesos asociados. Además, Airflow nos permite conectarnos a una infinidad de herramientas como por ejemplo Hive, Azure, Google, etc. De esta manera podemos establecer procesos de trabajos que hacen uso de diferentes fuentes de datos no están centralizados.

Cuando hablamos de la herramienta Apache Airflow debemos tener una serie de conceptos claros.

1. DAG: un DAG (Directed Acyclic Graph) es una representación visual y lógica de un flujo de trabajo.
2. Task y Operator: las “Task” son componentes internos que sirven para manejar el estado de los operadores.

De manera muy superficial podemos resumir que un DAG es un conjunto de tareas (Tasks) que tiene asociada un operator (BashOperator, PythonOperator, etc.). Para realizar la construcción de un DAG debemos conocer aspectos básicos del lenguaje de programación Python.

3.7 JupyterLab

JupyterLab es un entorno de desarrollo interactivo de código abierto que permite a los usuarios crear y compartir documentos que contienen código, texto y visualizaciones interactivas. En el contexto del Big Data, JupyterLab se utiliza comúnmente como una herramienta para el análisis y procesamiento de datos [20].

Con JupyterLab, los usuarios pueden escribir código en diferentes lenguajes de programación, como Python, R, Julia y Scala, y ejecutarlo en tiempo real. Además, JupyterLab permite la visualización de datos a través de gráficos, tablas y otros elementos visuales interactivos, lo que facilita el análisis y la exploración de grandes conjuntos de datos.

Otra ventaja de JupyterLab es que es altamente personalizable y extensible, lo que significa que los usuarios pueden agregar paquetes y extensiones para adaptarse a sus necesidades específicas. Esto lo hace ideal para trabajar con Big Data, donde los conjuntos de datos pueden ser complejos y difíciles de analizar con herramientas estándar.

En resumen, JupyterLab se utiliza en el contexto del Big Data como una herramienta para el análisis y procesamiento de datos. Permite a los usuarios escribir y ejecutar código en diferentes lenguajes de programación, visualizar datos y personalizar el entorno para adaptarse a sus necesidades específicas.

Capítulo 4
Desarrollo

Hasta el momento hemos hablado de los fundamentos teóricos básicos para el presente trabajo, seguidamente se ha explicado cada una de las tecnologías que se van a usar para poder cumplir con los objetivos.

En el presente capítulo se abordará cada una de las actividades que se fueron realizando de manera cronológica. Dentro de estas actividades, están aquellas que sirvieron como una primera toma de contacto para las tecnologías y posteriormente, aquellas que son la base del presente trabajo.

4.1 Pruebas de concepto de las tecnologías

4.1.1 Entendiendo Apache Hadoop

Una vez que se tuvo claro qué es Apache Hadoop y cuál es su estructura, se procedió a la realización de una prueba de concepto mediante la herramienta Docker. Para poder generar la instancia del contenedor se utilizó la imagen del repositorio Github "big-data-europe" [21]. Una vez que se puso en funcionamiento los contenedores asociados al servicio, como se puede observar en la figura 4.1, se procedió a realizar la carga de un archivo jar que contiene un pequeño código para contar palabras o números en el contenedor "NameNode". Esto se hizo para poder lanzar una tarea "MapReduce", para ello se tuvo que ejecutar el comando *hadoop jar <archivo jar> <clase a utilizar> input output.* A continuación, en la figura 4.2 se puede apreciar el resultado de la ejecución del comando.

Hay que destacar que una de las ventajas de Apache Hadoop es que cuenta con una interfaz de usuario web, dicha interfaz es accesible a través del puerto 50070. Por lo que podemos navegar libremente y ver la información almacenada en cada uno de los "DataNodes".

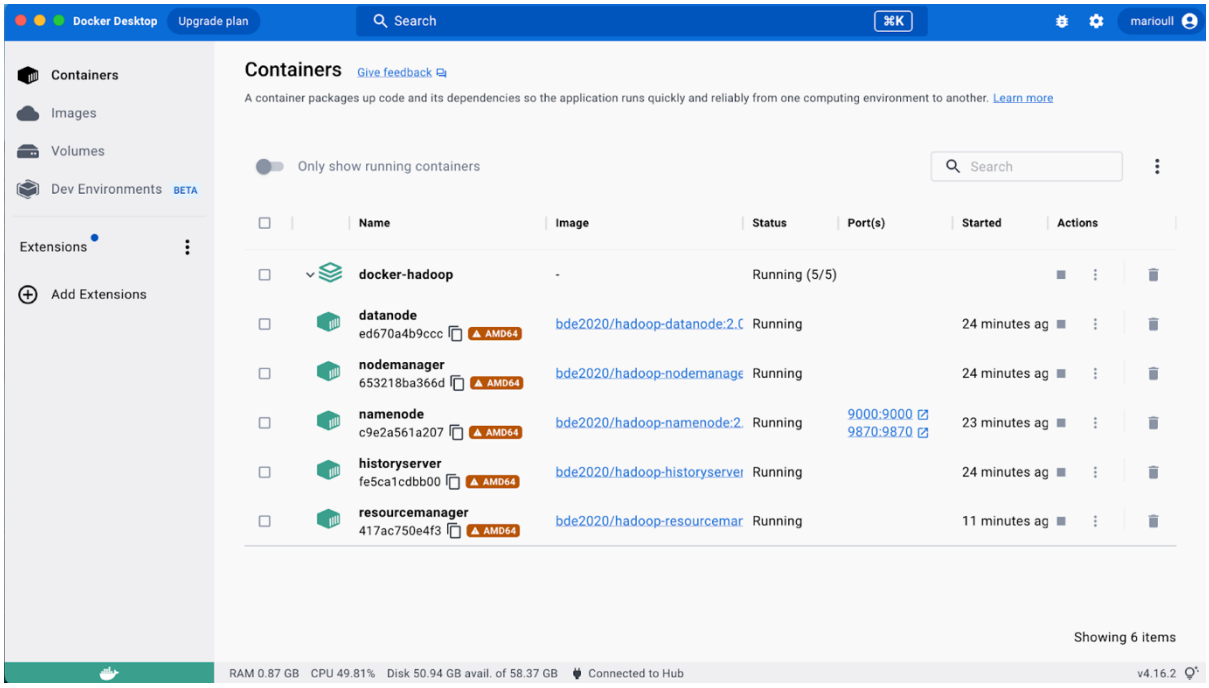

Figura 4.1: Docker Desktop con los servicios de Apache Hadoop

```
root@c9e2a561a207:/# hdfs dfs -cat /user/root/input/prueba.txt
2023-02-20 08:54:08,449 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
1 linea
2 linea
3 linea
root@c9e2a561a207:/# hdfs dfs -cat /user/root/output/part-r-00000
2023-02-20 08:54:32,127 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
1       1
2       1
3       1
linea   3
```

Figura 4.2: Resultado de la ejecución del archivo jar en Hadoop

4.1.2 Entendiendo Apache Hive y Apache Spark

Apache Hive y Apache Spark son herramientas poderosas que se utilizan en el procesamiento y análisis de Big Data, pero tienen diferentes funcionalidades y casos de uso.

Tanto Apache Hive como Apache Spark se pueden usar juntos para proporcionar una poderosa solución de procesamiento de Big Data. Hive se puede usar para administrar y consultar datos estructurados almacenados en HDFS, mientras que Spark se puede usar para tareas de procesamiento y análisis más complejas que requieren un cálculo más rápido y análisis más avanzados. Además, Spark puede leer datos de tablas de Hive y usarlos como entradas para trabajos de Spark, lo que proporciona una integración perfecta entre las dos herramientas.

Para poder entender estas dos herramientas se procedió a la creación de las instancias necesarias de los contenedores que ejecutan los servicios. Como era de esperar, necesitamos los servicios de Apache Hadoop, Apache Hive y también Apache Spark.

Una vez que se obtuvieron los servicios, se procedió a crear una tabla en Hive, a realizar una inserción y finalmente, ejecutar una consulta. Para ello se entró en el contenedor que tiene el servicio de Hive mediante el comando *docker exec -it.* Una vez estando allí, se procedió a realizar los siguientes pasos:

1. Crear una tabla mediante un fichero de extensión hql (figura 4.3)

```
create database if not exists testdb;
use testdb;
create external table if not exists employee (
  eid int,
  ename string,
  age int,
  jobtype string,
  storeid int,
  storelocation string,
  salary bigint,
  yrsofexp int
)
row format delimited
fields terminated by ','
lines terminated by '\n'
```

Figura 4.3: Contenido de un fichero hql para la creación de una tabla Hive

2. Realizar un "insert" en la tabla employee.
3. Finalmente, realizar un "select" para ver el contenido de la tabla.

Por otro lado, en cuanto a Apache Spark (figura 4.4), se procedió a realizar una serie de operaciones para trabajar con un fichero csv el cual contiene información sobre el salario de los empleados de la ciudad de Chicago.

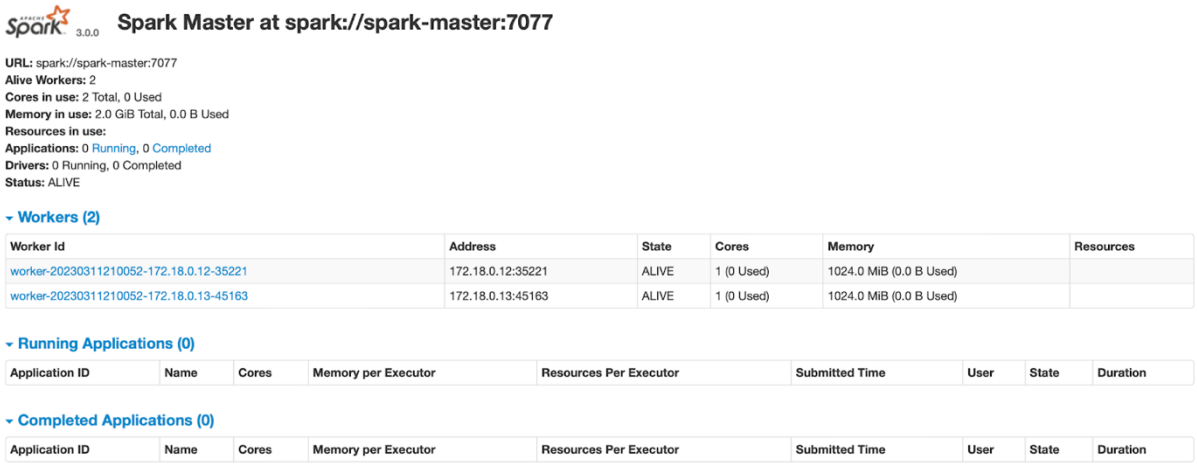

Figura 4.4: Información sobre Apache Spark

Para poder realizar las operaciones sobre el csv lo primero que se hizo fue lanzar el proceso de la "Spark-shell", una vez allí se ejecutaron los siguientes comandos:

1. val dataset = spark.read.option("header","true").csv("/opt/workspace/xzkq-xp2w.csv")
2. dataset.head()
3. dataset.select("Hourly Rate").summary().show()

La ejecución de estos comandos dio como resultado la salida que se puede apreciar en la figura 4.5.

```
[scala> dataset.head()
 res4: org.apache.spark.sql.Row = [AARON, JEFFERY M,SERGEANT,DEPARTMENT OF POLICE,F,SALARY,null,125634,null]

[scala> dataset.select("hourly_rate").summary().show()
+-------+------------------+
|summary|       hourly_rate|
+-------+------------------+
|  count|               194|
|   mean|38.624072164948444|
| stddev|11.747930450085674|
|    min|              15.4|
|    25%|              36.1|
|    50%|             39.25|
|    75%|              45.9|
|    max|             59.44|
+-------+------------------+
```

Figura 4.5: Resultados al ejecutar comandos haciendo uso de Spark

4.1.3 Entendiendo Apache Airflow y Apache Hue

Una vez que se había hecho una prueba de concepto de la mayoría de las herramientas, finalmente se procedió al estudio del funcionamiento de estas dos herramientas. Para poder adentrarme en el funcionamiento de estas plataformas lo que realicé fue un pequeño DAG en Apache Airflow para la creación de una base de datos en Apache Hive, la extracción de los datos mediante la librería snscrape y, finalmente, la carga de tweets a la base de datos. Posteriormente, se procedió a la visualización de estos datos mediante la interfaz de usuario web de Apache Hue.

El DAG está formado por un solo nodo, pero está compuesto por diferentes llamadas a funciones. Si vemos más a fondo el DAG, podemos ver que hay dos funciones, la primera, es la que podemos ver la figura 4.6 y que se encarga de traer los datos de los tweets y crear un dataframe con una estructura definida. La segunda función, figura 4.7, se encarga de

realizar una conexión con Apache Hive para poder ejecutar comandos. En este caso, utilizamos la conexión para la creación de una tabla que va a almacenar los datos, pero también la utilizamos para realizar inserciones a la tabla. Por último, realizamos el cierre de la conexión.

```python
def etl_process():
    query = "cajasiete"
    tweets = []
    User = []
    limit = 50

    for tweet in sntwitter.TwitterSearchScraper(query).get_items():

        if len(tweets) == limit:
            break
        else:
            tweets.append([tweet.date, tweet.user.id , tweet.id ,tweet.rawContent, tweet.retweetCount, tweet.likeCount, tweet.lang, tweet.user.location])
            User.append([tweet.user.id, tweet.user.username, tweet.user.profileImageUrl,tweet.user.followersCount])



    dfTweet = pd.DataFrame(tweets, columns=['datetweet', 'userid', 'tweetid', 'tweet', 'retweetcount', 'likecount', 'lang', 'locationtweet'])


    return dfTweet
```

Figura 4.6: Función Python para la extracción de tweets

```python
def create_hive_table():
    conn = hive.Connection(host="host.docker.internal", port=10000, username="hive", database="tweetsdb")
    cursor = conn.cursor()

    # Creamos una tabla en Hive
    create_table_query = """
       CREATE EXTERNAL TABLE IF NOT EXISTS tweets (
            DateTweet STRING,
            UserID STRING,
            TweetID STRING,
            Tweet STRING,
            RetweetCount STRING,
            LikeCount STRING,
            Lang STRING,
            LocationTweet STRING

            )
            ROW FORMAT DELIMITED
            FIELDS TERMINATED BY ','
            LINES TERMINATED BY '\n'
    """
    cursor.execute(create_table_query)

    datos = etl_process()

    print(datos.head())
    print(datos.columns)
    print(datos.describe())

    insert_query = "INSERT INTO TABLE tweets VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    for row in datos.itertuples(index=False):
        cursor.execute(insert_query, (row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7]))

    # Cerramos la conexión
    conn.commit()
    cursor.close()
    conn.close()

    print("Tabla creada exitosamente")
```

Figura 4.7: Función Python para la creación y carga de datos en Hive

Por otro lado, en cuanto a Apache Hue, lo que se hizo fue analizar y probar el entorno que nos proporciona. De esta manera se puedo hacer consultas sobre la base de datos creada sin necesidad de acceder directamente al contenedor que tiene el servicio de Apache Hive.

4.1.4 Entendiendo JupyterLab

En cuanto a esta herramienta lo que se hizo como primer acercamiento fue la realización de un pequeño proceso ETL de datos del clima recogidos por La Agencia Estatal de Meteorología (AEMET).

Para poder realizar esta actividad lo primero que se hizo fue establecer la conexión de Apache Hive mediante Apache Spark, tal y como se muestra en la figura 4.8.

```
spark = SparkSession\
        .builder\
        .appName("pyspark-notebook")\
        .master("spark://spark-master:7077")\
        .config("spark.executor.memory", "512m")\
        .config("hive.metastore.uris", "thrift://hive-metastore:9083")\
        .config("spark.sql.warehouse.dir", "/user/hive/warehouse")\
        .enableHiveSupport()\
        .getOrCreate()

spark.sparkContext.setLogLevel("ERROR")
```

Figura 4.8: Ejemplo de creación de una Spark Session con el servicio Hive

Seguidamente, realizamos la extracción de los datos mediante la URL que nos proporciona la AEMET, para que una vez obtenido los datos podamos realizar los cambios necesarios, como, por ejemplo, el de establecer un estándar de anotación para la fecha en que se recogen los datos (figura 4.9).

```python
def get_aemet_data():
    url = "https://www.aemet.es/es/eltiempo/observacion/ultimosdatos_C447A_datos-horarios.xls?k=coo&l=C447A&datos=det&w=0&f=temperatura&x="
    r = requests.get(url)
    df = pd.read_excel(r.content, skiprows=3)
    return df
```

```python
datos = get_aemet_data()
datos = datos.rename(columns={"Fecha y hora oficial": 'Fecha', "Temperatura (ºC)" : 'Temperatura_c',
                              "Velocidad del viento (km/h)": 'Velocidad_del_viento_km_h', "Dirección del viento": 'Direccion_del_viento',
                              "Racha (km/h)": 'Racha_km_h', "Dirección de racha":'Direccion_de_racha',
                              "Precipitación (mm)": 'Precipitacion_mm', "Presión (hPa)":'Presion_hPa',
                              "Tendencia (hPa)": 'Tendencia_hPa', "Humedad (%)": 'Humedad_percent'})
```

```python
for index, row in datos.iterrows():
    fecha_no_f = row['Fecha']
    try:
        fecha_objeto = datetime.strptime(fecha_no_f, '%Y-%m-%d %H:%M:%S')
        datos.iloc[index, datos.columns.get_loc('Fecha')] = fecha_objeto.strftime('%d/%m/%Y %H:%M')
    except ValueError:
        print(f'La fecha {fecha_no_f} no tiene el formato correcto')
```

Figura 4.9: Bloques de código en JupyterLab para la extracción y limpieza de los datos

Al momento de tener los datos ya listos para ser almacenados, se procede a realizar los siguientes pasos:

1. Seleccionar una base de datos de destino.
2. Almacenar los datos en una tabla.

Estos pasos los podemos ver la figura 4.10.

```python
spark.sql("use open_weather_datadb")
```

```
DataFrame[]
```

```python
spark_df.write.mode('overwrite').saveAsTable('weather')
```

```python
a = spark.sql("select * from weather")
```

```python
a.show(3)
```

```
[Stage 2:>                                                          (0 + 1) / 1]
+----------------+-------------+-----------------------+-----------------+---------+-----------------+----------------+-----------+-------------+----------------+
|           Fecha|Temperatura_c|Velocidad_del_viento_km_h|Direccion_del_viento|Racha_km_h|Direccion_de_racha|Precipitacion_mm|Presion_hPa|Tendencia_hPa|Humedad_percent|
+----------------+-------------+-----------------------+-----------------+---------+-----------------+----------------+-----------+-------------+----------------+
|02/04/2023 06:00|         17.4|                     17|              Sur|       30|          Sudeste|               0|      950.0|         -1.1|              40|
|02/04/2023 05:00|         17.4|                     17|              Sur|       30|          Sudeste|               0|      950.2|         -1.5|              36|
|02/04/2023 04:00|         18.1|                     18|          Sudeste|       26|              Sur|               0|      950.6|         -1.7|              33|
+----------------+-------------+-----------------------+-----------------+---------+-----------------+----------------+-----------+-------------+----------------+
only showing top 3 rows
```

Figura 4.10: Carga de la información en la base de datos Hive

4.2 Caso específico

Una vez que hemos hecho una prueba de concepto con cada una de las tecnologías, se procedió a unificar todas estas plataformas y a usarlas para la extracción, la transformación, almacenamiento, análisis de los datos y la realización de un modelo ARIMA. En este caso en concreto, se han utilizado los datos sobre los balances de la empresa Cajasiete [22] desde el año 2018 hasta el 2022, específicamente sobre los activos de cada entidad.

4.2.1 Extracción, transformación y almacenamiento de los datos

Como se comentó previamente, los datos que se utilizaron para realizar este caso en específico fueron los que proporciona La Unión Nacional de Cooperativas de Crédito (UNACC) [23]. Una vez que se tuvo localizada la fuente de origen se procedió a la realización de un DAG en Apache Airflow, que a continuación procederé a explicar.

Como bien sabemos, para la construcción de un DAG debemos hacerlo mediante el lenguaje Python. Para realizar el proceso de la extracción de los datos lo que se hizo fue establecer una función, tal y como se muestra en la figura 4.11, que se encargara de este paso.

```python
def fetch_information_2018_2019():

    #Here we get the balance sheets from the URL and then we turn into excel file.
    response = requests.get("https://www.unacc.com/wp-content/uploads/2020/03/Balances.xlsx", verify=False)
    excel_file = pd.ExcelFile(response.content)

    #here we get the sheet names from the excel file that contains the word 'Individual' and this  search start from
    #the sheet number 13.
    sheet_names = [sheet_name for sheet_name in excel_file.sheet_names[13:] if "Individual" in sheet_name]

    #this piece of code create a list of DataFrames, one for each sheet in the excel file that contains
    #the word 'Individual'
    dfs = [
    pd.read_excel(excel_file, sheet_name=sheet_name, skiprows=4)
    for sheet_name in sheet_names]

    return dfs
```

Figura 4.11: Función Python para la extracción de los balances del 2018 y 2019

Una vez que los datos ya los tenemos en un dataframe de la librería Pandas se realiza una limpieza de los datos, para ello, se creó otra función (figura 4.12) que se encargará de aspectos como la eliminación de valores nulos.

```python
def clean_dataframe(df: pd.DataFrame) -> pd.DataFrame:

    # Clean blanks
    df = df.dropna(how="all", axis="rows")
    df = df.dropna(how="all", axis="columns")
    df = df.rename(columns={df.columns[0]: "EPIGRAFE"})
    if 'Unnamed: 6' in df.columns:
        df = df.drop(columns=['Unnamed: 6'])

    df = df.dropna(subset=['EPIGRAFE'])

    # Clean date
    fecha_datos = df.iloc[0, -1]
    df = df.iloc[1:]
    df = df.reset_index(drop=True)

    # Orden epigrafe
    df["ORDEN_EPIGRAFE"] = df.index

    # Unpivot
    df = pd.melt(df, id_vars=["EPIGRAFE", "ORDEN_EPIGRAFE"], var_name="ENTIDAD", value_name="VALOR")
    df["FECHA"] = fecha_datos

    # Nan values
    df['VALOR'].fillna(0, inplace=True)

    return df
```

Figura 4.12: Función Python para la limpieza de los datos

El siguiente paso para realizar, es la conexión con nuestra base de datos creada por Apache Hive. Para ello, se utilizó la librería Pyhive [24] para establecer la conexión entre Apache Airflow y Apache Hive. Ya creada la conexión se procedió a la definición de dos funciones, una para la creación de las tablas (figura 4.13) y otra para la inserción de los datos (figura 4.14).

```python
def create_hive_table():
    conn = hive.Connection(host="host.docker.internal", port=10000, username="hive", database="Balancesdb")
    cursor = conn.cursor()

    # Creation of hive table for 2018 and 2019 data
    create_table_query_2018_2019 = """
       CREATE EXTERNAL TABLE IF NOT EXISTS balances1819 (
            epigrafe STRING,
            orden_epigrafe STRING,
            entidad STRING,
            valor STRING,
            fecha STRING,
            nivel_epigrafe STRING,
            cod_entidad STRING
            )
            ROW FORMAT DELIMITED
            FIELDS TERMINATED BY ','
            LINES TERMINATED BY '\n'
    """
```

Figura 4.13: Función Python para la conexión y creación de tablas en Hive

```python
def insert_data():
    conn = hive.Connection(host="host.docker.internal", port=10000, username="hive", database="Balancesdb")
    cursor = conn.cursor()

    balance_data_18_19 = cleaning_process(fetch_information_2018_2019())
    balances_data20 = cleaning_process(fetch_information_2020())
    balances_data21 = cleaning_process(fetch_information_2021())
    balances_data22 = cleaning_process(fetch_information_2022())

    insert_query = "INSERT INTO TABLE balances1819 VALUES (%s, %s, %s, %s, %s, %s, %s)"
    for row in balance_data_18_19.itertuples(index=False):
        cursor.execute(insert_query, (row[0], row[1], row[2], row[3], row[4], row[5], row[6]))
```

Figura 4.14: Función Python para la conexión e inserción de los datos en Hive

Finalmente, obtendríamos un DAG que nos ayuda a automatizar la extracción, transformación y almacenamiento de los datos (figura 4.15).

Figura 4.15: DAG del caso específico en Apache Airflow

Luego con la ayuda de Apache Hue podemos observar si se han cargado correctamente los datos, tal y como se muestra en la figura 4.16.

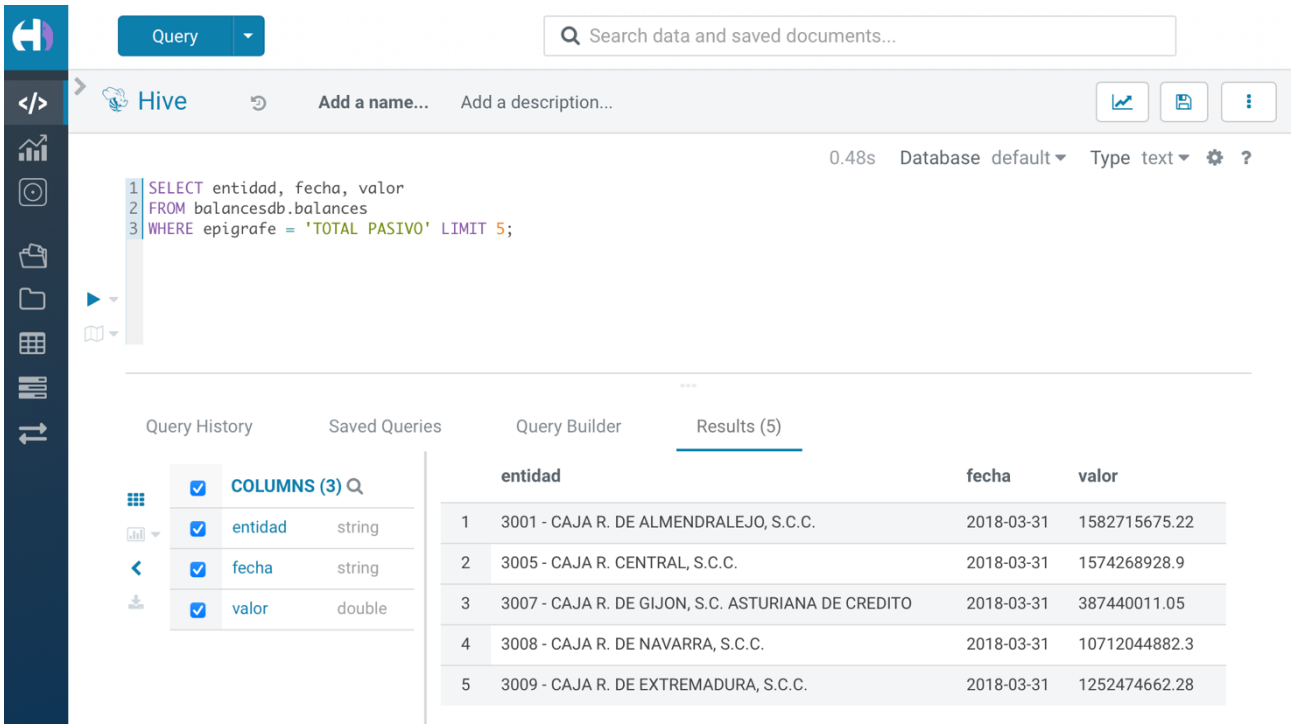

Figura 4.16: Interfaz de usuario web de Apache Hue

4.2.2 Análisis exploratorio de los datos

Como se comentó anteriormente, el EDA es un proceso que podemos utilizar como un primer acercamiento con los datos que tenemos. En definitiva, se pretende conocer la calidad de los datos que tenemos, así como de establecer si es viable o no trabajar con ellos.

En primer lugar, empezamos conociendo los metadatos de nuestro conjunto de datos, esto mediante la ayuda de la librería Pandas [25], en este caso vamos a ver el ejemplo con los datos del 2018 y 2019 de los balances de Cajasiete. El valor de los metadatos se puede ver en la figura 4.17.

```
Cantidad de Filas y columnas: (71920, 7)
Nombre columnas: Index(['EPIGRAFE', 'ORDEN_EPIGRAFE', 'ENTIDAD', 'VALOR', 'FECHA',
       'NIVEL_EPIGRAFE', 'COD_ENTIDAD'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
Int64Index: 71920 entries, 0 to 8844
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   EPIGRAFE        71920 non-null  object
 1   ORDEN_EPIGRAFE  71920 non-null  int64
 2   ENTIDAD         71920 non-null  object
 3   VALOR           71920 non-null  float64
 4   FECHA           71920 non-null  object
 5   NIVEL_EPIGRAFE  71920 non-null  int64
 6   COD_ENTIDAD     71920 non-null  object
dtypes: float64(1), int64(2), object(4)
memory usage: 4.4+ MB
```

Figura 4.17: Metadatos de los balances del 2018 y 2019

En segundo lugar, creamos un gráfico de barras para visualizar como se distribuyen los datos. En este caso (figura 4.18), utilicé el valor total pasivo de cada entidad.

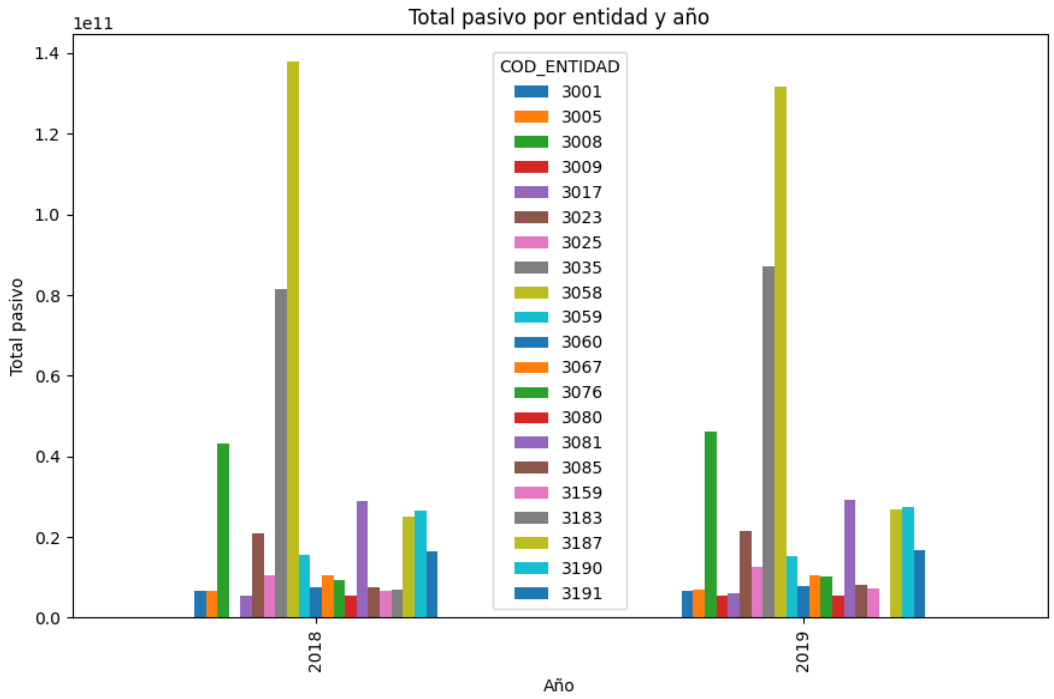

Figura 4.18: Gráfico de barras para el total pasivo por entidad

Otro aspecto muy importante a la hora de realizar un EDA es la implementación de un proceso de detección de outliers. Es por ello, que en este caso se optó por la utilización del método intercuartílico [26]. El IQR se define como la diferencia entre el tercer y primer cuartil de los datos. Para identificar a los outliers lo que hacemos es que cualquier valor que esté por encima de Q3 + 1.5 * IQR o por debajo de Q1 – 1.5 * IQR se considera outlier. El código se puede ver en la figura 4.19.

```
# Filtra los datos para incluir solo la columna "VALOR"
df_valores = df[["COD_ENTIDAD", "VALOR"]]

# Calcula los límites del boxplot
Q1 = df_valores["VALOR"].quantile(0.25)
Q3 = df_valores["VALOR"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Encuentra las entidades que son outliers
outliers = df_valores[(df_valores["VALOR"] < lower_bound) | (df_valores["VALOR"] >
upper_bound)]["COD_ENTIDAD"].unique()
```

Figura 4.19: Código Python para la aplicación del IQR

Como resultado del proceso comentado anteriormente, nos da el resultado ilustrado en la figura 4.20. Este resultado nos sirve para tener en cuenta estas entidades al momento de realizar un modelo de machine learning.

```
['3001' '3005' '3007' '3008' '3009' '3016' '3017' '3018' '3020' '3023'
 '3025' '3029' '3035' '3045' '3058' '3059' '3060' '3067' '3070' '3076'
 '3080' '3081' '3085' '3089' '3095' '3096' '3098' '3102' '3104' '3105'
 '3110' '3111' '3112' '3113' '3115' '3117' '3118' '3119' '3121' '3123'
 '3127' '3130' '3134' '3135' '3138' '3140' '3144' '3150' '3152' '3157'
 '3159' '3160' '3162' '3165' '3166' '3174' '3179' '3183' '3186' '3187'
 '3190' '3191' 'TOTAL']
```

Figura 4.20: Resultado tras aplicar el proceso IQR

4.2.3 Modelo ARIMA

Para la realización del modelo ARIMA se utilizaron los datos que hacen referencia a los activos de la entidad Cajasiete, caja rural, S.C.C (3076) y la herramienta JupyterLab.

Como primer paso para la realización de nuestro modelo ARIMA recuperamos los datos de nuestra base de datos en Apache Hive. Una vez que tenemos los datos que vamos a utilizar procedemos a verificar si nuestros datos son estacionarios, es decir, si los datos mantienen una distribución estadística constante a lo largo del tiempo, sin mostrar tendencias o cambios en la varianza. Para ver si nuestros datos son estacionarios o no, visualizamos nuestros datos tal y como se muestra en la figura 4.21.

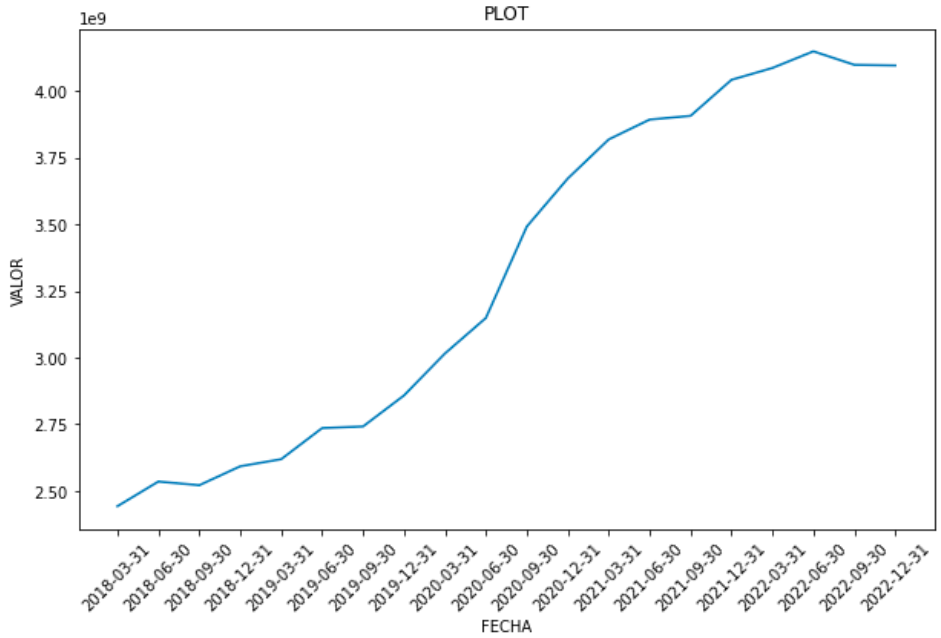

Figura 4.21: Gráfico de los datos por trimestre

Como se puede observar en la figura 4.21, estamos ante unos datos que no son estacionarios debido a que vemos patrones de crecimiento a lo largo del tiempo.

Otra manera de saber si nuestros datos son estacionarios es mediante la prueba de Dickey-Fuller [27], a grandes rasgos lo que analizamos en el p-valor de manera que si es menor que 0.05 se concluye que los datos son estacionarios de lo contrario no serían estacionarios. En nuestro caso este valor es de 0.9970, por lo que podemos afirmar que los datos no son estacionarios.

Para convertir nuestros datos en estacionarios, lo que tenemos que hacer es hallar las diferencias de los valores. En nuestro caso, para que los valores sean estacionarios se tuvo que hallar las segundas diferencias. En la figura 4.22, vemos como se distribuían los valores y en la figura 4.23 vemos como lo hacen con las segundas diferencias.

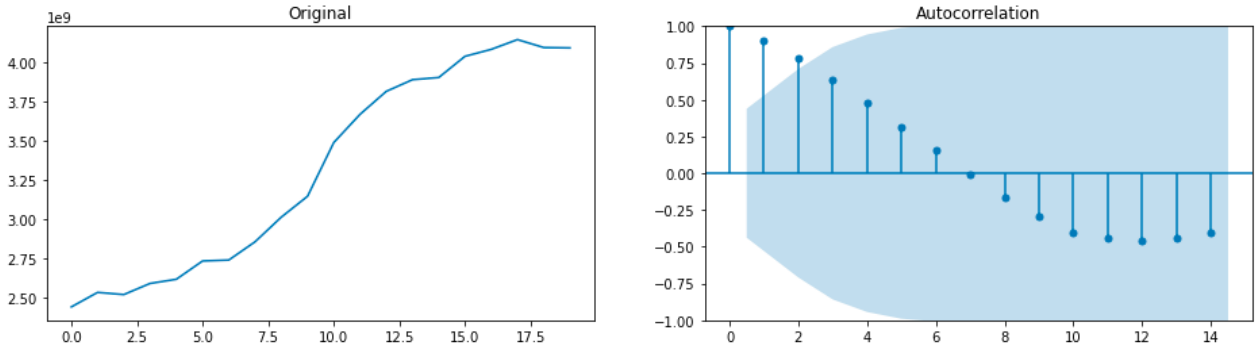

Figura 4.22: Valores sin aplicar ninguna diferencia

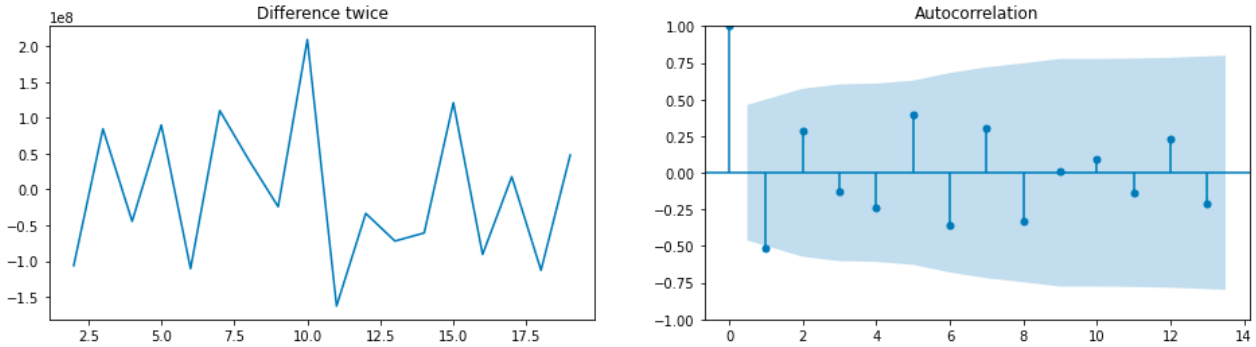

Figura 4.23: Valores con segunda diferencia

Una vez que tenemos nuestros valores estacionarios, procedemos a hallar los parámetros P y Q del modelo ARIMA. En este modelo, elegí un valor 4 para P y un valor 2 para Q.

Finalmente, al tener los parámetros de nuestro modelo ARIMA, procedemos a ver como son las predicciones (figura 4.24).

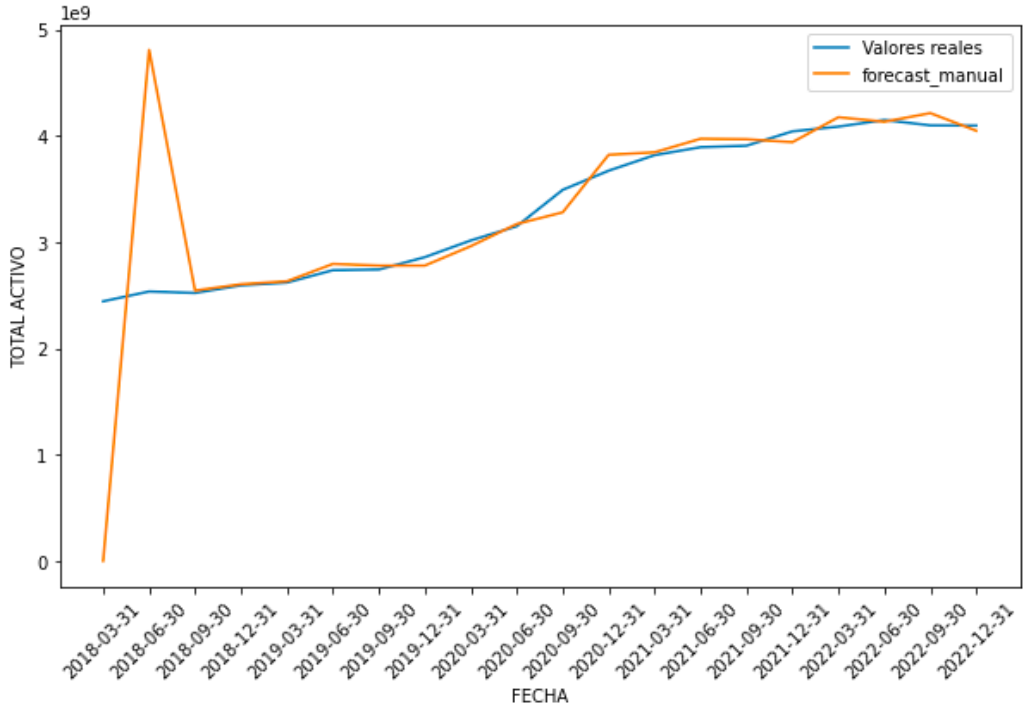

Figura 4.24: Gráfico de los valores reales y las predicciones con el modelo ARIMA manual

Como podemos ver en la figura 4.24, los valores predichos se tratan de ajustar a los valores reales exceptuando los primeros.

Cabe destacar que el modelo ARIMA se puede establecer mediante la librería pmdarima [28] de forma automática. En este caso generó un modelo ARIMA (0,2,0), diferente al que establecimos manualmente ARIMA (4,2,1). La comparación de estos dos modelos lo podemos ver en la figura 4.25.

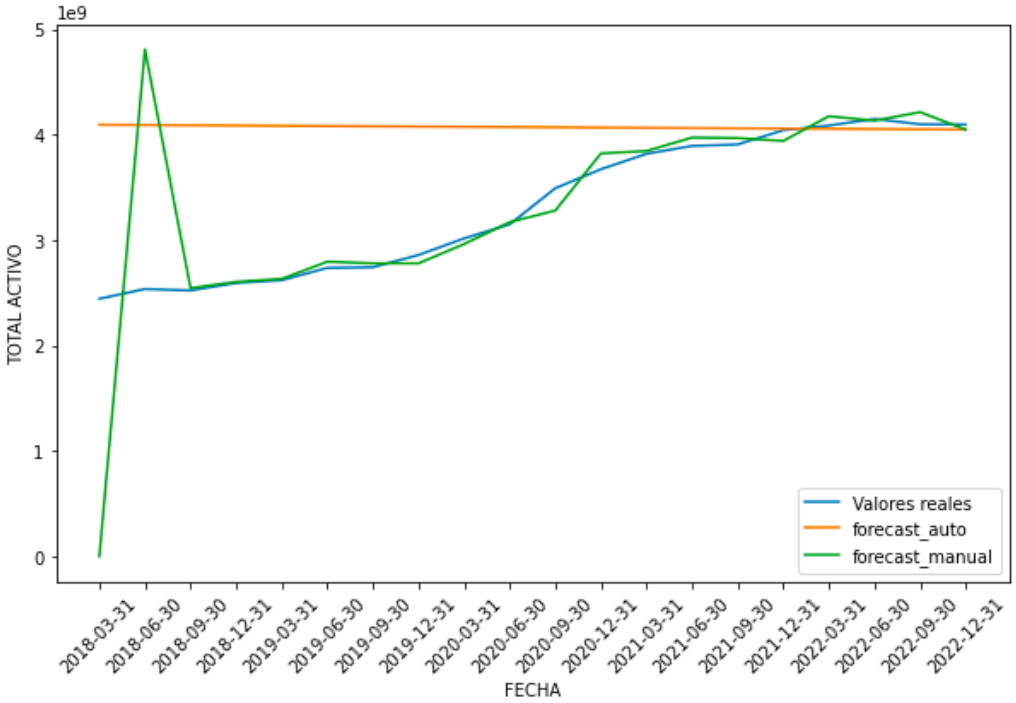

Figura 4.25: Gráfico con los valores reales, modelo ARIMA manual y automático

Una vez que tenemos claro nuestro modelo, procedemos a dividir nuestro conjunto de datos en dos. Una parte para entrenamiento y otra para prueba. De esta manera obtenemos los siguientes resultados ilustrados en la figura 4.26.

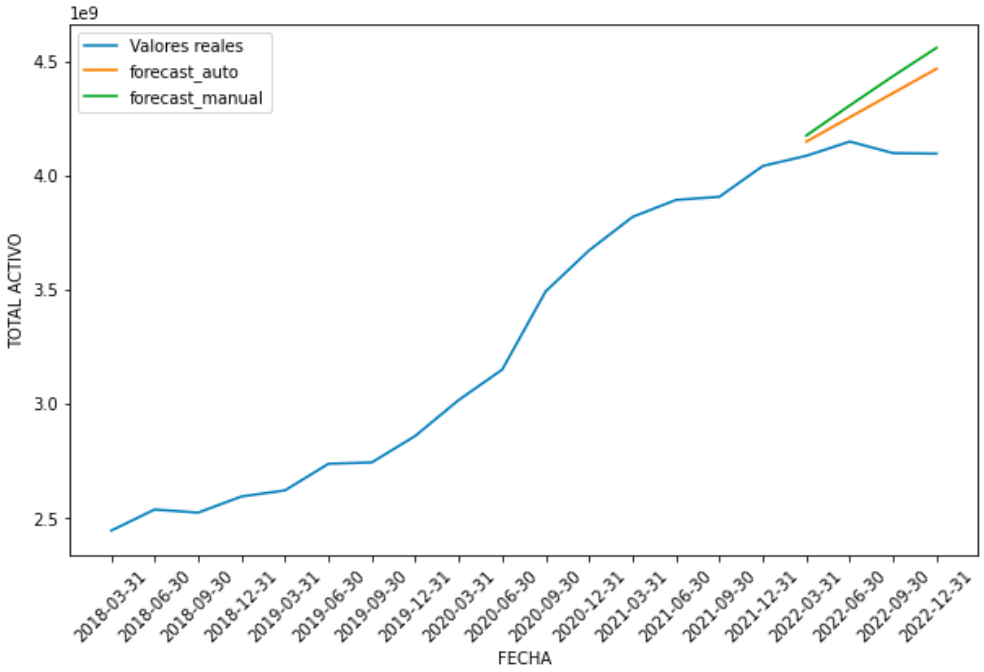

Figura 4.26: Gráfica con los valores reales y las predicciones mediante el uso de conjuntos

Posteriormente, realizamos unas predicciones fuera del rango de tiempo, el cual se muestra en la figura 4.27

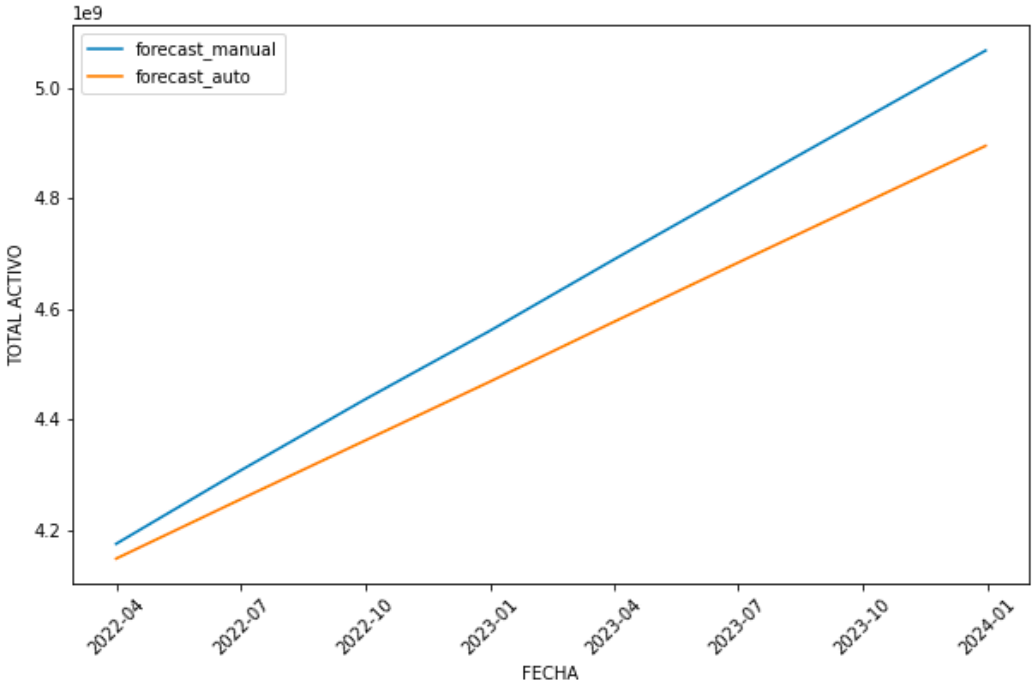

Figura 4.27: Gráfico con las predicciones para años posteriores

Para evaluar la eficacia del modelo debemos analizar el MAE, MAPE y RMSE. Los valores se pueden observar en la tabla 4.1 para cada modelo.

Modelos	MAE	MAPE	RMSE
Manual ARIMA (4,2,1)	262440882.86322594	0.06395386062312519	301236501.3777132
Auto ARIMA (0,2,0)	201214483.2933334	0.0490443272908284	236396853.65874314

Tabla 4.1: Valores de la eficiencia de los modelos

4.2.4 Visualización de los datos

Para la visualización de los datos se optó por el uso de la herramienta Power BI [29], debido a que esta herramienta nos permite crear reportes de manera creativa e interactiva.

El reporte creado en Power BI se divide en dos páginas, la primera (figura 4.28), muestra los valores generales de todas las entidades desde el año 2018 hasta el 2023.

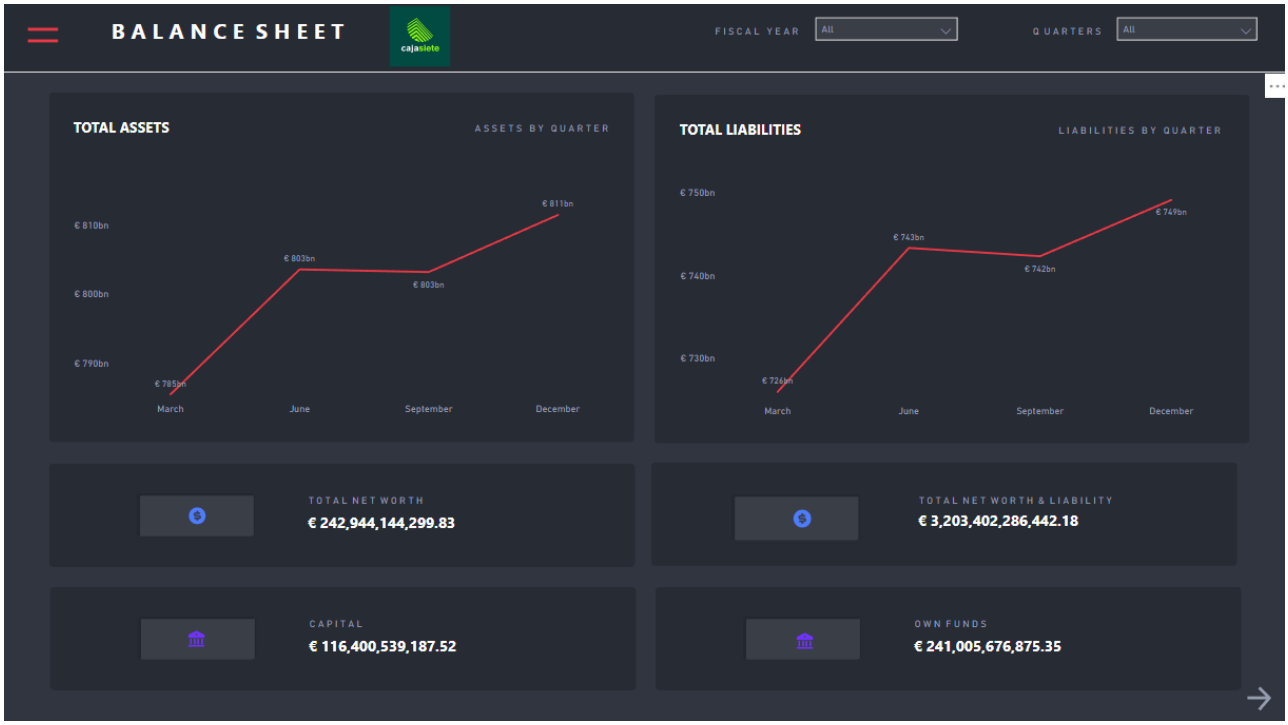

Figura 4.28: Página principal del Power BI que muestra información general

La segunda página (figura 4.29), muestra el top 3 de entidades según sus activos, así como una matriz con todos los valores desglosados.

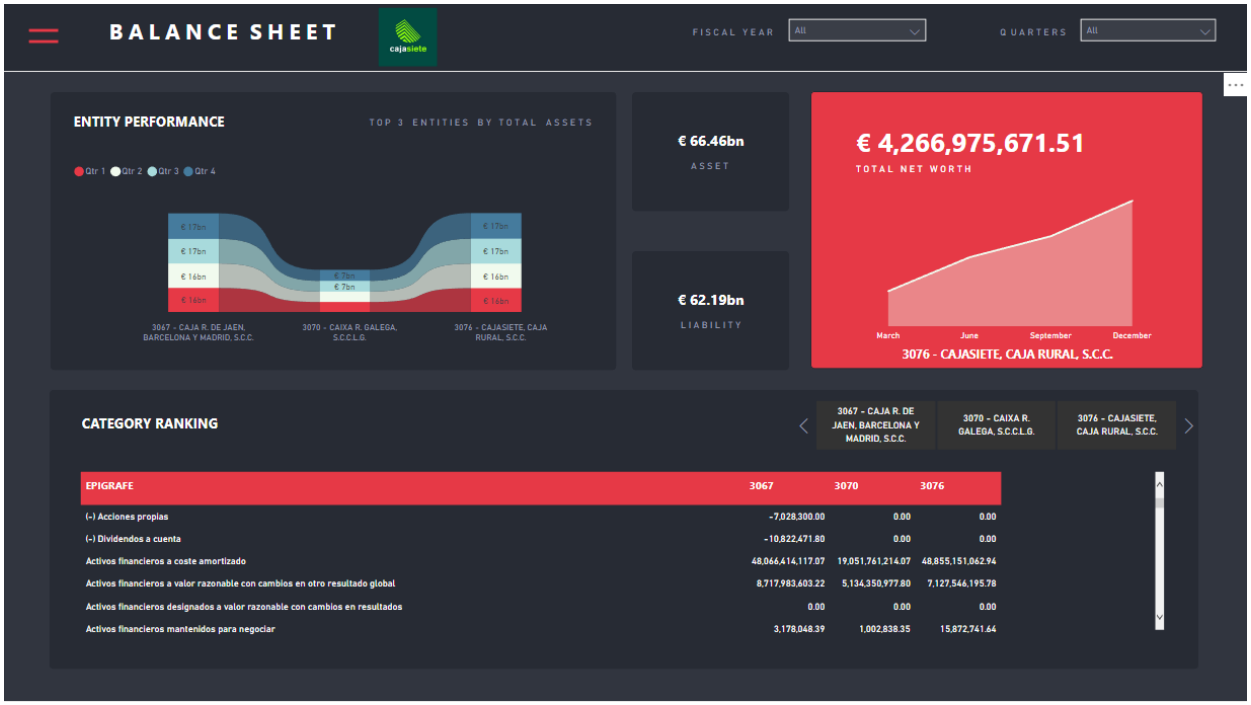

Figura 4.29: Página del Power BI que muestra el ranking de las entidades

Como podemos ver Power BI nos facilita la visualización de los datos, haciendo que las personas que hagan uso de este "dashboard" puedan enfocarse en los datos que verdaderamente importan.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

Establecer una plataforma de datos donde se puedan efectuar procesos de Big Data no resulta una tarea fácil por el trabajo de configuración y la curva de aprendizaje de estas herramientas.

A lo largo del presente trabajo, se ha podido aprender una gran variedad de herramientas que están bajo un marco de trabajo de código abierto. Esto es una gran ventaja debido a que la unión de estas tecnologías se puede llevar a un entorno de producción para dar soluciones al tratamiento del Big Data sin que haya un factor monetario importante.

Con este trabajo no solo se busca evaluar diferentes plataformas de datos para el tratamiento de la información, sino que también se busca promover la curiosidad en aquellas personas interesadas en el mundo de los datos para que puedan crear sus propios entornos de trabajo. Por otro lado, también se pretende dar a conocer la importancia del machine learning como proceso para dar valor añadido a los datos, aunque no sea objetivo de este trabajo.

Otro aspecto importante y que cada día toma más relevancia es la parte visual de los datos, debido a que se busca establecer una parte creativa e interactiva de los datos que ayuden a los interesados a conocer su negocio y tomar decisiones sobre los datos que están observando.

5.2 Líneas futuras

Hay una infinidad de caminos por los cuales se puede seguir este trabajo. Algunas líneas son:

1. Utilizar el entorno creado en este proyecto para establecer una gran carga de datos con los cuales se pueda establecer un modelo más complejo de machine learning, y si cabe la posibilidad de añadir nuevas tecnologías.
2. Combinar herramientas del Big Data presentes en entornos privados, tales como, Snowflake, Microsoft Azure o incluso AWS. Esto se haría con el objetivo de potenciar la plataforma de datos actual del proyecto.

Capítulo 6
Summary and Conclusions

6.1 Conclusions

Establishing a data platform where Big Data processes can be carried out is not an easy task due to the configuration work and the learning curve of these tools.

Throughout this work, it has been possible to learn a wide variety of tools that are under an open-source framework. This is a great advantage because the union of these technologies can be taken to a production environment to provide solutions for Big Data treatment without there being a significant monetary factor.

This work not only seeks to evaluate different data platforms for information processing, but also seeks to promote curiosity in those people interested in the world of data so that they can create their own work environments. On the other hand, it is also intended to publicize the importance of machine learning as a process to add value to the data, although it is not the objective of this work.

Another important aspect that is becoming more relevant every day is the visual part of the data, because it seeks to establish a creative and interactive part of the data that helps interested parties to know their business and make decisions about the data they are observing.

6.2 Future work

There are an infinity of paths by which this work can be followed. Some lines are:

1. Use the environment created in this project to establish a large data load with which a more complex machine learning model can be established, and if possible, add new technologies.
2. Combine Big Data tools present in private environments, such as Snowflake, Microsoft Azure or even AWS. This would be done with the aim of strengthening the project's current data platform.

Capítulo 7
Presupuesto

Este capítulo se muestran los costes estimados del proyecto.

7.1 Costes de hardware

Tipos	Descripción	Precio
MacBook Pro	Ordenador donde se realizó todo el proyecto	1200€

Tabla 7.1: Costes de hardware

7.2 Costes de recursos humanos

Horas de trabajo	Coste por hora	Total
600	18,84€	11304€

Tabla 7.2: Costes de recursos humanos

7.3 Costes totales

Hardware	Recursos Humanos	Total
1200€	11304€	12504€

Tabla 7.3: Costes totales

Capítulo 8
Apéndice A

8.1 Repositorio GitHub

En el siguiente enlace se encuentra el repositorio que contiene el trabajo que se ha realizado a lo largo del proyecto.

https://github.com/alu0101330457/Big-Data.git

Bibliografía

[1] "Big Data: Qué es y por qué importa". https://www.sas.com/es_es/insights/big-data/what-is-big-data.html(accedido el 11 de mayo de 2023).

[2] "¿Qué es Open Data? |". Inicio - Universidad de La Laguna. https://www.ull.es/catedras/catedrabob/que-es-open-data/(accedido el 11 de mayo de 2023).

[3] C. Domínguez García, "Predicción de los niveles de polución atmosférica en Tenerife mediante técnicas de Machine Learning", trabajo de grado, Universidad de la Laguna, Santa Cruz de Tenerife, 2019. dirección: http://riull.ull.es/xmlui/handle/915/16555.

[4] Helena. "Big data analytics ¿Qué es y por qué es tan importante? | AyudaLey Datos". Ayuda Ley Protección Datos. https://ayudaleyprotecciondatos.es/big-data/analytics/(accedido el 21 de mayo de 2023).

[5] "¿Qué es Open Data? |". Inicio - Universidad de La Laguna. https://www.ull.es/catedras/catedrabob/que-es-open-data/(accedido el 21 de mayo de 2023).

[6] "Análisis Exploratorio de Datos con Pandas en Python". Aprende Machine Learning. https://www.aprendemachinelearning.com/analisis-exploratorio-de-datos-pandas-python/(accedido el 21 de mayo de 2023).

[7] "Detección de outliers en Python". Aprende Machine Learning. https://www.aprendemachinelearning.com/deteccion-de-outliers-en-python-anomalia/(accedido el 21 de mayo de 2023).

[8] "What is a Data Warehouse? | IBM". IBM - Deutschland | IBM. https://www.ibm.com/topics/data-warehouse(accedido el 21 de mayo de 2023).

[9] "What is a Workflow? | IBM". IBM - Deutschland | IBM. https://www.ibm.com/topics/workflow(accedido el 21 de mayo de 2023).

[10] "Guía de Aprendizaje". Aprende Machine Learning. https://www.aprendemachinelearning.com/guia-de-aprendizaje/(accedido el 21 de mayo de 2023).

[11] "8 Series Temporales | Estadística y Machine Learning con R". Home | Bookdown. https://bookdown.org/content/2274/series-temporales.html(accedido el 21 de mayo de 2023).

[12] "Introducción a las series temporales (2): modelo ARIMA con R". The Machine Learners. https://www.themachinelearners.com/series-temporales-arima/(accedido el 21 de mayo de 2023).

[13] "Contenedores de Docker | ¿Qué es Docker? | AWS". Amazon Web Services, Inc. https://aws.amazon.com/es/docker/(accedido el 21 de mayo de 2023).

[14] "Apache Hadoop: What is it and how can you use it?" Databricks. https://www.databricks.com/glossary/hadoop(accedido el 21 de mayo de 2023).

[15] "Why developers like Hadoop". StackShare. https://stackshare.io/hadoop(accedido el 21 de mayo de 2023).

[16] "Apache Hive". Apache Hive. https://hive.apache.org(accedido el 21 de mayo de 2023).

[17] "Apache Spark™ - Unified Engine for large-scale data analytics". Apache Spark™ - Unified Engine for large-scale data analytics. https://spark.apache.org(accedido el 21 de mayo de 2023).

[18] "Hue - The open source SQL Assistant for Data Warehouses". Hue - The open source SQL Assistant for Data Warehouses. https://gethue.com(accedido el 21 de mayo de 2023).

[19] "Home". Apache Airflow. https://airflow.apache.org(accedido el 21 de mayo de 2023).

[20] "Project Jupyter". Project Jupyter | Home. https://jupyter.org(accedido el 21 de mayo de 2023).

[21] "GitHub - big-data-europe/docker-hadoop: Apache Hadoop docker image". GitHub. https://github.com/big-data-europe/docker-hadoop.git(accedido el 21 de mayo de 2023).

[22] "Comprometidos | Cajasiete". Comprometidos | Cajasiete. https://www.cajasiete.com/es(accedido el 21 de mayo de 2023).

[23] "2022 - UNACC". UNACC. https://www.unacc.com/estados-financieros/2022-2/(accedido el 21 de mayo de 2023).

[24] "PyHive". PyPI. https://pypi.org/project/PyHive/(accedido el 21 de mayo de 2023).

[25] "pandas - Python Data Analysis Library". pandas - Python Data Analysis Library. https://pandas.pydata.org(accedido el 21 de mayo de 2023).

[26] "IQR (Rango intercuartílico)". Moved. https://docs.oracle.com/cloud/help/es/pbcs_common/PFUSU/insights_metrics_IQR.htm#PFUSU-GUID-CF37CAEA-730B-4346-801E-64612719FF6B(accedido el 21 de mayo de 2023).

[27] "Contraste de Dickey-Fuller | Economipedia". Economipedia. https://economipedia.com/definiciones/contraste-de-dickey-fuller.html#:~:text=El%20contraste%20de%20Dickey-Fuller,mediante%20un%20contraste%20de%20hipótesis.(accedido el 21 de mayo de 2023).

[28] "pmdarima". PyPI. https://pypi.org/project/pmdarima/(accedido el 21 de mayo de 2023).

[29] "Data Visualisation | Microsoft Power BI". Data Visualization | Microsoft Power BI. https://powerbi.microsoft.com/en-au/(accedido el 21 de mayo de 2023).