Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

IMPLEMENTACIÓN DE MODELOS DE ENSEMBLES PARA BIG DATA

Alumno: Alberto Moreno Molina

Tutores: Prof. Dña. María Dolores Pérez Godoy
Prof. D. Antonio Jesús Rivera Rivas

Dpto: Informática

Junio, 2020

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rivera Rivas y Doña María Dolores Pérez Godoy, tutores del Proyecto Fin de Carrera titulado: Implementación de modelos de ensembles para Big Data, que presenta Alberto Moreno Molina, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2020

El alumno:

MORENO MOLINA ALBERTO - 77386885G

Firmado digitalmente por MORENO MOLINA ALBERTO - 77386885G
Fecha: 2020.06.19 20:06:05 +02'00'

Fdo.: Alberto Moreno Molina

Los tutores:

RIVERA RIVAS ANTONIO JESUS - 26015863B

Firmado digitalmente por RIVERA RIVAS ANTONIO JESUS - 26015863B
Nombre de reconocimiento (DN): c=ES, serialNumber=26015863B, sn=RIVERA RIVAS, givenName=ANTONIO JESUS, cn=RIVERA RIVAS ANTONIO JESUS - 26015863B
Fecha: 2020.06.22 12:02:43 +02'00'

Fdo.: Antonio Jesús Rivera Rivas

PEREZ GODOY MARIA DOLORES - 75093149N

Firmado digitalmente por PEREZ GODOY MARIA DOLORES - 75093149N
Fecha: 2020.06.22 13:55:12 +02'00'

Fdo.: María Dolores Pérez Godoy

AGRADECIMIENTOS

A mis tutores, Antonio y Loli, por guiarme y aconsejarme durante la realización de este Trabajo de Fin de Grado, además de una gran ayuda proporcionada en cuanto a temario e ideas, y también por adaptarse a las diversas circunstancias y las posibles complicaciones que han podido surgir durante este tiempo.

A mi pareja, Carmen, por apoyarme incondicionalmente, motivarme para lograr mis metas, hacerme reir en los peores días y soportarme. Por animarme y alegrarme cada día, y ser la persona que mejor me entiende. Por compartir tanto los buenos como los malos momentos, pero siempre codo con codo

A mis padres y hermana, Maria Dolores, Pedro e Irene, por apoyarme durante este y todos los proyectos de mi vida y animarme a conseguir aquello que quiero, por compartir mis éxitos y alegrías, y ayudarme a superar aquellos fracasos y malos momentos.

A mi grupo de amigos, 'La Mafia', por todos las buenas experiencias vividas juntos, ánimos y ayuda obtenida por vuestra parte en todos los ámbitos posibles, y por las buenas relaciones que hemos podido conseguir a lo largo de nuestra vida

A mis compañeros de Universidad más cercanos, por tantas clases juntos, las vueltas entre clase y clase alrededor de la Universidad para despejarnos y charlar, y por esos quebraderos de cabeza con ciertas prácticas y exámenes. A pesar de todo el trabajo duro, hemos conseguido finalizar el grado y obtener nuestro título, y conocer a personas maravillosas.

ÍNDICE

1. INTRODUCCIÓN, MOTIVACIÓN Y OBJETIVOS DEL PROYECTO

En la actualidad existe una gran demanda de extracción de conocimiento a partir de colecciones de datos. Bajo el término de **Ciencia de Datos (Data Science)** aparecen un conjunto de disciplinas que permiten el análisis de datos y la obtención de conocimiento. Cuando los datos de los que se dispone son de gran volumen, diversidad de formato y generados a gran velocidad **(Big Data)**, los métodos clásicos existentes de extracción de conocimiento no pueden ser directamente aplicados sobre ellos, por lo que se necesita desarrollar nuevos paradigmas.

Una de las disciplinas más importantes dentro de la Ciencia de Datos y la encargada de la obtención de conocimiento es la **minería de datos (data mining)**. Técnicas importantes, que está obteniendo buenos resultados dentro de la minería de datos, son los **multiclasificadores y ensembles**, o combinaciones de modelos que mejoran el rendimiento de los modelos básicos que lo forman.

Con la realización de este trabajo se pretende avanzar hacia un campo novedoso y con gran proyección de futuro, en el que se desarrollan modelos de Data Science, en particular al desarrollo de modelos ensembles que sean capaces de afrontar la problemática asociada a Big Data.

Este proyecto de fin de grado consiste en un trabajo teórico-experimental, cuyo objetivo es el desarrollo de **técnicas de ensemble**, para ciencias de datos y diferentes tipos de conjuntos de datos, que utilizan como clasificadores base algoritmos de la biblioteca de **Machine Learning (ML de Apache Spark)** aplicadas a las tareas de clasificación y regresión.

Se han usado algunas de las tecnologías y lenguajes de programación más novedosos en el contexto de Big Data y Data Science, que permiten un procesamiento distribuido de los datos, como son **Apache Spark, Scala y ML**.

Al tratar un lenguaje de programación totalmente diferente al que he estado acostumbrado a lo largo de la carrera, ha sido un reto, pero a su vez supone un avance en mis conocimientos y un camino de aprendizaje.

1.1. Introducción

Vivimos en una sociedad en la que se dispone de un número, cada vez más amplio, de dispositivos tales como: ordenadores, teléfonos móviles, dispositivos inteligentes, etc. Conectados a la red, y generando grandes cantidades de información, como por ejemplo la cantidad de pasos realizados en un día, ubicaciones en las que hemos estado, ritmo cardíaco, etc.

Cuando hablamos de **Big Data**, nos referimos a grandes cantidades de datos entre las que se encuentran conjuntos de datos estructurados, semiestructurados y no estructurados, cuyo tamaño (volumen), complejidad (variabilidad) y velocidad de crecimiento (velocidad) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas tradicionales.

Estas inmensas cantidades de datos no se pueden procesar con herramientas tradicionales, puesto que son poco eficientes y algo escasas, y además su coste computacional es elevado.

Big Data ofrece herramientas que almacenan y ordenan los datos, facilitan su comprensión y los hacen más accesibles y útiles.

En (Gartner, 2001) se definió el Big Data en base a tres factores: **Velocidad, Volumen y Variedad**, conocidas como las 3Vs.

- **Velocidad**: Velocidad a la que llegan los datos, ya que el Big Data los recibe y analiza en tiempo real.
- **Volumen**: Las bases de datos no dejan de crecer, por lo que alcanzar volúmenes inimaginables.
- **Variedad**: Los datos provienen de diversas fuentes y en diferentes formatos, y pueden ser tanto estructurados como no estructurados.

Las tecnologías de Big Data actuales permiten un menor coste de desarrollo, almacenamiento y procesamiento, así como una mayor facilidad y velocidad de ingesta de todo tipo de datos.

1.2. Motivación

Un nuevo estudio de Seagata, líder global en administración y almacenamiento de datos, previene un incremento de 10 veces el volumen actual de los datos mundiales para el año 2025. Este estudio, titulado Data Age 2025, realizado por IDC y patrocinado por Seagata, pronostica que la generación de datos para tal año ascenderá a un total de 163 zettabytes (IDC, 2017).

Esto supondrá nuevos desafíos para la recopilación, utilización y ubicación de la información, así como un procesamiento posterior que nos permita aprender de dichos datos, y que los transforme en conocimiento.

La necesidad de la Ciencia de Datos sobre estas grandes cantidades de datos es importante, ya que esos datos suelen ser sus fuentes de información para el análisis. La Ciencia de Datos logra analizar los grandes volúmenes de datos desordenados e incompletos, para poder llegar a conseguir grandes descubrimientos que impulsan decisiones como por ejemplo sobre operaciones y productos.

La necesidad de extraer conocimiento de grandes volúmenes de datos es sumamente importante, resulta muy útil para muchas empresas ya que proporciona respuestas a diversas preguntas. El conocimiento extraído permite identificar posibles problemas de una manera mucho más comprensible. Pero no solo sirve para identificar errores, también puede usarse de cara al futuro, de forma que se pueda aprovechar para identificar nuevas oportunidades.

A día de hoy, conceptos como minería de datos o Data Science los separa una línea muy delgada, la mayoría de esos conceptos conducen a la extracción de conocimiento de la información mediante técnicas estadísticas y/o Machine Learning.

Una buena forma de conseguir una extracción de conocimiento eficiente es a través del buen funcionamiento de las diversas técnicas de ensemble, ya que, a partir del conocimiento obtenido individualmente a partir de diferentes métodos, por ejemplo, usando métodos de MLlib, de tal forma que se obtenga un conocimiento más global e incluso mejorado respecto al obtenido por los clasificadores base.

La ventaja que obtenemos al combinar todos los métodos empleados individualmente, es que al poder ser diferentes, como cada método funciona de forma totalmente distinta, sus errores tienden a compensarse, consiguiendo así un mejor error de generalización.

El problema radica en que a día de hoy MLlib no dispone de esas técnicas de ensemble que facilitaría mucho la extracción de conocimiento de esos grandes volúmenes de datos.

1.3. Objetivos del proyecto

El **objetivo general del proyecto** es el desarrollo, implementación y evaluación de técnicas de ensemble para Big Data.

Los **objetivos específicos del proyecto** son los siguientes:

- Estudiar el problema de Big Data y las técnicas de Data Science para solucionarlo.
- Conocer técnicas de ensembles de minería de datos clásica y aplicadas a Big Data.
- Comprender las herramientas disponibles que permiten abordar la escalabilidad en Big Data, que permitan implementar el paradigma MapReduce.
- Instalación y configuración de los sistemas necesarios para implementar y ejecutar estos modelos.
- Familiarización con un lenguaje que permita la implementación de los modelos.

- Conocer las bibliotecas existentes de Machine Learning para Ciencia de Datos.
- Implementación de modelos de ensemble para aprendizaje automático que sigan los paradigmas antes citados.
- Análisis y evaluación de resultados.

1.4. Metodología desarrollada

La metodología seguida en el desarrollo del trabajo ha sido la siguiente:

- Revisión bibliográfica de todos los aspectos relacionados con el TFG.
- Elección de herramientas e instalación y configuración de las mismas.
- Implementación de los métodos de Data Science.
- Diseño de la experimentación.
- Experimentación sobre conjuntos de datos seleccionados.
- Análisis de los resultados obtenidos.
- Conclusiones.

1.5. Planificación del trabajo

El trabajo asociado a este T.F.G se ha distribuido en el periodo que transcurre entre febrero y finales de junio de 2020.

Para una buena planificación y correcto desarrollo del proyecto, se han ido contemplando diversos aspectos que se han ido realizando poco a poco para así garantizar una máxima eficiencia del trabajo elaborado, por ello las tareas se han ido dividiendo, de modo que hasta que no se terminaba una tarea no se comenzaba otra, y con ello nos asegurábamos de que todo se realizaba de la mejor manera posible y yendo paso por paso.

Nombre actividad	Fecha inicio	Duración en días	Fecha fin
Estudio sobre Big Data, Data Science y Minería de datos	01-feb	9	10-feb
Introducción con Scala	07-feb	7	14-feb
Pruebas con lenguaje programación Scala	12-feb	7	19-feb
Estudio sobre Spark y biblioteca SparkML	18-feb	15	04-mar
Instalación y configuración herramientas y bibliotecas	18-feb	9	27-feb
Tratamiento de datasets con Scala	04-mar	14	18-mar
Prueba funcionamiento métodos SparkML	18-mar	6	24-mar
Estudio teoría de ensembles	18-mar	7	25-mar
Desarrollo ensemble Bagging	24-mar	50	13-may
Configuración ficheros parámetros	24-mar	3	27-mar
Configuración y ejecucion en servidor	13-may	2	15-may
Pruebas en clúster con diferentes datasets	15-may	3	18-may
Corrección errores y modificación en código	18-may	23	10-jun
Pruebas reales de experimentos	10-jun	4	14-jun
Subir proyecto a spark-packages.org	12-jun	1	13-jun
Documentación codigo	10-jun	2	12-jun
Evaluación y análisis resultados de los experimentos	14-jun	4	18-jun
Documentación memoria	01-jun	19	20-jun
Resumen del proyecto	01-feb	140	20-jun

Tabla 1.1 Tareas del proyecto. Fuente: Propia

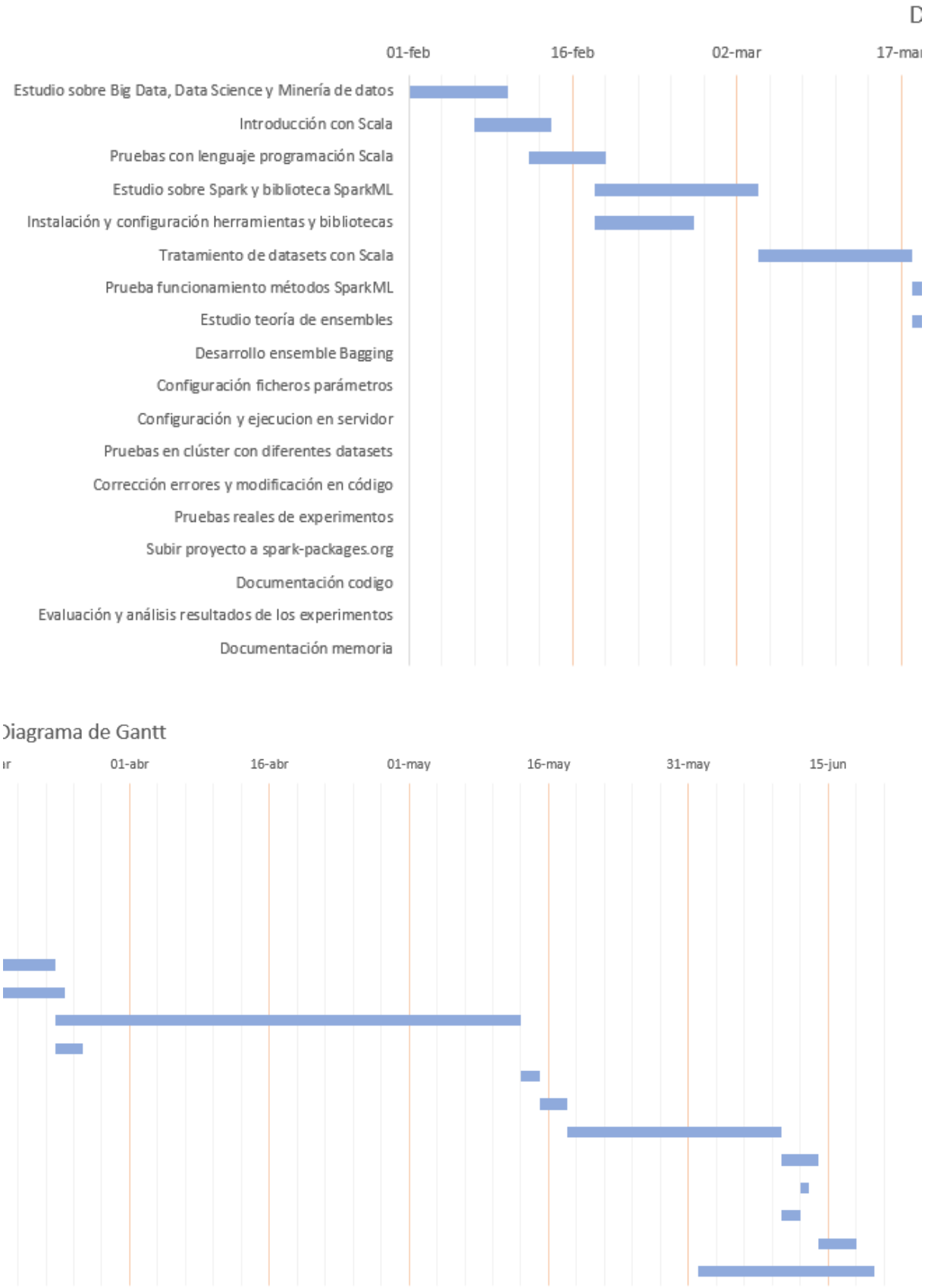


Figura 1.2 Diagrama de Gantt. Fuente: Propia

1.6. Estructura de la memoria

Este documento se estructura en tres grandes bloques, que son los siguientes:

El primer bloque denominado **Data Science**, el cuál consta de dos partes bien diferenciadas, una primera parte que nos introduce a dicho concepto y su relación con Big Data, y una segunda parte enfocada a la minería de datos donde se introduce dicho término, y seguidamente se habla del proceso KDD, tareas de la minería de datos y la clasificación, así como técnicas de evaluación del modelo de Machine Learning como es en nuestro caso la validación cruzada.

El segundo bloque trata sobre los **paradigmas de computación de Big Data**, más concretamente realizando un estudio teórico de Apache Hadoop y Apache Spark, comparación entre ambos y sus bibliotecas, así como una conclusión de cuál es la mejor solución entre ambos.

El tercer bloque trata la **teoría de ensembles**, así como una introducción a ellos, seguido de técnicas simples, errores que se pueden dar, y técnicas avanzadas. Haciendo hincapié en el **ensemble Bagging** que justo está en el apartado posterior, y en el cual nos hemos basado para el desarrollo del trabajo de fin de grado. Se expone tanto factores a tener en cuenta a la hora de desarrollar el ensemble, como un esquema explicativo de cómo funciona.

Por último, se tratará el **diseño e implementación del proyecto**, seguidamente de un **análisis de los diversos experimentos** realizados con los correspondientes resultados obtenidos y se realizarán una serie de **conclusiones** al respecto, junto con una zona explicativa de requisitos a tener en cuenta que han sido necesarios para poder conseguir el objetivo del proyecto, así también como un manual de usuario.

2. DATA SCIENCE

En este apartado, conseguiremos tener un mejor conocimiento sobre el concepto de Data Science, así como, la historia de esta área de conocimiento.

2.1 Historia sobre Data Science

En 2001, William S. Cleveland introdujo a la Ciencia de Datos como una disciplina independiente, extendiendo el campo de la estadística para incluir los avances en computación con datos *'Data science: and action plan for expanding the technical áreas of the field of statistics'* (Cleveland, 2001).

En abril del 2002, el *'Committee on Data for Science and Technology'* empezó la publicación del Data Science Journal, una revista digital dedicada al avance de la Ciencia de Datos, sus políticas de aplicación, sus prácticas y el manejo de datos de libre acceso.

Un año más tarde se comenzó a publicar The Journal of Data Science, donde los profesionales de datos podrían presentar sus perspectivas e intercambiar ideas.

En 2005, The National Science Board definió a los científicos de datos como *'científicos de computación e información, programadores de bases de datos y software, y expertos disciplinarios. Que son cruciales para la gestión exitosa de una colección digital de datos, cuya actividad primaria es realizar investigación creativa y análisis'*.

A partir de 2010, esta área comenzó a tomar una especial relevancia hasta llegar al auge que experimenta en la actualidad.

2.2 ¿Qué es Data Science?

Data Science es la ciencia que se encarga de extraer información o conocimiento de grandes cantidades de datos. Dicha ciencia combina la estadística, matemáticas e informática para interpretar los datos.

El principal objetivo de la Ciencia de Datos es la obtención de un conocimiento que pueda ser de utilidad para los expertos del área del problema.

Esta ciencia incluye también el Big Data, lo que significa, que en la Ciencia de Datos se aplican tanto técnicas de minería de datos como de otras áreas sobre grandes colecciones de datos con el fin de conseguir el mayor conocimiento usando herramientas Big Data (Alcalá-Fdez, y otros, 2011).

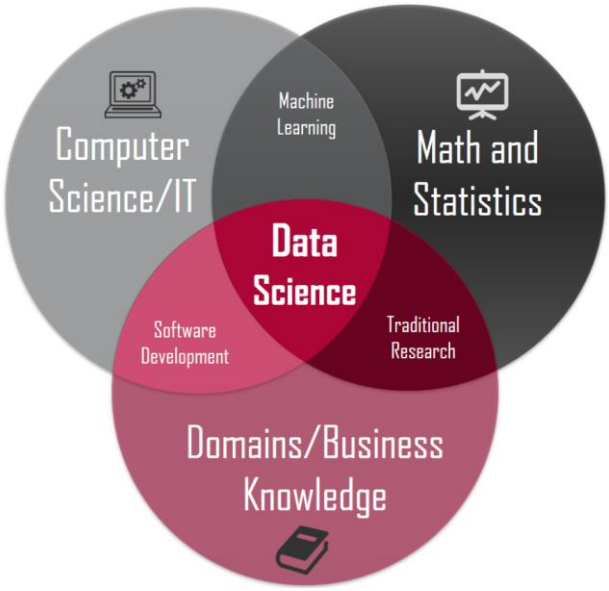


Figura 2.1 Disciplinas del Data Science. Fuente: (Edward, 2019)

Con la figura 2.1, podemos ver como todo está relacionado entre sí y partiendo de una serie de disciplinas, podemos relacionar sus conceptos principales para dar lugar a unas nuevas ideas que poder explorar.

2.3 Data Science y Big Data

Con Big Data nos referimos a enormes cantidades de datos que no se pueden procesar de forma efectiva con aplicaciones tradicionales (Monnapa, 2020). Debido a que contiene un tamaño considerable de datos, se encuentra con dificultades a la hora de almacenarse en bases de datos tradicionales, y por consiguiente procesarse en servidores estándar y analizarse con aplicaciones de uso diario.

Big Data se suele relacionar con Data Science, porque ésta última mencionada suele ser su fuente de información para el análisis, ya que analiza las grandes colecciones de datos desordenados e incompletos, para llegar a puntos que nos llevan a decisiones sobre operaciones y productos.

2.4 Fundamentos a la minería de datos

2.4.1 Introducción a la minería de datos

La **minería de datos o exploración de datos**, etapa de análisis del KDD (“Knowledge Discovery in Databases”), es el campo de las ciencias de la computación referido a un proceso el cuál intenta descubrir patrones en los datos. (Oded & Lior, 2010) (Kaspersky, 2019) Se basa en métodos de la inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos.

El principal objetivo de la minería de datos se basa en extraer conocimiento de un conjunto de datos y transformarla en una estructura que sea comprensible para su posterior utilización.

Este término es frecuentemente mal utilizado al referirse a cualquier tipo de datos a gran escala o también cualquier sistema de apoyo informático a la hora de tomar una decisión.

La tarea real de la minería de datos es el análisis automático o semi-automático de datos con el objetivo de conseguir extraer patrones relevantes hasta ahora desconocidos, como registros de datos (análisis clúster), registros pocos usuales (detección de anomalías) y dependencias (minería por reglas de asociación). Estos patrones se pueden ver como un resumen de los datos de entrada, y pueden ser utilizados para un análisis posterior, que nos harán lograr obtener resultados más precisos por un sistema de soporte de decisiones. Los pasos pertenecientes al proceso KDD son pasos previos al proceso de extracción de conocimiento, es decir, a la minería de datos.

Dependiendo de si los datos de entrada son etiquetados, podemos distinguir:

- **Aprendizaje supervisado:** Los datos de entrada han sido etiquetados de forma manual. Esto permite que los modelos puedan etiquetar datos nuevos a partir de los ya etiquetados. (Kotsiantis, 2007)

- **Aprendizaje no supervisado:** Al contrario que el mencionado anteriormente, los datos de entrada no son etiquetados, por lo que los modelos del caso anterior no funcionan y se usan técnicas alternativas. (Ghahramani, 2004)

- **Aprendizaje semi-supervisado:** Técnicas mixtas entre los dos conceptos explicados previamente. Por ejemplo, en un primer paso se aplica aprendizaje supervisado, y a continuación aprendizaje no supervisado para mejorar el modelo construido. (Zhu & Goldberg, 2009)

Además, contamos con dos tipos de minería de datos:

- **Minería de datos predictiva:** Se suele asociar al aprendizaje supervisado, y como objetivo obtener patrones que nos permitan etiquetar datos nuevos.
- **Minería de datos descriptiva:** Asociada al aprendizaje no supervisado, intenta descubrir el comportamiento y las interrelaciones entre los datos.

2.4.2 Proceso KDD (Knowledge Discovery in Databases)

Se compone de las siguientes fases (Azevedo & Santos, 2008):

- **Selección del conjunto de datos:** Tanto en lo que se refiere a las variables objetivo, como a las variables independientes, como al muestreo de los registros que haya disponibles.

- **Análisis de las propiedades de datos:** Histogramas, diagramas de dispersión, ausencia de datos y valores atípicos.

- **Transformación del conjunto de datos de entrada:** Preprocesamiento de los datos, con el objetivo de prepararlo para aplicar la mejor técnica de minería de datos que mejor se pueda adaptar a los datos y al problema.

- **Seleccionar y aplicar la técnica de minería de datos:** Se construye el modelo de clasificación o segmentación.

- **Extracción de conocimiento:** A través de una técnica de minería de datos, se obtiene un modelo de conocimiento, que representa patrones en función de los datos del problema. Se pueden utilizar distintas técnicas al mismo tiempo para obtener distintos modelos.

- **Interpretación y evaluación de los datos:** Se debe realizar una validación para comprobar que los resultados son válidos y satisfactorios.

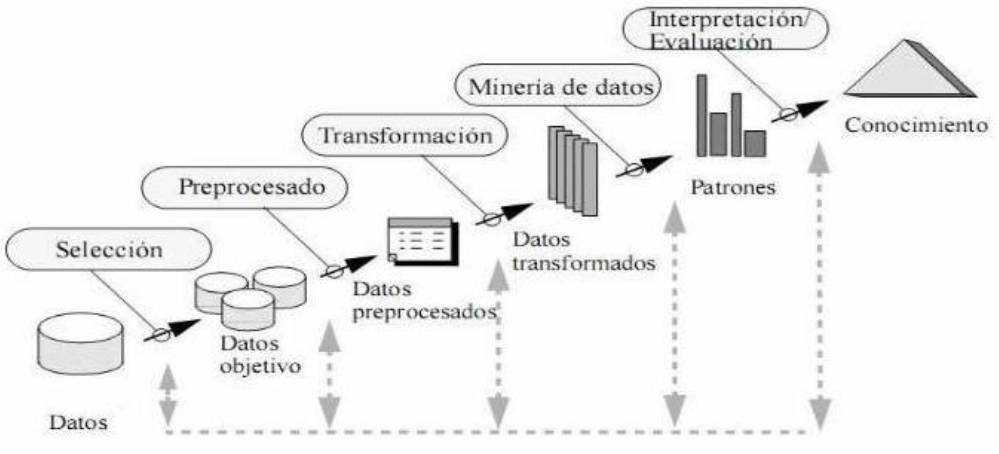


Figura 2.2 Proceso del KDD. Fuente: (Raynel, Sin fecha)

2.4.3 Tareas de la minería de datos

Existen multitud de tareas y técnicas diferentes sobre la minería de datos, entre todas ellas, las más relevantes son las siguientes (Berzal, 2014):

- **Clustering/Agrupamiento:** Trata de identificar un conjunto de categorías o conjuntos para describir los datos. Además, es aprendizaje no supervisado y forma parte de la minería de datos descriptiva.

- **Clasificación:** Trata de predecir la clase de un patrón a partir de patrones previamente clasificados. Forma parte del aprendizaje supervisado y de la minería de datos predictiva.

- **Regresión:** Predicción de valores reales. Aprendizaje supervisado y minería predictiva.

- **Series temporales:** Estudio de una variable a través del tiempo para, a partir de dicho conocimiento, poder realizar predicciones. Tarea de minería de datos predictiva.

- **Reglas de asociación:** Derivan de un tipo de análisis que extrae información por coincidencias. Minería de datos predictiva y no supervisada.

Aunque existen diferentes tareas y técnicas, para el desarrollo del proyecto, vamos a centrarnos en los de **Clasificación y Regresión**.

2.4.4 Clasificación

Dadas unas determinadas cantidades de instancias, la **clasificación** consiste en identificar a que clase pertenece cada una de ellas, partiendo de una colección de datos de entrenamiento con instancias cuyas categorías son conocidas.

En nuestra vida diaria hay muchos aspectos en los que emplean clasificadores, por ejemplo, durante el uso de la app Spotify, dependiendo de la música que escuchemos, nos clasifica la música de modo que nos recomendará canciones en base a lo que normalmente escuchamos. Un ejemplo más claro es el del email, en el que el spam es dirigido automáticamente a la carpeta de correo no deseado, o en caso de no ser spam a la carpeta de recibidos. Como podemos ver, hay una gran cantidad de problemas que pueden ser modelados como si se tratasen de problemas de clasificación.

Los modelos deben entrenarse, y los datos que se analizan contienen características que se pueden dividir en categorías. Hay que considerar también que no todos los algoritmos de clasificación funcionan para todos los datos, si no, que hay algoritmos que valen para valores discretos, otros para reales y enteros, etc. Por lo que tendrían que ser discretizados en grupos en algunas ocasiones.

Una vez tengamos la información representada a través de atributos, el algoritmo de machine learning usa dichos datos, y generará una función o clasificador, que tenga de entrada las características de dicha clase, y como salida la clase que se obtenga durante la predicción de la función que actúa como clasificador.

Durante la etapa de predicción, el clasificador tras lo aprendido de la información, predecirá a que clase pertenece cada nueva instancia. Es realmente importante que el conjunto de características de los datos de entrenamiento y de los datos nuevos sean iguales.

De igual forma, a la hora de realizar este proceso, si tenemos una determinada cantidad de datasets, y el clasificador se entrena con un rango de ellos, excepto el de prueba, se debe promediar sobre el resto de conjuntos de datos poniendo a prueba todos los restantes para obtener unos resultados precisos y no descompensados.

En cuanto a los métodos de clasificación, existen clasificadores lineales, redes neuronales, árboles de decisión, etc.

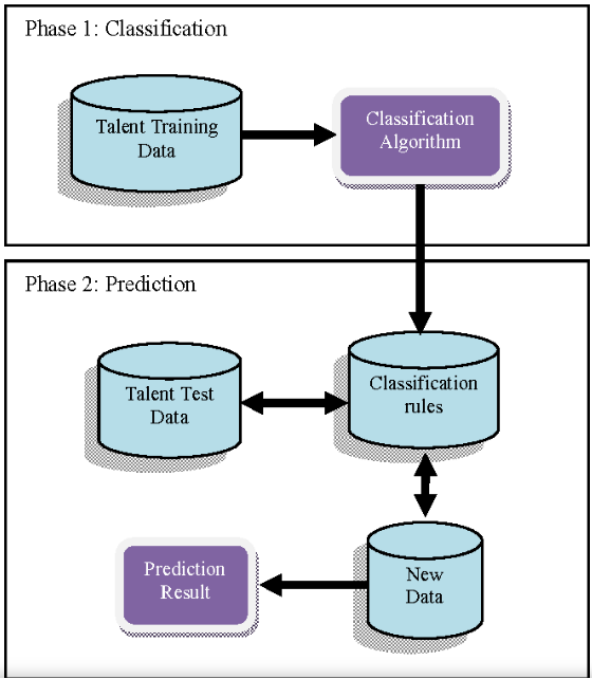


Figura 2.3 Proceso de clasificación. Fuente: (Hamidah, Abdul, & Zulaiha, 2009)

2.4.5 Regresión

La **regresión** es una herramienta muy útil para realizar predicciones o estimaciones. Para ello se basa en fórmulas matemáticas que demuestran la relación entre dos o más datos de forma consistente. En nuestro caso se relacionan las clases reales de un determinado conjunto de datos de entrenamiento y otro de test.

Se basa en el cálculo diferencial y por esa razón los resultados obtenidos no son totalmente exactos, si no que más bien se trata de una aproximación.

Es muy empleada en estadística, por ejemplo, empleando un análisis regresivo podemos predecir datos futuros a partir de series antiguas, y cuanto más exacta sea la ecuación empleada, más eficiente y fiable será la predicción.

Como en nuestro caso partimos de grandes volúmenes de datos, y los aplicados a regresión contienen clases con valores reales, hemos aplicado una media por cada instancia de cada conjunto de datos del que obtenemos una predicción. Esa media se aplicará sobre todos los modelos obtenidos al aplicar un método de la librería de MLlib de forma que, aunque se apliquen diferentes métodos de regresión, obtendremos al final una media generalizada para nuestro ensemble.

Obviamente hay muchos tipos de ecuaciones que son posibles aplicar a regresión, por esa misma razón es necesario tener un alto grado de conocimiento de las matemáticas, aparte de una gran cantidad de datos observados.

Un uso de regresión en la vida cotidiana, es por ejemplo en el caso de las empresas, para realizar previsiones de ventas, ingresos o evolución de mercados. Y no solo en el entorno empresarial, si no en muchos otros tipos de ámbitos como ingeniería, biología, medicina, etc.

De hecho, también la usan software de inteligencia artificial, de modo que puedan predecir resultados, para en futuras ocasiones evitar cometer los mismos errores.

2.4.6 Validación

El objetivo de la **validación** consiste en estimar el nivel de ajusto de un modelo a un cierto conjunto de datos de prueba independientes de las utilizadas al entrenar el modelo.

Hay diversos tipos de validación, pero nos centraremos en una que es de la más importantes, como es la **Validación Cruzada** y en sus diferentes formas de aplicación.

2.4.6.1 ¿Qué es la validación cruzada?

Esta técnica trata de evaluar los resultados en un determinado análisis, y se encarga de que los datos de entrenamiento y prueba son independientes entre sí. (Devijver & Kittler, 1982) (Jean-Philippe, 2014)

Para estimar la precisión de un modelo se hace uso de este proceso, a partir de un conjunto de datos de muestra, se divide en dos subconjuntos, uno en el que los datos se utilizan de entrenamiento para el modelo, y el otro conjunto de datos sirve de validación. Este método se repetirá sobre todo el conjunto de datos, en el que vayan cambiando los conjuntos de datos de prueba y entrenamiento, obteniendo así una media más genérica y no tan dependiente de una única evaluación.

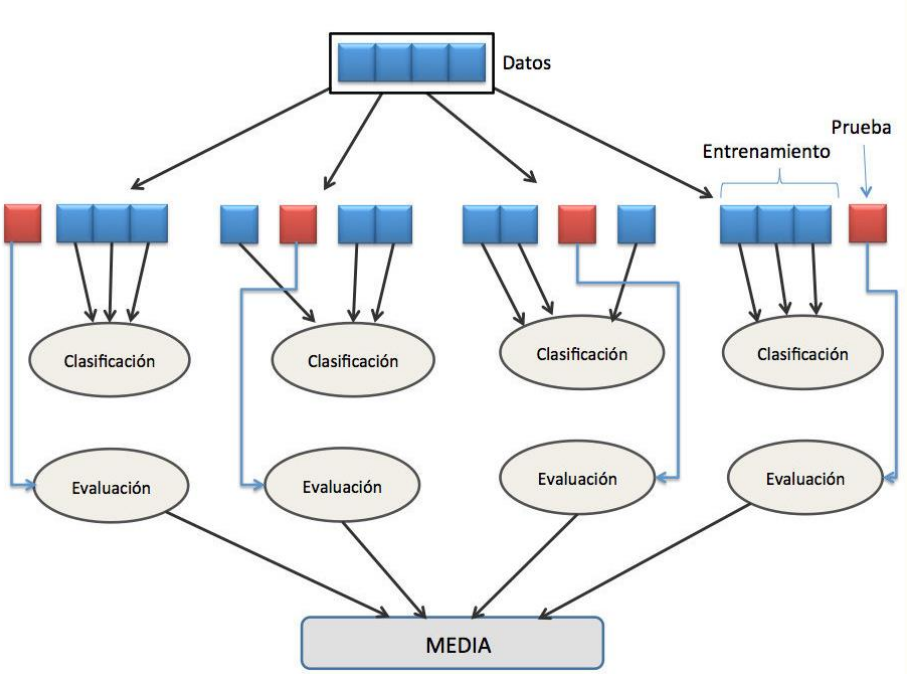


Figura 2.4 Esquema k-fold cross validation. Fuente: (Joan, 2011)

2.4.6.2 Validación cruzada dejando uno fuera

(Charles, 2011) Esta técnica implica dividir el conjunto de datos tantas veces como datos haya en el conjunto, de manera que en cada iteración se escogerá una muestra de datos que servirá de prueba, y el resto de muestras se utilizarán para el entrenamiento.

Con este método obtendremos una media contundente, y con un error muy bajo, no obstante, el coste computacional es elevado, ya que se deben de realizar una elevada cantidad de iteraciones, y en cada una se deben realizar análisis de datos de entrenamiento y prueba. El resultado final se obtiene realizando la media de todos los valores de errores obtenidos en cada iteración, con la expresión:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i.$$

Un ejemplo claro de este tipo de validación cruzada es el de la figura 2.4, y es el proceso del que haremos uso para el desarrollo del proyecto.

2.4.6.2 Validación cruzada de K iteraciones

(Payam, Lei, & Huan, 2008) En la validación cruzada de K iteraciones o *K-fold cross-validation* los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto (K-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba.

Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba, pero aun así tiene la desventaja de que es lento computacionalmente. Lo más común es utilizar validación cruzada de 10 iteraciones. En este tipo de validación cruzada con respecto a la anterior, la expresión cambia:

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i.$$

3. PARADIGMAS DE COMPUTACIÓN DE BIG DATA

3.1 Introducción al Big Data

Big Data hace referencia a colecciones de datos grandes y complejas, por lo que no se puede hacer uso de aplicaciones informáticas tradicionales de procesamiento de datos, si no que se necesitan otras mucho mejores para tratarlas de forma adecuada. La revolución de los datos masivos (Marilín, 2013).

Actualmente el término es usado en cuanto a lo referido de extraer valor de datos almacenados, y realizando predicciones de los patrones observados. Las dificultades se encuentran en la recolección y almacenamientos de los datos (Kusnetzky, 2010), búsquedas y análisis, y finalmente en las visualizaciones y representaciones. Pero el trabajar esos volúmenes de datos tan grandes es necesario en muchos casos para informes estadísticos y modelos predictivos que nos ayudarán en los análisis de negocios, enfermedades infecciosas, etc.

Este término se ha usado desde los 90, y muchos otorgan crédito a *John Mashey* (Mashey, 1998). Como bien he explicado anteriormente, el concepto hace referencia a un gran volumen de datos el cuál supera la capacidad del software para procesarlos en un tiempo determinado. Conforme pasa el tiempo, el volumen de datos sigue en aumento.

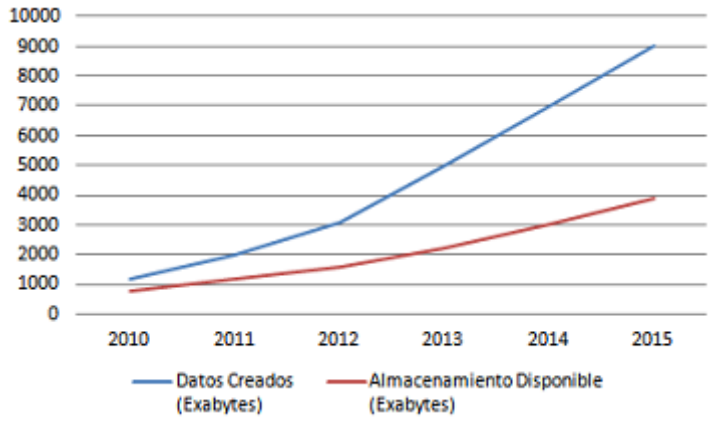


Figura 3.1 Datos creados vs Almacenamiento en Big Data. Fuente: (Tecnología-e-innovación, 2012)

Por lo mencionado anteriormente, siempre al definir Big Data, se recurre a las tres Vs de este término: Volumen, Velocidad y Variedad (Laney, 2001). Aunque algunos autores añaden algunas más, como Veracidad, Viabilidad, Validez, etc. Pero vamos a considerar únicamente las tres comentadas que son las principales a tener en cuenta.

En cuanto al **Volumen** podemos considerar que el volumen de datos que se maneja es muy grande, de muchos Terabytes. La cantidad de datos se multiplica con el tiempo y los sistemas tradicionales no son capaces de soportar toda esa carga. Por primera vez se están moviendo las aplicaciones a los datos, y no al revés como venía siendo normal (O'Reilly, 2014).

La **Velocidad** en Big Data se ocupa de datos que se generan a velocidad superior de un dato por segundo, o incluso en tiempo real. Por ejemplo, todas las transacciones que son generadas en la Bolsa de Wall Street a lo largo del día, y dichas operaciones se tienen que realizar en el menor tiempo posible. Es impresionante la cantidad de información que se genera en nuestro alrededor sin darnos cuenta, en un minuto es posible que se realicen más de cinco millones de búsquedas en Google, se suban a YouTube más de 400 horas de contenido, etc.

Por último, en cuanto a la **Variedad**, en Big Data los datos son variados, y no solo se trabaja con texto y números, si no con otros muchos tipos de datos como

fotografías, vídeos, entre otros. Y todos esos datos con frecuencia no son estructurados.

Aunque hay autores que no solo mencionan las tres características anteriores, si no, que añaden alguna más, teniendo en cuenta por ejemplo también la Veracidad, Viabilidad, Validez, etc. (Zikopoulos, Eaton, deRoos, Deutsch, & Lapis, 2011)

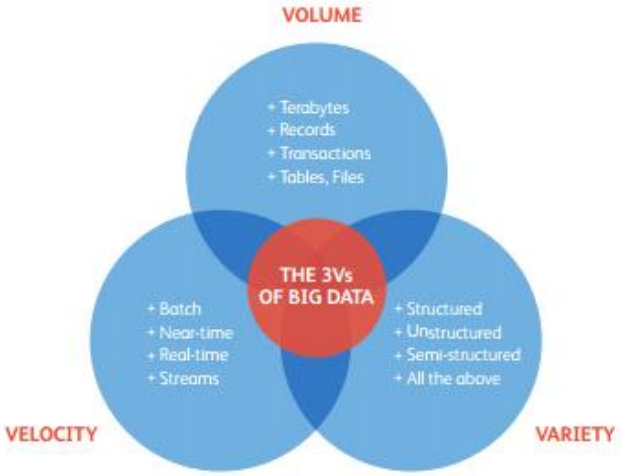


Figura 3.2 Tres Vs en Big Data. Fuente: (Researchgate, 2018)

Para terminar, explicaré un ejemplo de uso del Big Data en la actualidad (Datacentric, 2018). Hoy en día prácticamente todo el mundo usa alguna aplicación de pago donde ver series o películas. Vamos a entrar en el caso de Netflix, plataforma donde hay una gran flexibilidad para los usuarios donde pueden elegir qué ver en un gran catálogo de series, películas e incluso documentales.

Netflix hace uso del Big Data, puesto que recopilan información de todos sus clientes, y de acuerdo a lo visto, te van recomendando nuevas series y películas que ver. O incluso recomendaciones globales, por ejemplo, *'series más vistas'*.

Aparte de ser una plataforma, Netflix también realiza sus propias series y películas, y gracias al Big Data son capaces de conseguir los patrones adecuados para saber que le interesa al público y realizar una serie/película que

logre ser un gran éxito. Por lo que con la ayuda del Big Data y la información de sus clientes, pueden decidir el contenido de los capítulos, e incluso de su distribución.

3.2 Apache Hadoop

Apache Hadoop es un framework de licencia libre que es capaz de soportar aplicaciones distribuidas y es capaz de trabajar con petabytes de datos y multitud de nodos.

Hadoop es un proyecto de alto nivel Apache en el que colaboran tanto en la elaboración como utilización una comunidad global, y se utiliza el lenguaje de programación Java. Además, ofrece un servicio de alta disponibilidad. Implementa un paradigma computacional llamado MapReduce, en el cuál la aplicación se divide en pequeños fragmentos de trabajo, y cada uno se puede ejecutar en cualquiera de los nodos del clúster.

Originalmente fue desarrollado para ayudar en la distribución del motor de búsqueda llamado Nutch (Doug & Mike, 2002).

3.2.1 HDFS (Hadoop Distributed File System)

(Paloma, 2017) Se trata del sistema de ficheros que almacena los datos en Hadoop, y para afrontar la dificultad que tienen los grandes volúmenes de datos, HDFS es capaz de dividir los datos en bloques, y luego los redirecciona entre los diferentes nodos de datos que forman el clúster.

Aparte, de distribuir los bloques entre los nodos, también los replica para evitar problemas de pérdidas de datos. Con todo esto, se consigue obtener una alta escalabilidad.

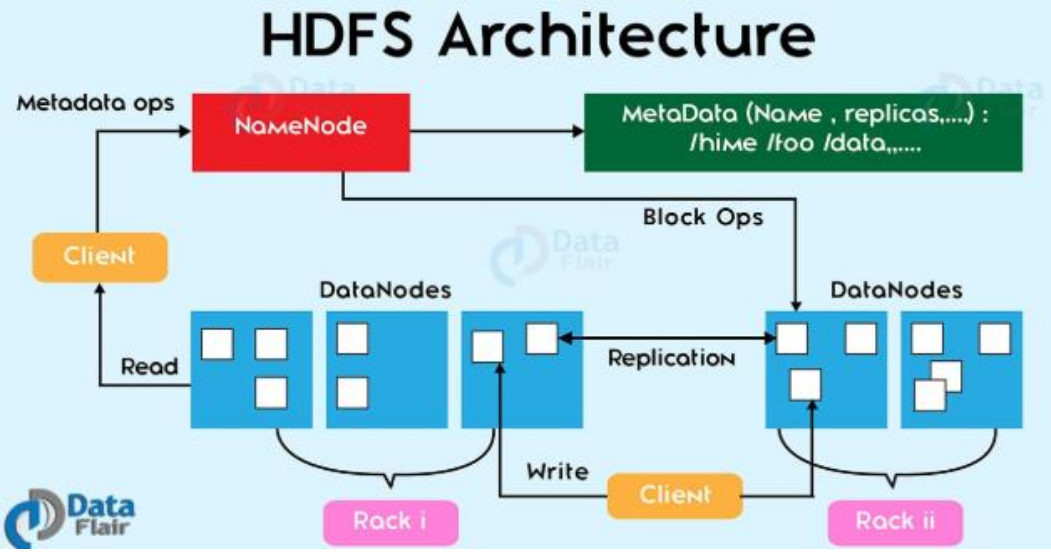


Figura 3.3 Arquitectura HDFS. Fuente: (Data-flair.training, 2020)

3.2.2 MapReduce

(Paloma, 2017) Este paradigma trabaja sobre HDFS y cuenta con dos fases, Map y Reduce:

- **Map:** Se encarga de dividir la tarea en subtareas y las ejecuta entre los diferentes nodos existentes.

- **Reduce:** Recoge la información obtenida de la función Map, y seguidamente agrupa los datos para conseguir una solución final.

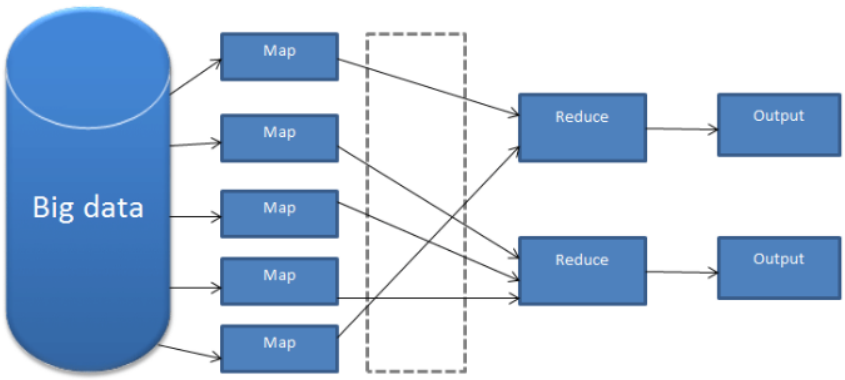


Figura 3.4 Map Reduce. Fuente: researchgate.net. Fuente: (Mohammed, 2016)

3.3 Apache Spark

Apache Spark (Zaharia, Chowdhury, Franklin, & Shenker, 2010) es un framework de programación con licencia Open Source para procesamiento de datos distribuidos con el objetivo de ser rápido y de propósito general (Karau, Konwinski, Wendell, & Zaharia, 2015).

Logra una alta velocidad al ejecutar cargas de trabajo en un tiempo bastante reducido, puesto que trabaja en memoria RAM, además consigue un alto rendimiento tanto para datos de lotes como para datos de transmisión. Ofrece facilidad de uso ya que permite distintos modos de programación, y también se nos facilita la creación de aplicaciones paralelas.

Contiene diferentes tipos de librerías, como **SQL, DataFrames, MLlib, GrapghX y Spark Streaming**, y su ejecución es posible en todas partes, de forma independiente o en la nube. Sobre todo, es utilizado por empresas destinadas al procesamiento de grandes colecciones de datos.

Una de las grandes cualidades de Spark, es su persistencia de datos en memoria para todas las operaciones. Cada nodo contiene las particiones que tiene en memoria y las reutiliza para otras acciones de esa colección de datos, y todo esto permite que las siguientes operaciones a realizar se efectúen más rápidamente.

Spark utiliza evaluaciones perezosas, esto conlleva que no efectúe ningún trabajo, a excepción de tener que hacerlo, y eso nos permite evitar el uso innecesario de memoria, lo que nos facilita trabajar con Big Data. Las transformaciones se evalúan de manera perezosa, y el trabajo real ocurre cuando se realiza una acción.

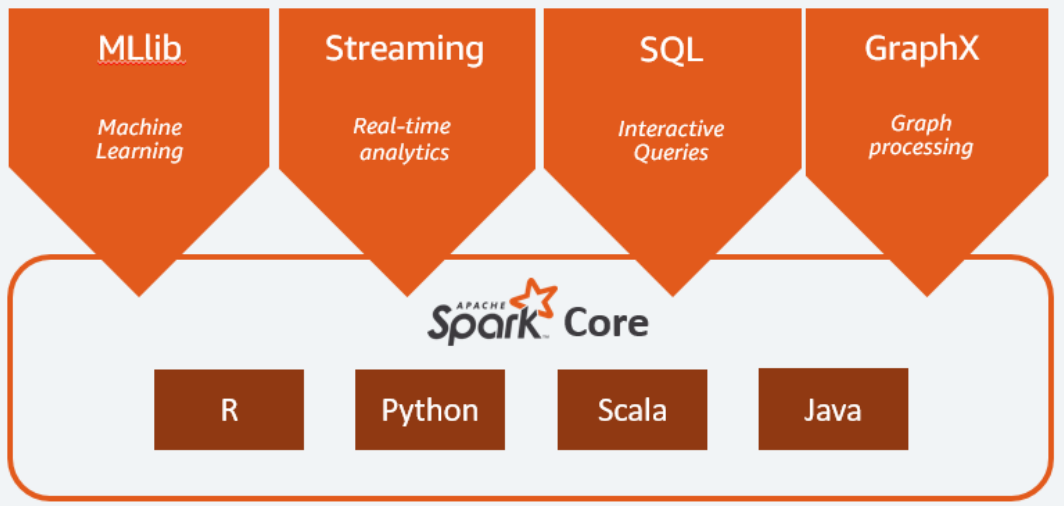


Figura 3.5 Estructura de Spark. Fuente: datanami.com. Fuente: (Alex, 2019)

3.3.1 Tipos de estructuras que permiten procesamiento paralelo

Con respecto a los **RDD (Conjuntos de datos distribuidos resilientes)**, son unos conjuntos de elementos que se ejecutan en paralelo en los nodos de un clúster y que son tolerantes a fallos. Estos RDD se crean partiendo de una colección Scala que ya estaba incluida en el programa del controlador y realizándose una transformación. O también se puede realizar una referencia a una colección de datos en un sistema de almacenamiento externo.

También se permite que Spark conserve un RDD en memoria, reutilizándolo de manera mucho más eficiente en operaciones que se efectúan en paralelo. Por último, los RDD ante algún fallo en los nodos, pueden recuperarse automáticamente.

Los **dataset** son colecciones de datos distribuidos que tienen ya una estructura, a diferencia de los RDD, que son conjuntos de datos desestructurados y definidos como una colección de elementos tolerante a fallos y son capaces de operar en paralelo. Estos conjuntos de datos son tan grandes que aplicaciones de procesamiento de datos tradicionales son incapaces de procesarlos debido a la gran cantidad de datos contenidos en la tabla o matriz.

En cambio, los **dataframes** son estructuras de datos tabulares que se componen de columnas y filas ordenadas. Cada fila se corresponde a un objeto de la muestra y cada columna a una variable, y en cuanto a esta característica de organización es similar a los datasets.

La principal diferencia radica en los valores que aceptan las columnas, ya que los dataframes admiten cualquier tipo de datos y los datasets no. Por lo que esta organización nos facilitará y hara más sencillo consultar, modificar o transformar el conjunto de datos contenidos en la hoja de datos.

Este último tipo de estructura es la que mayor porcentaje de aplicación contiene en el desarrollo del proyecto, ya que nos ha permitido una mayor facilidad, eficiencia y efectividad.

3.3.2 SHUFFLE

Función de Spark para repartir datos de forma que se puedan agrupar de forma diferente en las particiones. Esto implica copiar datos, y esto provoca que la combinación sea una operación compleja y con un gran coste, debido a que involucra E/S de disco, de red y la serialización de los datos.

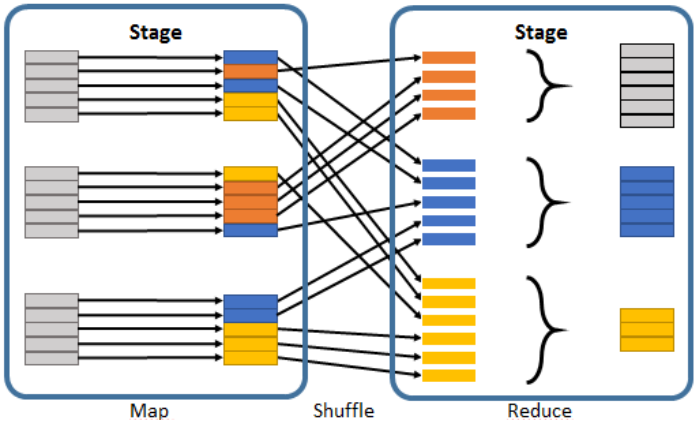


Figura 3.6 Aplicación de Shuffle. Fuente: (Usuario-StackOverflow, 2016)

3.3.3 DAG

Como explicamos anteriormente, Map Reduce consta únicamente de dos pasos, en cambio en Spark puede haber múltiples pasos y en forma de grafo, DAG (Directed Acyclic Graph).

Tras lanzar una aplicación Spark, se crea un DAG y cada nodo se corresponde con una etapa que contiene ciertos tipos de operaciones que serán aplicados al RDD correspondiente.

Como bien explicamos previamente, existen operaciones de transformación y acción, y las mencionadas primeramente son operaciones perezosas, que únicamente se ejecutan tras lanzar una acción, lo que hace que la paralelización sea más eficiente.

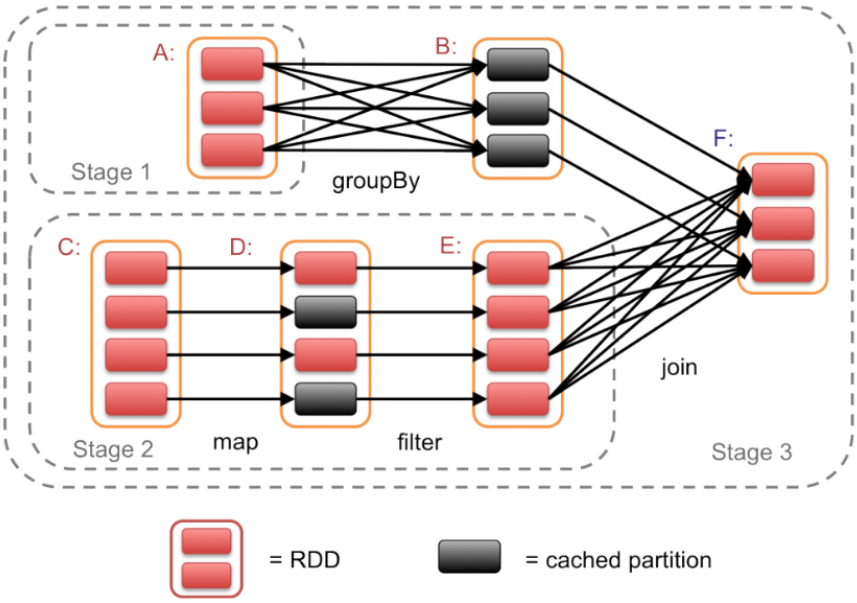


Figura 3.7 DAG en Spark. Fuente: (SoHPC-Team, 2015)

3.3.4 MLlib

(Datahack, 2019) Para el desarrollo del proyecto, vamos a hacer uso de la biblioteca **MLlib**, librería de Machine Learning, para poder acceder al uso de diferentes tipos de algoritmos de este concepto.

Estos algoritmos aprovechan Spark para que puedan ser escalados y paralelizados. Una vez instalada la librería nos encontramos con que existen dos tipos de APIs, API principal o Spark ML, basada en DataFrames, y API original o Spark MLlib, que aprovecha los RDDs. Pero principalmente se usa la primera, ya que utiliza estructuras más rápidas y sencillas.

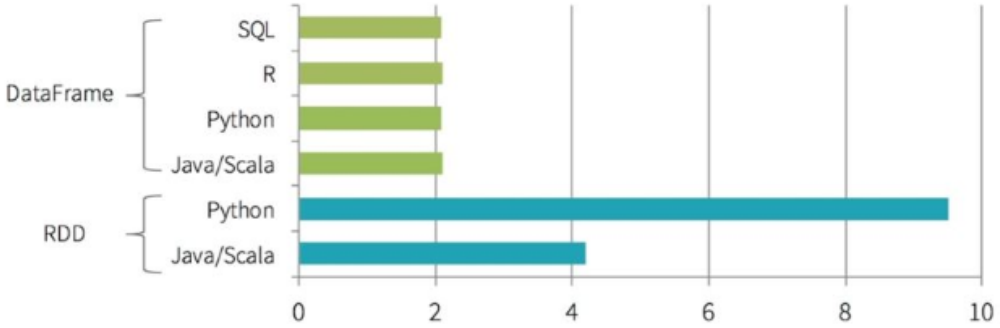


Figura 3.8 Ejemplo de tiempo de ejecución para una carga de trabajo de agregación. Fuente: (Datahack, 2019)

Aunque existen distintos tipos de algoritmos de Machine Learning, como Clasificadores y Regresores, nosotros vamos a centrarnos en los **Clasificadores**.

Debido al aprendizaje automático, los datos se ordenan según ciertas categorías, y hay que ver como asignar **etiquetas** a esos datos de entrada, y esto se realiza con dicho algoritmo de clasificación. En base a dichas etiquetas deberemos lograr clasificar y obtener un resultado sustento a los datos de los que partíamos.

3.3.5 Técnicas clasificación y regresión de MLlib

- **Logistic Regression:** Método para predecir una respuesta **binaria**. Para problemas de clasificación, el algoritmo genera un modelo de regresión binario. También puede usarse para r**egresión multinomial** para aquellos **problemas de clasificación multiclase**, pero en nuestro caso únicamente

lo hemos implementado para problemas binarios, ya que el otro no funcionaba correctamente.

- **Decision Tree:** Son una familia de métodos para tareas de aprendizaje automático. Son fáciles de interpretar, manejar características categóricas, etc. Se encuentran entre los mejores para tareas de **clasificación y regresión**, además de admitir **tanto clases binarias como multiclase**, utilizando características continuas y categóricas. La implementación divide los datos por filas, lo que permite un entrenamiento distribuido con millones o incluso miles de millones de instancias.

- **Random Forest:** Son conjuntos de árboles de decisión. Combinan muchos árboles de decisión para reducir el riesgo de sobreajuste. Admite **clasificación binaria y multiclase, y para regresión**, empleando características continuas y categóricas.

- **MPC (Multilayer Perceptron Classifier):** Clasificador basado en la red neuronal artificial de alimentación directa, y este método consta de múltiples capas de nodos, estando cada una de las capas conectadas a la siguiente capa en la red. Los nodos en la capa de entrada representan los datos de entrada y todos los demás nodos asignan entradas a salidas mediante una combinación lineal de las entradas con los pesos y sesgos del nodo y aplicando una función de activación. Puede ser empleado tanto para **clasificación binaria como multiclase**.

- **Naive Bayes:** Familia de clasificadores probabilísticos simples y multiclase basado en la aplicación del teorema de Bayes con fuertes supuestos de independencia entre cada par de características. Puede ser entrenado de manera muy eficiente, ya que con un solo paso sobre los datos que sirven de entrenamiento, calcula la distribución de probabilidad condicional de cada característica dada cada etiqueta. Para la predicción, aplica el teorema de Bayes para calculas la distribución de probabilidad condicional

de cada etiqueta dada una observación. Sirve **tanto para clasificación binaria como para multiclase**.

- **SVM (Linear Support Vector Machine):** Construye un hiperplano o un conjunto de hiperplanos en un espacio de dimensión alta o infinita, que puede usarse para clasificación, regresión u otras tareas. Intuitivamente, se logra una buena separación mediante el hiperplano que tiene la mayor distancia a los puntos de datos de entrenamiento más cercanos de cualquier clase, ya que en general cuanto mayor es el margen, menor es el error de generalización del clasificador. Admite la **clasificación binaria** con SVM lineal.

- **GBT (Gradient-Boosted Trees):** Conjuntos de árboles de decisión: Los GBT capacitan iterativamente los árboles de decisión para minimizar una función de pérdida. Admite para **clasificación binaria** y para la **regresión**, utilizando características continuas y categóricas. Aunque únicamente se ha implementado para regresión ya que daba problemas para clasificación. **Aún no admiten clasificación multiclase**.

- **Isotonic:** Pertenece a la familia de **algoritmos de regresión**. Se puede ver como un problema de mínimos cuadrados bajo restricción de orden. Esencialmente, es una función monotónica que mejor se ajusta a los puntos de datos originales.

3.4 Hadoop o Spark: ¿Cuál debemos elegir?

Apache Spark, mejora al paradigma de Map Reduce de Hadoop, contiene abstracciones a más bajo nivel y permite lenguaje SQL. En cuanto a las comparaciones de ambas plataformas contienen las siguientes diferencias (Victor, 2019):

- Si nos referimos al **rendimiento**, Spark es más rápido ya que trabaja en memoria RAM, reduciendo de manera considerable tiempos de procesamiento. Además, cuenta con un **planificador DAG**, que establece tareas a realizar y optimiza las operaciones.

- En **complejidad de desarrollo**, Map Reduce es implementado en Java además de ser compatible con otros lenguajes. Pero Spark es más sencillo ya que hay un gran esfuerzo por parte de la comunidad con el objetivo de mejorar dicho framework.

- En cuanto al **coste**, Spark hace uso de un clúster que tenga mucha memoria RAM, y Map Reduce un clúster que contenga más discos y que sean más rápidos para el procesamiento.

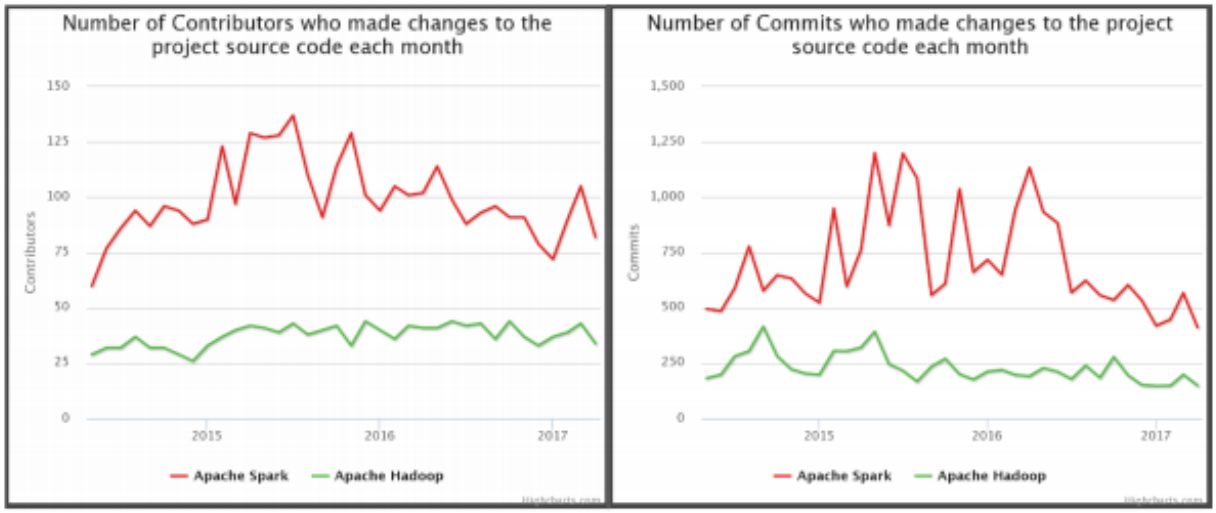


Figura 3.9 Nº de desarrolladores de ambas plataformas. Fuente: (Borja, 2017)

Aunque a simple vista, Spark nos da la impresión de que es el mejor framework a la hora de trabajar con Big Data, no contiene un sistema de ficheros distribuido, y por eso se apoya en Hadoop.

El sistema de fichero de Hadoop está integrado al descargar Spark, por lo que obtenemos las mejores de las soluciones al conseguir tener al mismo tiempo **Spark + Apache Hadoop**.

	Récord Hadoop	Récord Spark	Spark 1 petabyte
Tamaño datos	102.5 TB	100 TB	1000 TB
Tiempo	72 min	23 min	234 min
Nodos	2100	206	190
Características hardware nodos	2 2.3Ghz, Hexcore Xeon E5-2630, 64 GB memoria, 12x3TB discos	32 vCores - 2.5Ghz, Intel Xeon E5-2670 v2, 244GB memoria, 8x800 GB SSD	-
Procesadores	504000 físicos	6592 virtuales	6080 virtuales
Rendimiento disco	3150 GB/s	618 GB/s	570 GB/s
Red	Centro de datos 10Gbps	Virtual (EC2) 10 Gbps	-
Velocidad procesamiento medio	1.42 TB/min	4.27 TB/min	-
Velocidad procesamiento por nodo	0.67 GB/min	20.7 GB/min	22.5 GB/min

Figura 3.10 Comparativa de récords. Fuente: (Borja, 2017)

4. ENSEMBLES

4.1 Introducción a los ensembles

Con el **modelado de ensembles**, podemos llegar a conseguir una mejora de rendimiento en nuestro modelo, combinando las decisiones de los múltiples modelos.

Se va a explicar el concepto de **ensemble learning** a través de un ejemplo. Cuando un nuevo videojuego sale a la venta, previamente es probado por testers, con el fin de poder obtener calificaciones y críticas y que esto les pueda servir a la hora de introducir a los desarrolladores nuevos parches o básicamente tenerlo en cuenta para futuras entregas. No solamente se toma la opinión de un tester, ya que con eso no podríamos obtener el objetivo deseado, e incluso podría influir que dicho tester no le gusten dicho tipo de juegos y esto afectaría. Por lo que se contratan un conjunto de testers, pueden ser personas expertas y no expertas, por lo que tendríamos una diversidad de opiniones y calificaciones.

En este último caso, las respuestas serían más genéricas y diversificadas, y este enfoque es mejor para obtener calificaciones honestas. Y esta diversificación en Machine Learning se logra mediante la técnica ensemble learning.

4.2 Técnicas simples de ensemble

4.2.1 Votación Máxima

Esta técnica se usa principalmente para problemas de clasificación. Se utilizan múltiples modelos para hacer predicciones para cada punto de datos. Las predicciones con mayoría son las consideradas como predicción final.

Considerando el ejemplo anterior, si se utilizan 5 testers para un videojuego, 3 de ellos lo califican con 5 puntos sobre 5, y los otros dos con 4 puntos, por mayoría se obtiene la calificación de 5 puntos.

4.2.2 Promedio

Similar a la técnica anterior, pero realizando el promedio de predicciones de todos los modelos, y el resultado es usado como predicción final. Esta técnica se usa tanto para problemas de regresión, como para calcular probabilidades de problemas de clasificación.

Continuando el ejemplo anterior, se realizaría el promedio de las 5 calificaciones.

4.2.3 Promedio ponderado

Como se ha explicado en la introducción, puede que de todos los testers del videojuego, se tengan tanto testers expertos como no expertos, para que los resultados sean más genéricos.

Con esta técnica se tendrá en cuenta dicha diferencia entre cada uno, y se le dará una ponderación según el nivel del tester, que se multiplicará por su calificación dada, y la suma de todos los resultados será la predicción final.

4.3 Errores en ensemble learning

El error de cualquier modelo se puede expresar matemáticamente:

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Figura 4.1 Error en ensemble learning. Fuente: (Tavish, 2015)

- **Error de Sesgo:** Nos ayudan a identificar cuánto, en promedio, las predicciones son diferentes de los valores reales. Un alto error de sesgo, nos indica que el modelo es de bajo rendimiento.

- **Variación:** Cuantifica cómo las predicciones realizadas en la misma observación son diferentes entre ellas. Un modelo con alta variación se sobreajustará a la población de entrenamiento, y no tendrá un buen enfoque en aquellas observaciones que vayan más allá del entrenamiento.

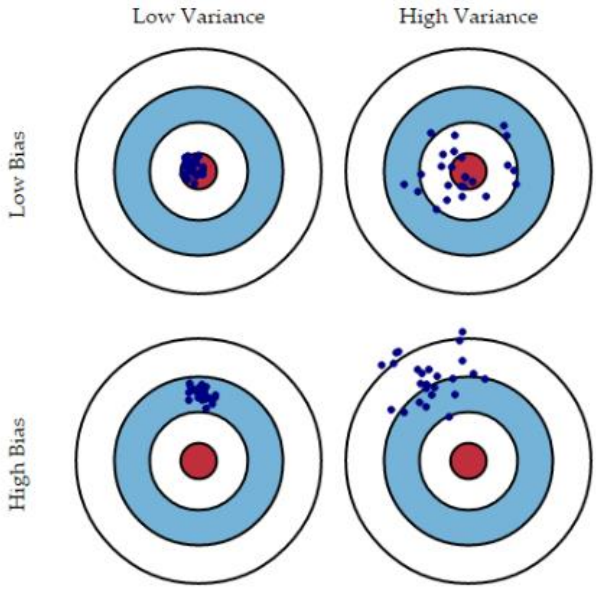


Figura 4.2 Variaciones en los errores. Fuente: (Scott, 2015)

A mayor complejidad del modelo, se podrá ver una reducción en el error, ya que el modelo obtendrá un sesgo menor. No obstante, a medida que se hace más complejo, el modelo comenzará a sufrir una alta variación.

Por lo que lo mejor es conseguir un equilibrio entre ambos errores, consiguiendo una buena compensación.

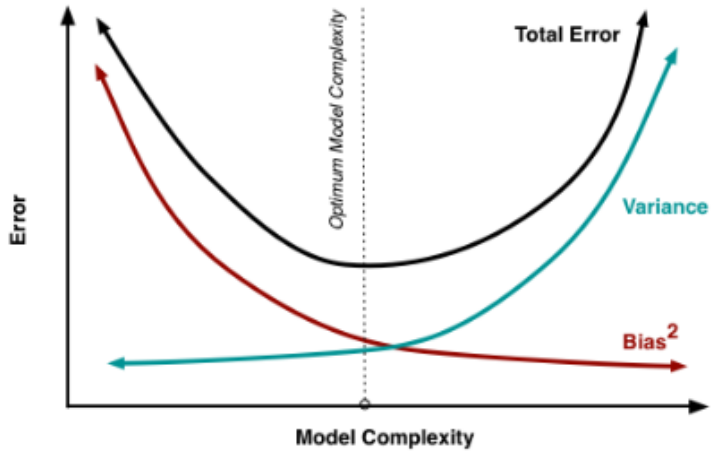


Figura 4.3 Complejidad del modelo y error. Fuente: (Scott, 2015)

4.4 Técnicas avanzadas de ensemble

Ahora hablaremos de diferentes técnicas que crean múltiples modelos y luego los combinan para producir mejores resultados. Los métodos de ensemble normalmente producen soluciones más precisas que un solo modelo.

El término *modelo* hace referencia a la salida del algoritmo que se entrenó con datos, y éste se usa para hacer predicciones. Cuando estos modelos se usan como entradas de métodos de ensemble, se denominan *modelos base*.

4.4.1 Stacking

Técnica de aprendizaje de ensembles que hace uso de predicciones de múltiples modelos (por ejemplo, árbol de decisión, SVM, etc.) para construir a partir de estos, un nuevo modelo.

Todos los algoritmos se entrenan con los datos disponibles, y posteriormente a través de un algoritmo combinados se entrena para obtener una predicción final usando todas las predicciones del resto de algoritmos.

Stacking generalmente produce un rendimiento mejor que cualquiera de los modelos entrenados y se ha utilizado con éxito tanto en tareas de aprendizaje supervisado como no supervisado.

El modelo final se utiliza para hacer predicciones en el conjunto de prueba. Y con esto se puede conseguir una disminución de sesgo y error de varianza.

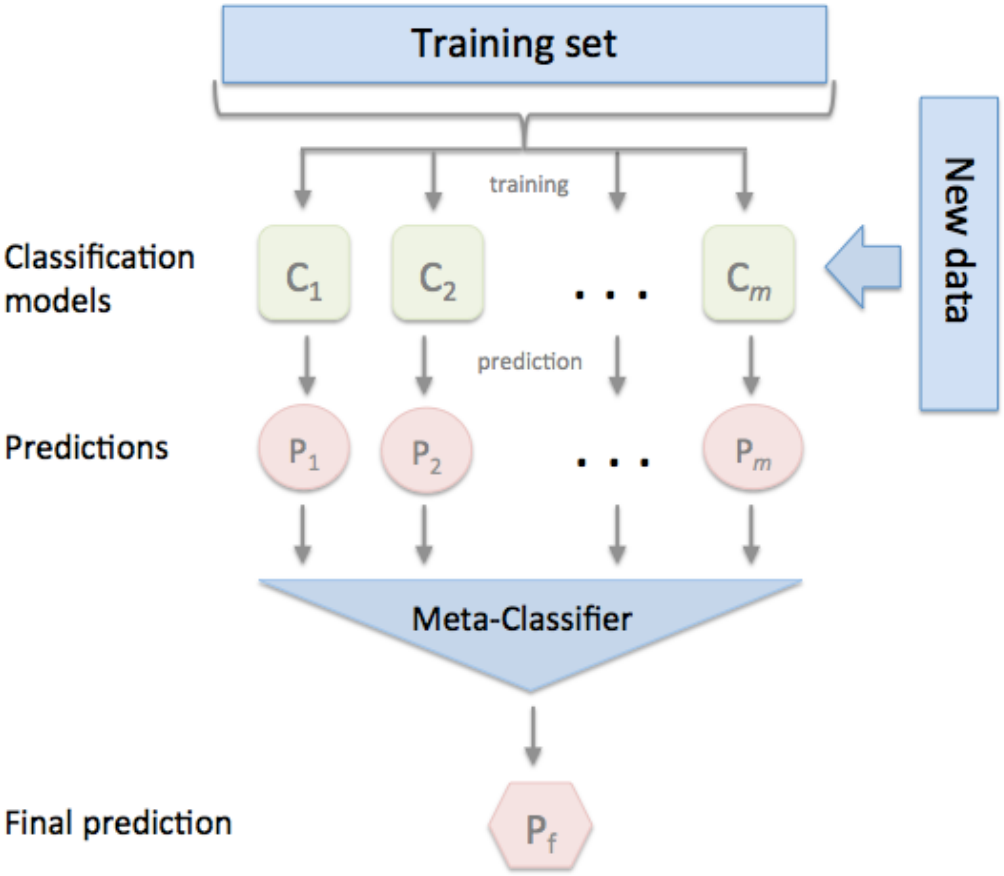


Figura 4.4 Ensemble Stacking.

4.4.2 Boosting

Se trata de un proceso secuencial, donde cada modelo trata de corregir los errores del modelo anterior.

Consiste en combinar los resultados de varios clasificadores débiles para obtener un clasificador robusto, teniendo en cuenta que cada uno tenga diferente peso en función de la exactitud de sus predicciones.

La idea es que en cada iteración se incremente el peso de los objetos mal clasificados por el predictor en esa iteración, por lo que en la construcción del próximo predictor estos objetos serán más importantes y será más problable clasificarlos correctamente.

Esta técnica disminuye el error de sesgo y crea modelos predictivos sólidos, no obstante, en muchas ocasiones se puede ajustar demasiado a los datos de entrenamiento.

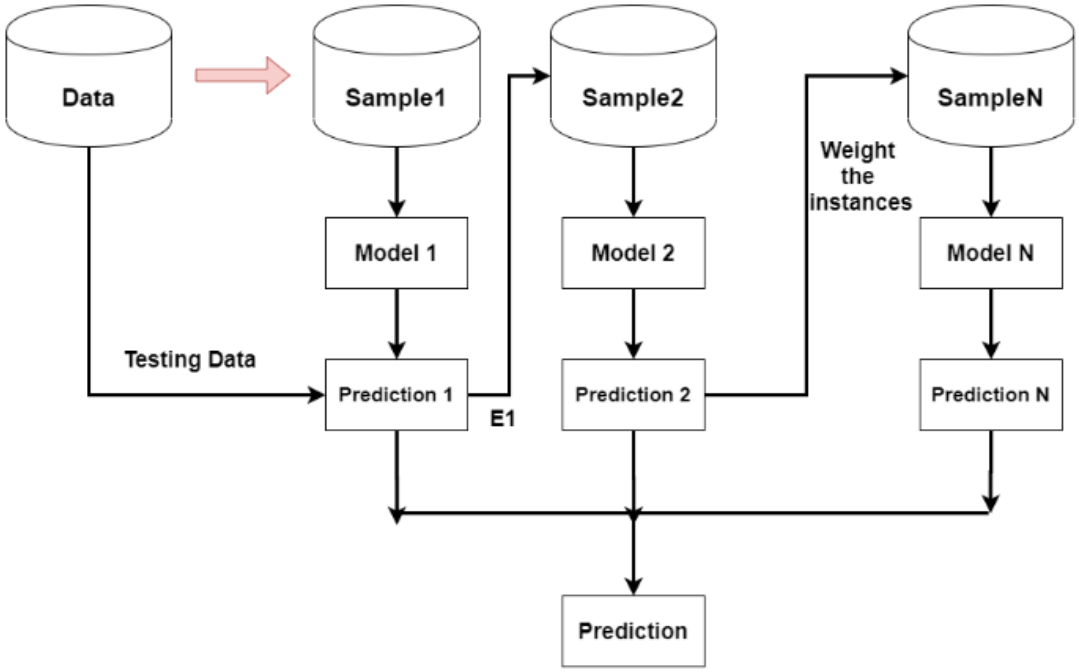


Figura 4.5 Ensemble Boosting.

4.4.3 Bagging

Consiste en combinar resultados de múltiples modelos para obtener un resultado generalizado. Debido a que se pueden obtener mismos resultados puesto que tienen la misma entrada, se crean subconjuntos de observaciones a partir del

conjunto de datos original, con reemplazo. Dichos subconjuntos pueden tener un tamaño menor que el conjunto original.

Es un método que se basa en el voto mayoritario o el promedio según el caso.

Este método reduce el error al combinar varios clasificadores con alta varianza cómo, por ejemplo, los árboles de clasificación, y se han mostrado en varios trabajos las mejoras obtenidas en las performances en problemas de clasificación y regresión. El fundamento teórico se basa en que el error promedio que se obtiene sobre la muestra de entrenamiento es mayor o igual al error obtenido por el estimador Bagging.

Se trata de un algoritmo muy sencillo de utilizar y fácil de adaptar a cualquier método de aprendizaje con que se trabaja.

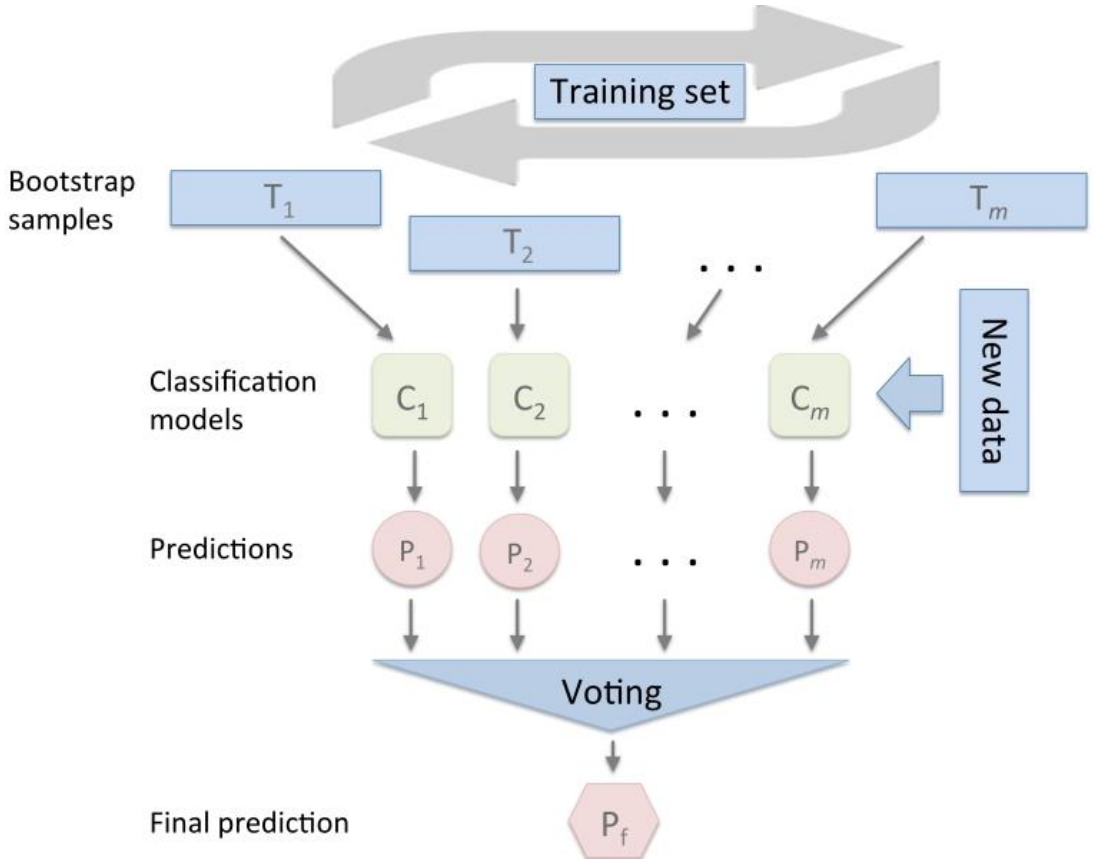


Figura 4.6 Ensemble Bagging.

5. Descripción del algoritmo a desarrollar

En este apartado, se presentarán tanto de forma teórica como detallada el funcionamiento del algoritmo de ensemble Bagging, que luego se va a seguir, mediante un enfoque distribuido, para realizar la propuesta de este proyecto.

Este modelo es usado para la tarea de clasificación, aunque también puede ser usado para regresión, pero su tarea nativa es la primera mencionada.

Cuando entrenamos un modelo, no importa realmente si estamos realizando una clasificación o una regresión, lo que tenemos que darnos cuenta es que debemos dar una entrada y obtendremos una correspondiente salida, y que ésta gira en torno al conjunto de datos de entrenamiento y test que hayamos ofrecido al método.

Obviamente los resultados están sujetos a variabilidad puesto que, si se hubieran observado otro conjunto de datos diferente, se habría obtenido un modelo distinto y por consiguiente resultados distintos.

El enfoque de este método es simple, a través de varios modelos independientes, queremos promediar sus predicciones y obtener con esto un modelo final, el cual tendrá una varianza mucho más baja. Para obtener las predicciones de los modelos independientes no podremos pasar conjuntos de datos distintos ya que entonces requerirá de muchos datos, por esa razón a través del conjunto de datos de entrenamiento y un porcentaje obtendremos **N** subconjuntos de datos de entrenamiento que serán los utilizados en los correspondientes modelos de entrenamiento, y estos subconjuntos de datos serán diferentes entre unos y otros.

El pseudocódigo de la división del conjunto de datos en varios subconjuntos es el siguiente:

```
bootstrap(trainingData, porcentaje, nSubset): Array[Dataframe] {
        subDF = Array[Dataframe]
        rows = (porcentaje*trainingData.count)
        for i <- 0 to nSubset-1 {
                subDF = Random_rows(trainingData,rows)
        }
        return subDF
}
```

Una vez obtenidas las distintas muestras, que actuarán como otro conjunto de datos independiente extraído de la distribución verdadera, se deberán mandar a los distintos métodos de clasificación o regresión para que sean entrenados y obtengamos los modelos con sus correspondientes predicciones.

Todos estos modelos de predicciones se deberán unificar en un solo modelo, que será el correspondiente modelo Bagging, mediante técnicas como, por ejemplo, selección por mayor número de voto en caso de clasificación, o por media para regresión.

De tal manera que, al obtener nuestro modelo final, éste pueda ser evaluado frente a la clase verdadera de Test. Tras realizar la evaluación, los resultados no cambian la respuesta esperada, si no que reduce su varianza.

El pseudocódigo del funcionamiento del ensemble Bagging es el siguiente:

```
Bagging(mClasificadores,trainingData,testData,porcentaje) {
        subDF = boostrap(trainingData,mClasificadores,porcentaje)
        arrayDF = Array[Dataframe]
        for i <- 0 to mClasificadores-1 {
                arrayDF = método_clasificación()
        }
        RDDpredicciones = union_predicciones_métodos(arrayDF)
        if ("clasificacion")
                clasificacion = voting(RDDpredicciones)
                evaluación_modelo(clasificacion)
        else
                regresion = averaging(RDDpredicciones)
                evaluación_modelo(regresion)
}
```

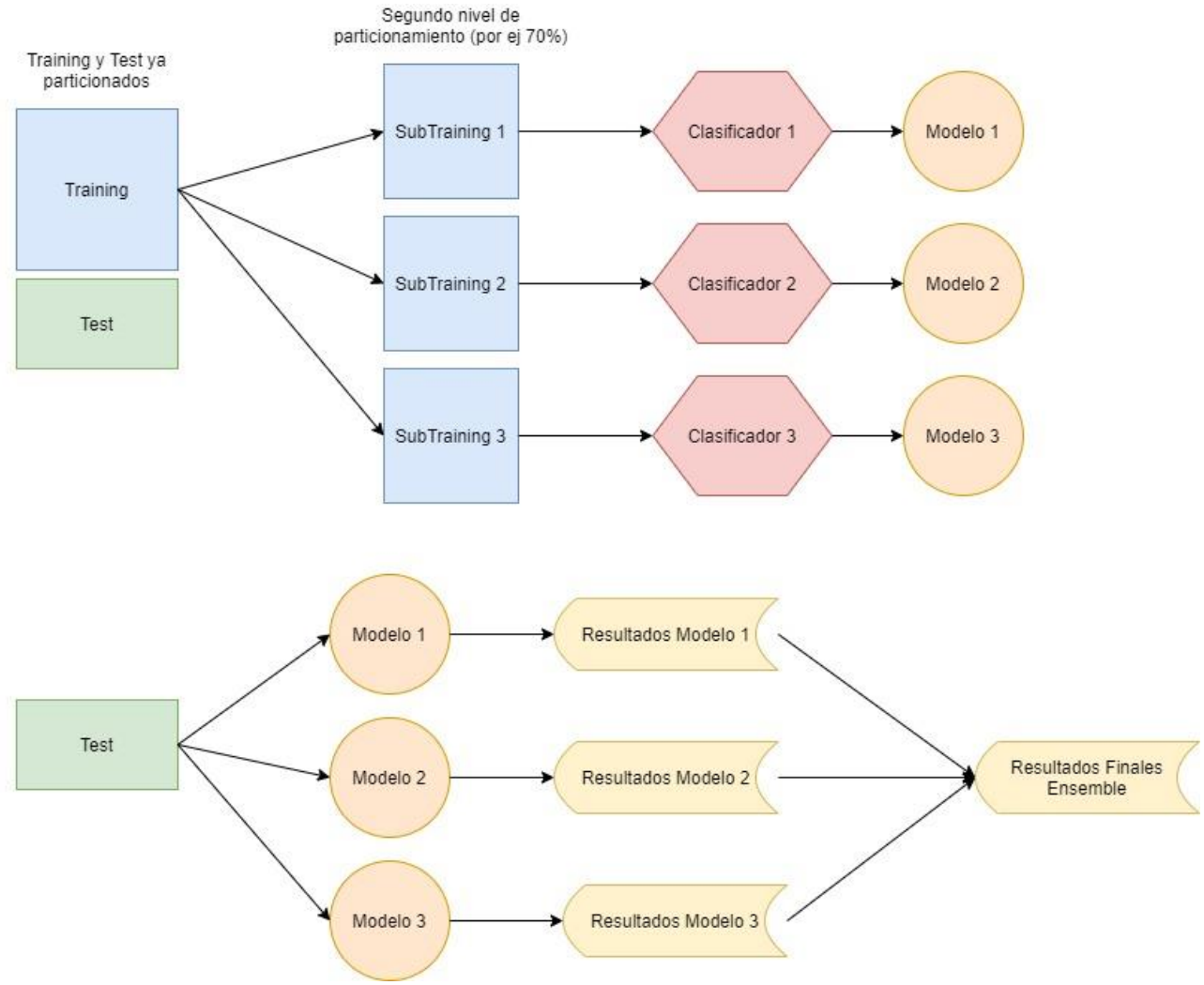


Figura 5.1 Esquema general del algoritmo Bagging. Fuente: Propia

Hay varias formas posibles de agregar los múltiples modelos ajustados en paralelo. Para un problema de clasificación, la clase producida por cada método, se puede ver como un voto, y la clase que obtenga el mayor número de votos se devolverá como la mejor clase predicha, y esto es conocido como votación dura. También se puede considerar las probabilidades de cada clase devuelta por todos los modelos, se promedian las probabilidades y se mantiene la clase que mayor probabilidad de promedio tiene, y estaríamos tratando con una votación suave.

Para los problemas de regresión, las salidas individuales se pueden promediar para obtener una salida del modelo final.

$$s_L(.) = \frac{1}{L}\sum_{l=1}^{L} w_l(.) \qquad \text{(simple average, for regression problem)}$$

$$s_L(.) = \arg\max_k [card(l|w_l(.) = k)] \qquad \text{(simple majority vote, for classification problem)}$$

Figura 5.2 Voto y media para clasificación y regresión. Fuente: Propia

6. DISEÑO E IMPLEMENTACIÓN

Este capítulo, presenta la fase de diseño que se ha realizado para llevar a cabo la implementación del algoritmo Bagging.

Una de las grandes ventajas de este ensemble, es que puede ser paralelizado, ya que los diferentes modelos se ajustan independientemente unos de otros.

Además de la fase de diseño, se exponen detalles de implementación que han influido para la realización de este Trabajo de Fin de Grado.

6.1 Diseño

Una vez se ha descrito de forma teórica el ensemble que se ha escogido para este trabajo, se pasa a la fase de diseño. Este método es desarrollado haciendo uso de la biblioteca de MLlib, por lo que es necesario realizar un estudio de cómo está estructurada la biblioteca para identificar qué métodos de clasificación o regresión son los que podemos utilizar para nuestros conjuntos de datos y cuáles queremos utilizar.

Los datasets que utilizaremos son los más voluminosos incluidos en el repositorio KEEL (Alcalá-Fdez, Sánchez, & otros, 2005).

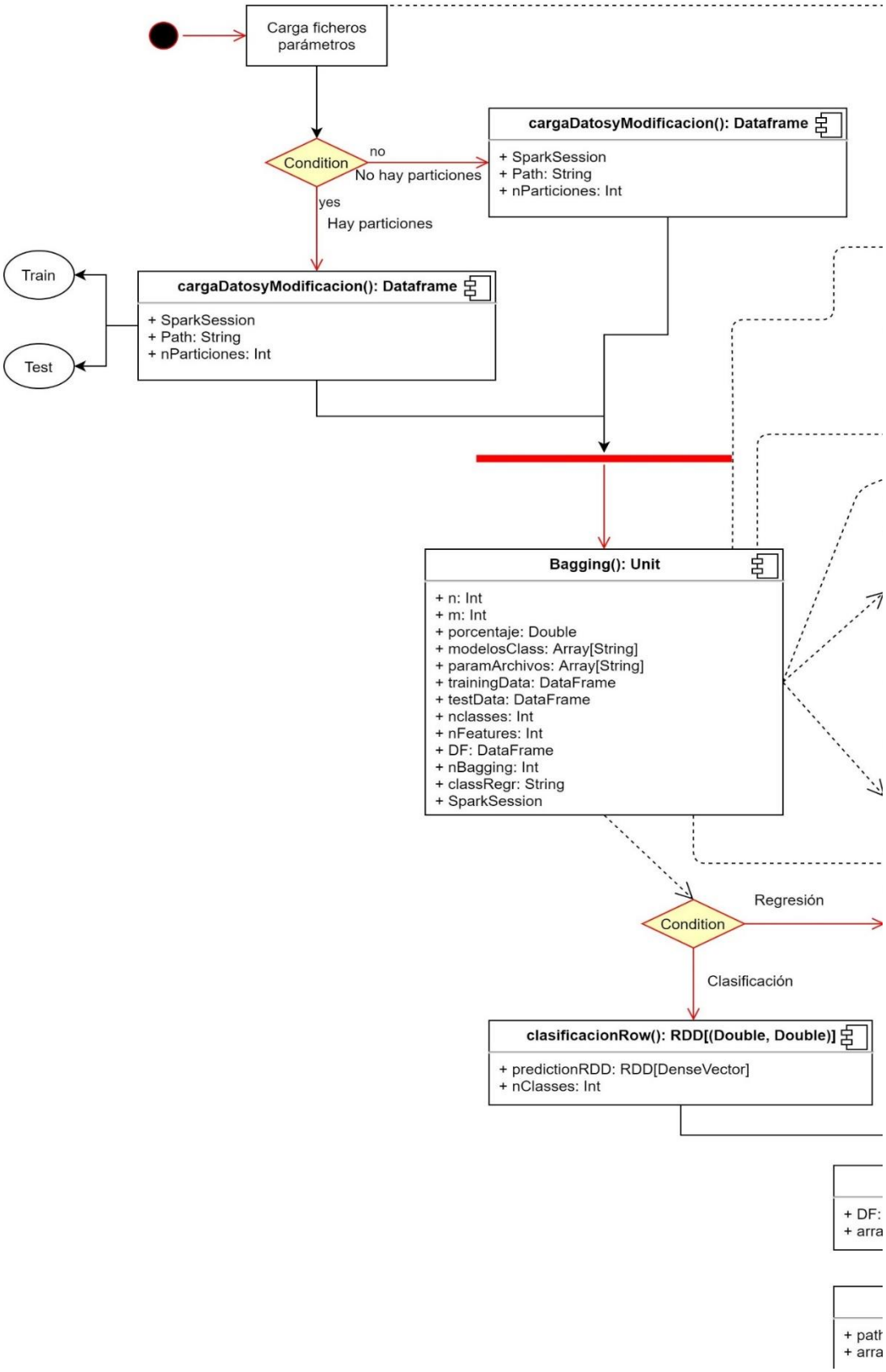
Spark está dividida en dos paquetes, spark.mllib y spark.ml. Nosotros nos centraremos en usar spark.ml que es la API que está siendo actualmente desarrollada.

Siguiendo con la organización de código de esta biblioteca, usaremos los métodos que aparecen el en el paquete *org.apache.spark.ml.classification* ya que aquí estarán los correspondientes métodos de clasificación de los cuales haremos uso. También *org.apache.spark.ml.regression* en el caso de querer realizar tareas de regresión.

Al comienzo del diseño, se decidió usar paquete ML, ya que en él había implementados más algoritmos tanto de clasificación como de regresión. Para este paquete el tipo de dato distribuido es el DataFrame y con él es con el que se

ha trabajado mayoritariamente. En algunas partes del código se han utilizado también RDDs para facilitar ciertas tareas.

A continuación, un UML del software desarrollado:



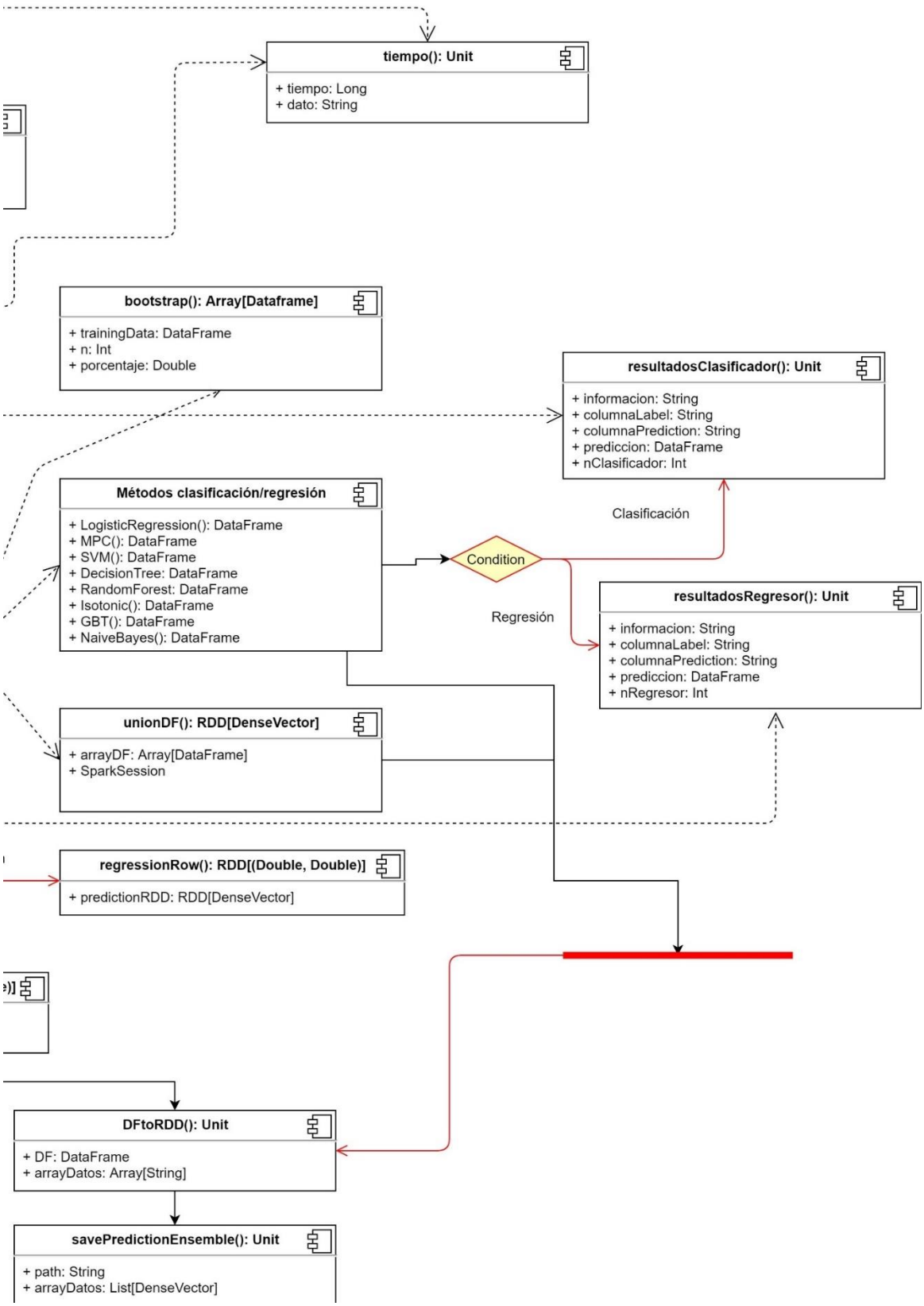


Figura 6.1 UML del proyecto desarrollado. Fuente: Propia

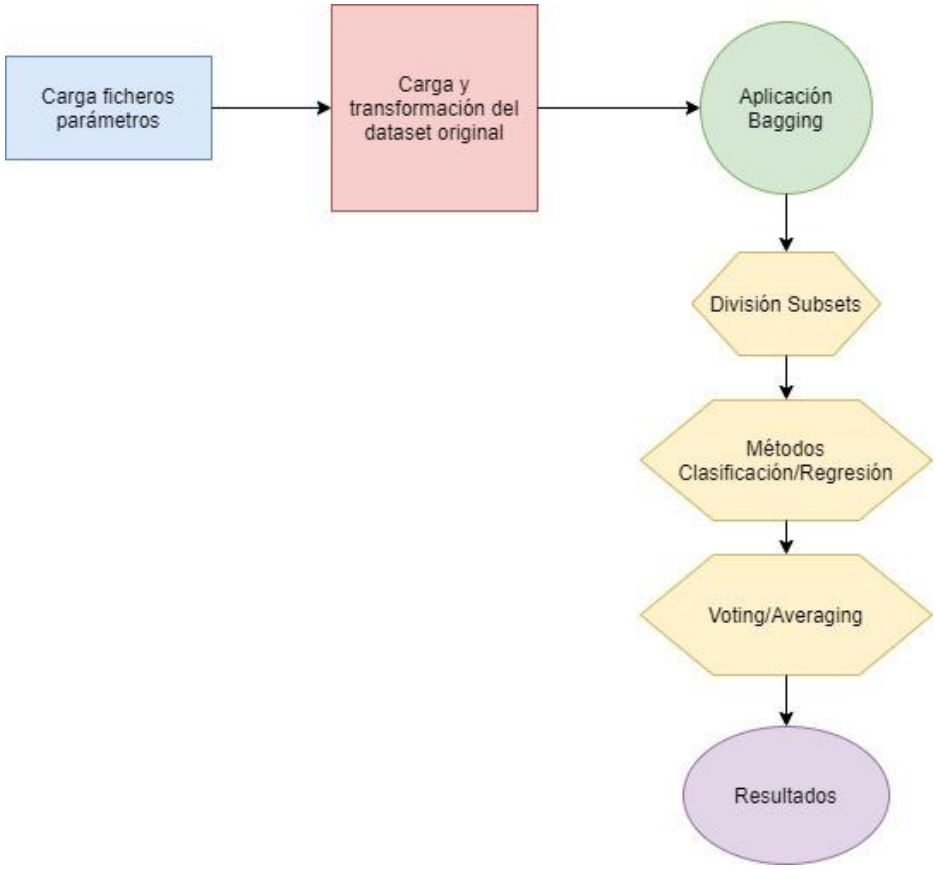


Figura 6.2 Diseño del programa. Fuente: Propia

6.2 Implementación

Aunque Apache Spark proporciona APIs para Scala, Java, Python y R, este framework está desarrollado en Scala y es por esto que esta es la API más pura, flexible y la primera en actualizarse.

Scala es un lenguaje de programación multi-paradigma diseñado por Martin Odersky que combina el paradigma funcional y la orientación de objetos (Odersky, y otros, 2004). La implementación por defecto corre sobre la JVM, aunque ya existen proyectos para su ejecución en el sistema operativo de forma nativa, además es compatible con el código Java existente.

Una de las características de Scala es que tiene un sistema de tipos muy rico. Esto significa que la comprobación de tipos se realiza durante la compilación, y

no durante la ejecución, pudiendo así evitar muchos errores en tiempo de compilación.

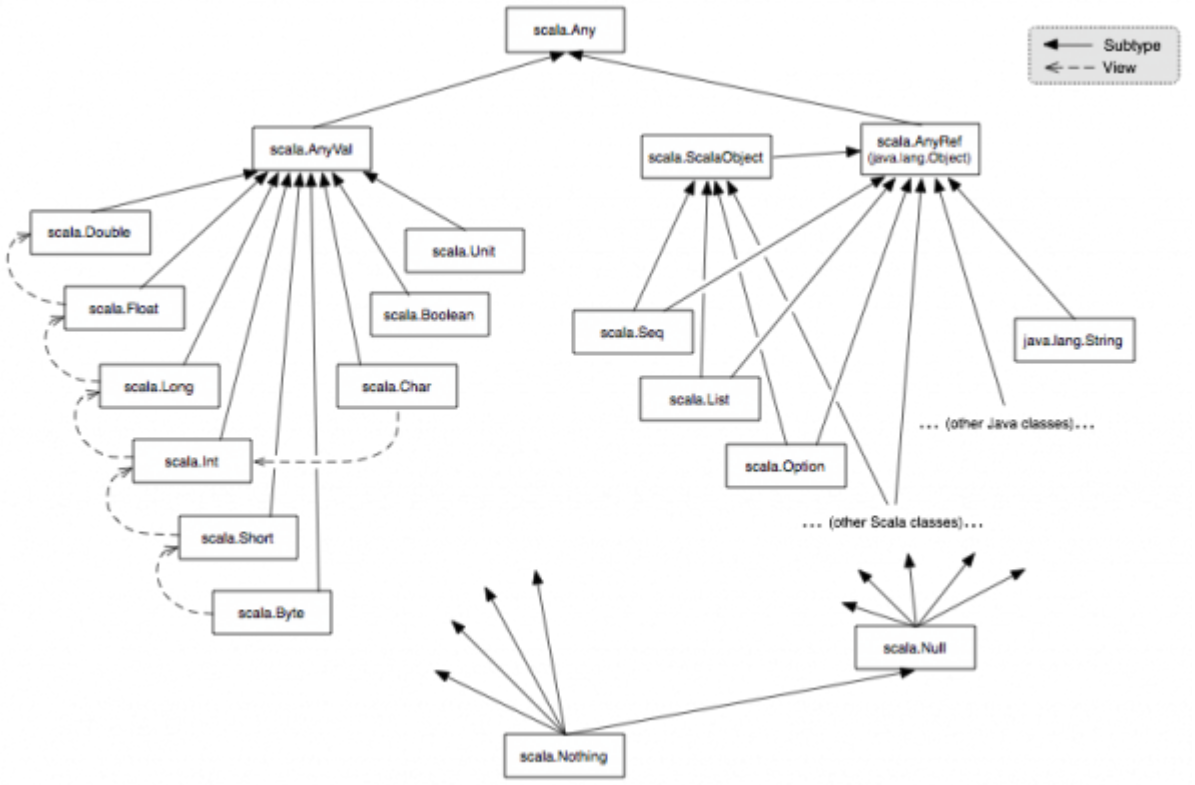


Figura 6.3 Sistema de tipos de Scala. Fuente: (Odersky, y otros, 2004)

A la hora de implementar el código del ensemble Bagging distribuido, se han tenido que abordar diferentes etapas. Primeramente, tuvimos que tratar de manera cuidadosa y eficiente el dataset que se le pasaba, ya que todos los datasets no mantienen la misma estructura, ni tienen el mismo tipo de datos.

Por esa misma razón primero debíamos tratar los datasets internamente para que todos mantuvieran la misma forma, tal que contuviesen únicamente los datos, quitándoles las cabeceras informativas en las que suele aparecer los atributos que se usan con su tipo de datos, clases, etc.

Una vez realizado este proceso, siguiendo el esquema de la figura 6.2, pasaríamos a cargar el dataset y se emplearían sus correspondientes transformaciones. Por ejemplo, las características se pondrían dentro de un rango, para así lidiar con valores negativos que nos darían problemas al emplear

los métodos de clasificación o regresión, y también en caso de que en un dataset la clase sea de tipo nominal, ésta se transformara a tipo numérico. Esto último también se aplicaría al caso de las características nominales.

Y por último se realiza una transformación para agrupar en un vector las características de cada ejemplo para todo el conjunto de datos, de forma que las instancias tengan el formato adecuado para poder entrenar los algoritmos de MLlib. Quedándose finalmente el formato del conjunto de datos como una columna que contiene los **vectores de características**, y otra columna de **label**.

Cuando ya tenemos el conjunto de datos transformado, estará listo para que se le pueda aplicar cualquier tipo de método de clasificación o regresión, dichos métodos se aplicarán ya directamente dentro del método Bagging.

Como bien se ha explicado en el apartado de ensemble Bagging, el conjunto de datos de entrenamiento se particionará en tantos subconjuntos de datos como clasificadores queramos ejecutar y posteriormente se aplicarán los métodos de clasificación o regresión, obteniendo en una estructura de Array todos los DataFrame de predicciones devueltos por los métodos, y se unirán para dejarlo en un único DataFrame, con sus correspondientes columnas de predicciones y la columna con la clase real de Test.

Este DataFrame se transformará en RDD para poder realizar de una forma más cómoda la clasificación por voto o regresión por media, ya que con este tipo de estructura y con los métodos que contiene es muy fácil el acceso a los datos y en nuestro caso su tratamiento para la obtención de los votos o medias.

Tras finalizar uno de los dos procedimientos, obtendremos una estructura de datos que será un RDD de pares de dobles, independientemente si hemos escogido clasificación o regresión.

Y finalmente, al RDD devuelto que contendrá el modelo final Bagging se le aplicará un método para transformarlo de nuevo a DataFrame para que se pueda pasar a los métodos de evaluación correspondientes donde se compararán los valores predichos frente a los reales para el dataset de Test, y con esto poder obtener datos posteriores como precisión, error, etc.

6.3 Subida de proyecto a Spark-packages

Una vez terminado el proyecto, hemos decidido publicarlo en la página de paquetes de Spark.

Para ello previamente deberemos subirlo a Github, donde deberemos añadir un archivo con las licencias de Apache Spark, y un archivo README.md donde habrá una descripción de nuestro proyecto.

Cuando tengamos completado eso, y esté público en la plataforma, iremos a https://spark-packages.org/register donde vincularemos nuestro GitHub, y podremos seleccionar el repositorio previamente subido, añadiendo una descripción, y mandaremos una petición para que nos lo publiquen.

Es una plataforma en el que hay muchísimos paquetes publicados por diferentes tipos de personas, donde poder dar a conocer algo, además de que enlazan con tu GitHub y pueden acceder a tu perfil rápidamente y ver tus trabajos, así como usarlos.

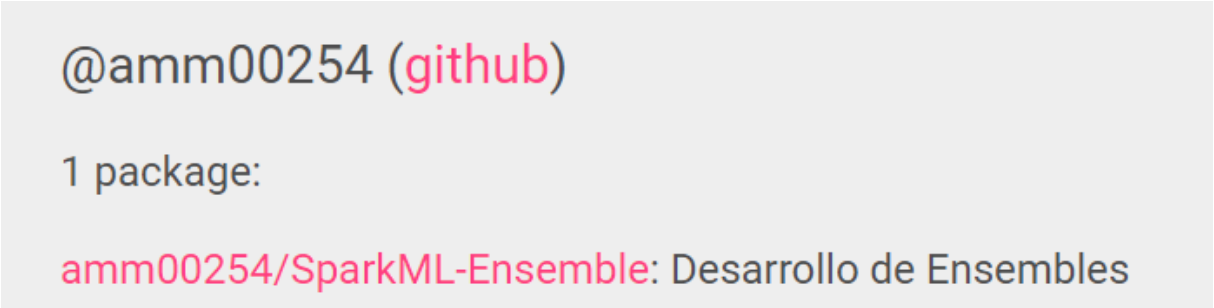


Figura 6.4 Proyecto publicado.

En referencia a la figura 6.4, se puede ver con nuestro proyecto de fin de grado finalmente está publicado en la plataforma, y cualquier persona que acceda puede utilizarlo.

Para acceder al proyecto únicamente se debe acceder a través del enlace: https://spark-packages.org/package/amm00254/SparkML-Ensemble

7. EXPERIMENTACIÓN Y ANÁLISIS DE LOS RESULTADOS

En este apartado se expondrán los resultados que se han obtenido con el algoritmo desarrollado y que realiza la función del ensemble Bagging, tanto para conjuntos de datos con clases binarias, multiclase o reales.

Se puede aplicar tanto para tareas de clasificación como de regresión, empleando los distintos métodos de MLlib, por lo que podremos discutir la eficiencia y eficacia de cada método e incluso la del modelo Bagging.

También se destacará como se reduce el tiempo al emplear diferente número de particiones internas de Spark.

Se escogerán conjuntos de datos de distintos tamaños y características, teniendo en cuenta la dificultad añadida de tener que ejecutar en un clúster en el que varios compañeros pueden realizar también sus experimentaciones.

Con el objetivo de que los resultados obtenidos sean fiables, las pruebas se realizarán sobre conjuntos de datos ya particionados, utilizando *5-folds cross validation* (5 validaciones cruzadas).

7.1 Características de la experimentación

Para la siguiente experimentación se hace uso de KEEL como herramienta clásica de minería de datos, de donde se escogerán distintos tipos de conjuntos de datos que se usarán para probar el ensemble Bagging desarrollado. Con este objetivo, las características de los conjuntos de datos obtenidos del repositorio KEEL son los siguientes.

		Atributos				
Dataset	Nº Ejemplos	Reales	Enteros	Nominales	Nº Clases	Tipo
Poker	1.025.010	0	10	0	10	Clasificación
Census	142.521	1	12	28	3	Clasificación
Fars	100.968	5	0	24	8	Clasificación
Connect4	67.557	0	0	42	3	Clasificación
Shuttle	58.000	0	9	0	7	Clasificación
Mv	40.768	7	3	0	-	Regresión

Tabla 7.1 Características de los diferentes conjuntos de datos.

Fuente: Propia

Como se puede apreciar en la tabla 7.1, el dataset con el número de instancias mayor es Poker, siendo éste también el más grande del repositorio KEEL.

A continuación, se hace una descripción semántica de los distintos datasets:

- **Poker:** Cada ejemplo de este conjunto de datos representa una mano de póker que consiste en cinco cartas de la baraja. Cada carta se describe usando dos atributos (palo y rango), lo que suma un total de diez atributos. El atributo clase de este conjunto de datos representa el estado de la partida.

- **Census:** El conjunto de datos de census fueron extraídos en 1994 de Estados Unidos. Contiene atributos continuos y nominales, que describen ciera información social (edad, raza, sexo, estado civil, …) sobre los ciudadanos registrados. La tarea es predecir si los ingresos del ciudadano exceden los cincuenta mil dólares anuales.

- **Fars:** El conjunto de datos Fars (Fatality Analysis Reporting System), es una recopilación de estadísticas sobre accidentes automovilísticos realizadas por el Centro Nacional de Estadísticas y Anásisis de EEUU. El conjunto de datos específico utilizado contiene información sobre todas las personas involucradas en accidentes automovilísticos en EEUU durante 2001, donde la mayoría de sus atributos están representados con valores nominales. El atributo de clase describe el nivel de lesión sufrida.

- **Connect-4:** Esta base de datos contiene posiciones del juego conecta-4 con una malla de tamaño 6x7, en ninguna instancia el juego está terminado, por lo que la tarea a la hora de clasificar es predecir que jugador ganará la partida o si habrá empate.

- **Shuttle:** Éste conjunto de datos tiene su origen en la NASA, originalmente se usó para extraer reglas que determinasen en qué condiciones era preferible que una nave aterrizase automáticamente a ser aterrizada manualmente. Tiene siete clases, perteneciendo la primera en el 80% de las instancias por lo que se debería obtener una precisión muy alta. Consta de nueve atributos, todos ellos numéricos, y un total de 58.000 instancias.

- **Mv:** Conjunto de datos artificiales con dependencias entre los valores de los atributos. Los casos son generados usando diferentes métodos.

Como se ha mencionado anteriormente, los datasets utilizados utilizarán validación cruzada *k-fold*, permitiéndonos así garantizar que todos los resultados son independientes de la partición de entrenamiento y test que se escojan.

Se han escogido 5 particiones para la ejecución de los diferentes conjuntos de datos en la experimentación, utilizando éstas del repositorio de Keel.

También se han tenido en cuenta a la hora de la ejecución, las diferentes particiones internas de Spark, permitiéndonos esto una buena paralelización de los datos y logrando una reducción del tiempo empleado. Se han utilizado particiones internas de 1, 2, 4 y 8.

La experimentación se ha realizado en un clúster del CEATIC (Centro de Estudios Avanzados en Tecnologías de la Información y Comunicación) a través de nodos Big Data.

Las características del hardware del clúster son las siguientes:

- Plataforma: R424F3
- Número de nodos: 16
- Procesador: 2x Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
- Memoria RAM: 64 GB
- Almacenamiento: 6x 500 GB

7.2 Resultados de cada dataset

Para poder comparar los resultados obtenidos, se realizarán las medias de cada método de clasificación/regresión realizado, consiguiendo así la media de los distintos métodos empleados, junto al resultado global del ensemble Bagging, pudiendo así comparar su precisión respecto a la realización por separado de cada método.

Como la mayoría de los experimentos se han realizado para clasificación, obtendremos resultados que contienen la *Accuracy* (Precisión) de los modelos base y del modelo final Bagging.

Los valores obtenidos de *Accuracy* se calculan como la proporción entre las predicciones correctas que ha hecho el modelo y el total de predicciones.

Con respecto a los resultados del experimento de regresión, más adelante se explicarán cómo se calculan cada uno de ellos y a que hacen referencia.

7.2.1 Tablas de resultados del dataset 'Poker'

Se han ejecutado las 5 particiones del conjunto de datos *Poker*, y se han obtenido los siguientes resultados de *Accuracy* tanto de los métodos de clasificación como el del modelo Bagging:

	MPC	Decision Tree	Random Forest	Resultado Bagging
Partición 1	0,544	0,528	0,535	0,539
Partición 2	0,545	0,533	0,540	0,546
Partición 3	0,544	0,540	0,538	0,551
Partición 4	0,543	0,536	0,534	0,542
Partición 5	0,545	0,546	0,546	0,549
Media de métodos	0,544	0,536	0,538	
Media de medias	0,540			
Media Bagging	0,545			

Tabla 7.2 Resultados de Accuracy para dataset Poker.
Fuente: Propia

Como se puede ver en la tabla 7.2, los resultados de los métodos individuales de clasificación y del modelo Bagging son similares, siendo este último un poco superior.

Se trata de una media de precisión del 54,0% de los diversos clasificadores empleados, con respecto al 54,5% que se ha conseguido con el modelo Bagging, por lo que hemos conseguido mejorar la precisión del dataset, en su mayor medida, habiendo usado una gran variedad de clasificadores los cuáles trabajan internamente de forma distinta.

Con esto podemos decir que la eficiencia y eficacia del ensemble ha sido buena.

		Carga datos	División subdatasets	Clasificadores (Train + Test)	Unión predicciones	Voto
Partición interna 1	Partición 1	0:00:47	0:00:13	0:39:19	0:00:18	0:00:00
	Partición 2	0:00:16	0:00:10	0:38:32	0:00:09	0:00:00
	Partición 3	0:00:07	0:00:06	0:35:11	0:00:09	0:00:00
	Partición 4	0:00:10	0:00:06	0:35:10	0:00:09	0:00:00
	Partición 5	0:00:06	0:00:07	0:33:35	0:00:09	0:00:00
	Suma Tiempos	0:01:26	0:00:42	3:01:47	0:00:54	0:00:00
	Tiempo total	3:04:49				
Partición interna 2	Partición 1	0:00:48	0:00:08	0:36:34	0:00:15	0:00:00
	Partición 2	0:00:09	0:00:04	0:37:12	0:00:13	0:00:00
	Partición 3	0:00:09	0:00:05	0:33:49	0:00:10	0:00:00
	Partición 4	0:00:09	0:00:05	0:33:18	0:00:09	0:00:00
	Partición 5	0:00:06	0:00:08	0:32:11	0:00:09	0:00:00
	Suma Tiempos	0:01:21	0:00:30	2:53:04	0:00:56	0:00:00
	Tiempo total	2:55:51				
Partición interna 4	Partición 1	0:00:46	0:00:11	0:31:45	0:00:17	0:00:00
	Partición 2	0:00:07	0:00:03	0:30:09	0:00:08	0:00:00
	Partición 3	0:00:07	0:00:02	0:30:08	0:00:10	0:00:00
	Partición 4	0:00:05	0:00:03	0:30:19	0:00:08	0:00:00
	Partición 5	0:00:07	0:00:03	0:30:40	0:00:10	0:00:00
	Suma Tiempos	0:01:12	0:00:22	2:33:01	0:00:53	0:00:00
	Tiempo total	2:35:28				
Partición interna 8	Partición 1	0:00:45	0:00:04	0:29:31	0:00:08	0:00:00
	Partición 2	0:00:05	0:00:03	0:29:55	0:00:09	0:00:00
	Partición 3	0:00:06	0:00:03	0:30:04	0:00:09	0:00:00
	Partición 4	0:00:05	0:00:04	0:30:04	0:00:09	0:00:00
	Partición 5	0:00:05	0:00:02	0:30:04	0:00:09	0:00:00
	Suma Tiempos	0:01:06	0:00:16	2:29:38	0:00:44	0:00:00
	Tiempo total	2:31:44				

Tabla 7.3 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Poker.
Fuente: Propia

Independientemente de la partición del dataset, podemos apreciar en la tabla 7.3 que según aumentamos el número de particiones internas de Spark, se va reduciendo el tiempo.

Conforme mayor es el número de instancias del conjunto de datos, más apreciable será la diferencia de tiempos.

Cabe destacar como se puede ver como entre la partición interna 1,2 y 4 hay una diferencia de tiempo más notoria, en cambio entre la 4 y 8 la diferencia de tiempo no es tan grande como en las anteriores.

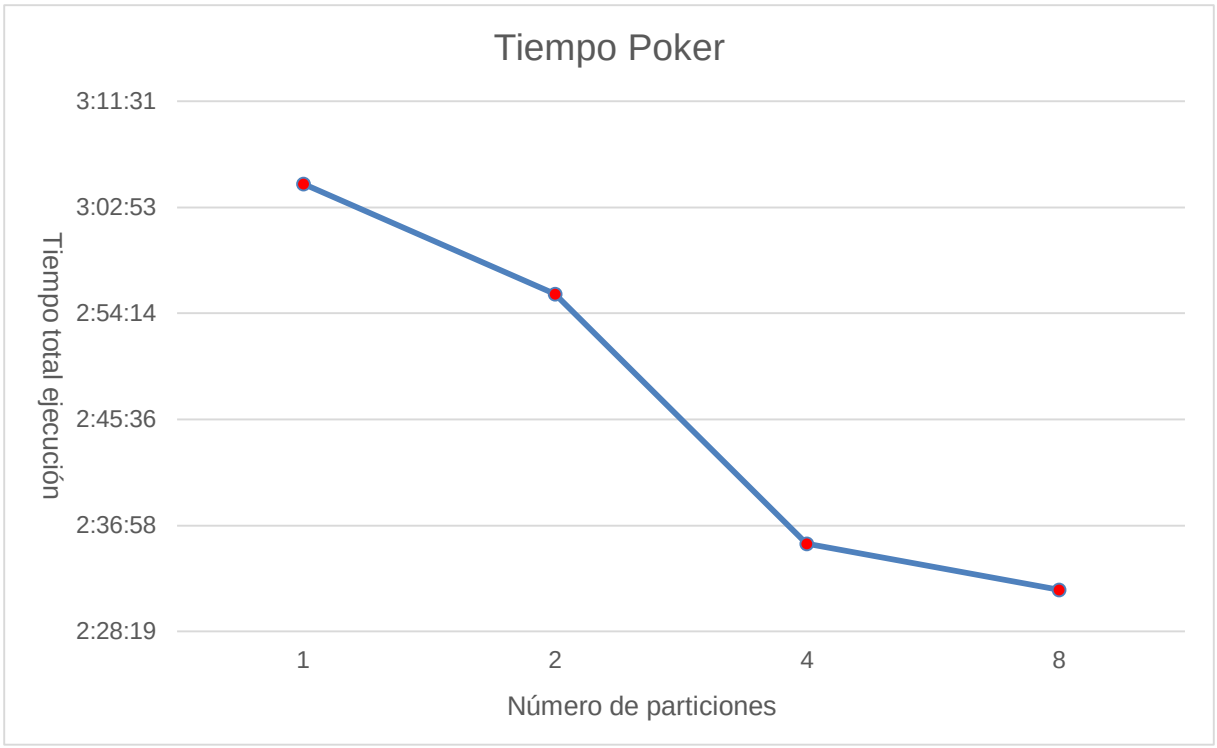


Figura 7.1 Comparación de tiempos con diferentes particiones internas para dataset Poker.
Fuente: Propia

Como mencionamos previamente, se puede comprobar como a medida que aumenta el número de particiones internas, se va reduciendo el tiempo.

Con conjuntos de datos con instancias grandes se nota mucho la diferencia de reducción de tiempo. Aunque se puede comprobar que en ciertas ocasiones un número elevado de particiones internas no asegura una disminución de tiempo. En la figura 7.1, se puede apreciar que la diferencia de tiempos entre 4 y 8 particiones es mínima, a diferencia de particiones como entre 1, 2 y 4, que es más significativo. Esto se debe a que el volumen de datos manejado en cada partición no tiene el suficiente tamaño como para que la paralelización sea rentable, dado que el tiempo de comunicación entre los nodos de ejecución es superior o cercano al tiempo de procesamiento.

7.2.2 Tablas de resultados del dataset 'Census'

Se han ejecutado las 5 particiones del conjunto de datos *Census*, y se han obtenido los siguientes resultados de *Accuracy* tanto de los métodos de clasificación como el del modelo Bagging:

	Naive Bayes	Logistic Regression	MPC	Decision Tree	Random Forest	Resultado Bagging
Partición 1	0,939	0,938	0,948	0,944	0,941	0,940
Partición 2	0,939	0,938	0,949	0,944	0,942	0,941
Partición 3	0,939	0,938	0,948	0,944	0,941	0,941
Partición 4	0,939	0,938	0,949	0,944	0,942	0,941
Partición 5	0,939	0,938	0,948	0,943	0,942	0,941
Media de métodos	0,939	0,938	0,948	0,944	0,941	
Media de medias	0,942					
Media Bagging	0,941					

Tabla 7.4 Resultados de Accuracy para dataset Census.
Fuente: Propia

Como se puede comprobar en la tabla 7.4, los resultados individuales de clasificación y del modelo Bagging son similares.

Estamos hablando de una media de precisión del 94,2% de los diversos clasificadores empleados, con respecto al 94,1% que se ha conseguido con el modelo Bagging, por lo que hemos conseguido conservar la precisión.

En este caso no se ha conseguido mejorar la precisión, pero sí que se ha logrado equilibrar obteniendo finalmente una precisión casi idéntica a la media de todos los clasificadores utilizados, teniendo en cuenta que cada uno funciona internamente de una manera distinta.

Se puede decir que el ensemble se ha comportado correctamente

		Carga datos	División subdatasets	Clasificadores (Train + Test)	Unión predicciones	Voto
Partición interna 1	Partición 1	0:02:03	0:00:14	0:25:55	0:00:42	0:00:00
	Partición 2	0:01:27	0:00:15	0:24:49	0:00:36	0:00:00
	Partición 3	0:01:19	0:00:15	0:24:55	0:00:41	0:00:00
	Partición 4	0:01:23	0:00:14	0:24:26	0:00:37	0:00:00
	Partición 5	0:01:20	0:00:11	0:23:35	0:00:41	0:00:00
	Suma Tiempos	0:07:32	0:01:09	2:03:40	0:03:17	0:00:00
	Tiempo total	2:15:38				
Partición interna 2	Partición 1	0:01:57	0:00:11	0:19:39	0:00:39	0:00:00
	Partición 2	0:01:19	0:00:12	0:19:34	0:00:45	0:00:00
	Partición 3	0:01:19	0:00:10	0:18:55	0:00:36	0:00:00
	Partición 4	0:01:20	0:00:10	0:19:20	0:00:35	0:00:00
	Partición 5	0:01:23	0:00:09	0:19:04	0:00:47	0:00:00
	Suma Tiempos	0:07:18	0:00:52	1:36:32	0:03:22	0:00:00
	Tiempo total	1:48:04				
Partición interna 4	Partición 1	0:01:57	0:00:12	0:17:42	0:00:48	0:00:00
	Partición 2	0:01:13	0:00:12	0:17:33	0:00:44	0:00:00
	Partición 3	0:01:13	0:00:12	0:17:59	0:00:40	0:00:00
	Partición 4	0:01:13	0:00:12	0:17:41	0:00:48	0:00:00
	Partición 5	0:01:12	0:00:12	0:17:43	0:00:42	0:00:00
	Suma Tiempos	0:06:48	0:01:00	1:28:38	0:03:42	0:00:00
	Tiempo total	1:40:08				
Partición interna 8	Partición 1	0:01:51	0:00:09	0:16:59	0:00:44	0:00:00
	Partición 2	0:01:09	0:00:11	0:16:50	0:00:39	0:00:00
	Partición 3	0:01:08	0:00:11	0:16:54	0:00:43	0:00:00
	Partición 4	0:01:09	0:00:10	0:16:49	0:00:40	0:00:00
	Partición 5	0:01:09	0:00:10	0:16:43	0:00:38	0:00:00
	Suma Tiempos	0:06:26	0:00:51	1:24:15	0:03:24	0:00:00
	Tiempo total	1:34:56				

Tabla 7.5 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Census.
Fuente: Propia

Al igual que en el ejemplo del dataset *Poker*, al tratarse de un conjunto de datos con un número de instancias elevado, la reducción de tiempo al usar diferente número de particiones internas va a ser notoria.

Como se puede apreciar en la tabla 7.5, a medida que aumentamos el número de particiones internas de Spark, se va reduciendo el tiempo, siendo mucho más significativo con 1, 2 y 4 particiones internas.

Al principio al aumentar el número de particiones, se aprecia muy bien la reducción de tiempo, pero comparando la partición 4 con la 8, se puede ver como ya no es tan significativa la disminución de tiempo a comparación con la partición 1 y 2, o 2 y 4.

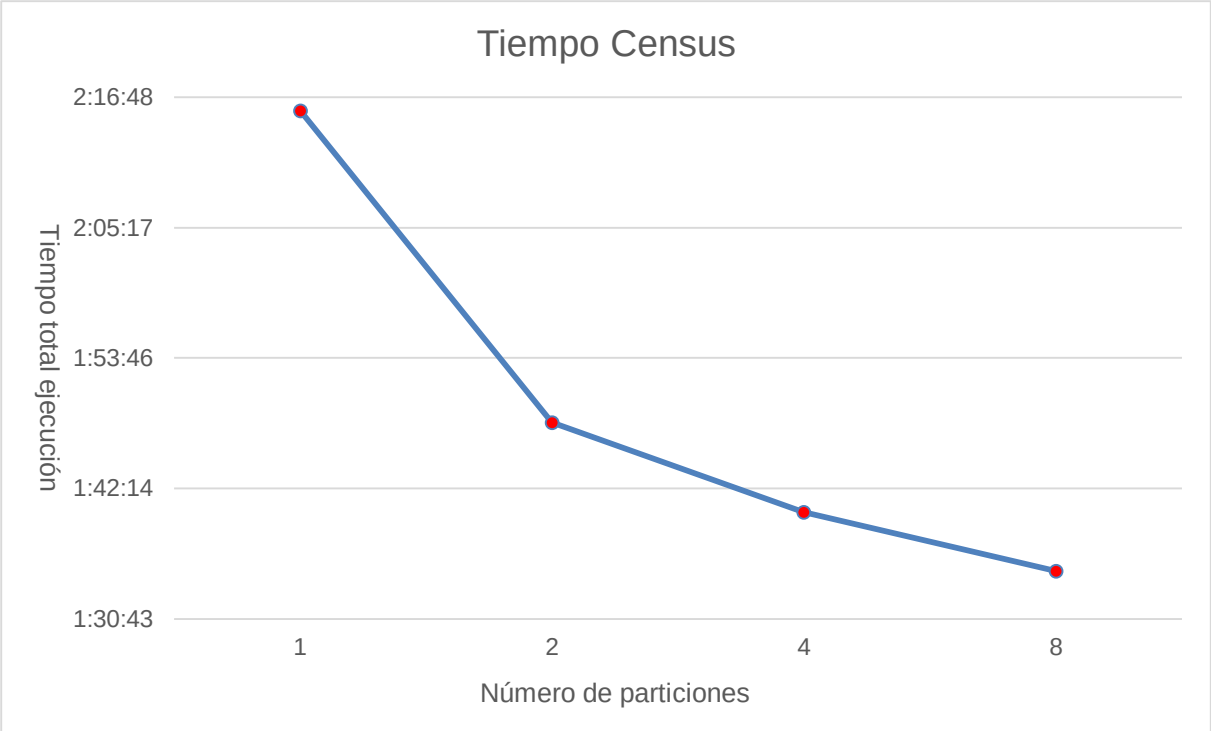


Figura 7.2 Comparación de tiempos con diferentes particiones internas para dataset Census.
Fuente: Propia

Como se puede apreciar en la figura 7.2, la diferencia de tiempos entre 4 y 8 particiones no es tan grande como la diferencia entre las particiones internas 1, 2 y 4, que es más significativo.

7.2.3 Tablas de resultados del dataset 'Fars

Se han ejecutado las 5 particiones del conjunto de datos *Fars*, y se han obtenido los siguientes resultados de *Accuracy* tanto de los métodos de clasificación como el del modelo Bagging:

	Naive Bayes	MPC	Decision Tree	Random Forest	Resultado Bagging
Partición 1	0,606	0,781	0,777	0,778	0,784
Partición 2	0,601	0,781	0,779	0,779	0,782
Partición 3	0,606	0,785	0,784	0,782	0,790
Partición 4	0,607	0,784	0,779	0,779	0,787
Partición 5	0,609	0,783	0,778	0,783	0,786
Media de métodos	0,606	0,783	0,779	0,780	
Media de medias	0,737				
Media Bagging	0,786				

Tabla 7.6 Resultados de Accuracy para dataset Fars.
Fuente: Propia

Observando la tabla 7.6, se puede ver que hay más diferencias entre los resultados de los métodos de clasificación y los del modelo Bagging, consiguiendo el ensemble Bagging mejores resultados de clasificación.

Estamos hablando de una media de precisión del 73,7% de los diversos clasificadores empleados, con respecto al 78,6% que se ha conseguido con el modelo Bagging, por lo que la precisión se ha mejorado bastante, siendo la diferencia de un 4,9%.

Se puede decir que el ensemble Bagging ha conseguido mejorar la precisión frente a la cantidad de métodos de clasificación empleados, por lo que podemos decir que el ensemble ha funcionado muy bien.

		Carga datos	División subdatasets	Clasificadores (Train + Test)	Unión predicciones	Voto
Partición interna 1	Partición 1	0:01:28	0:00:07	0:10:51	0:00:20	0:00:01
	Partición 2	0:00:38	0:00:07	0:10:02	0:00:18	0:00:00
	Partición 3	0:00:32	0:00:06	0:09:52	0:00:19	0:00:00
	Partición 4	0:00:34	0:00:07	0:10:03	0:00:17	0:00:00
	Partición 5	0:00:34	0:00:07	0:09:54	0:00:19	0:00:00
	Suma Tiempos	0:03:46	0:00:34	0:50:42	0:01:33	0:00:01
	Tiempo total	0:56:36				
Partición interna 2	Partición 1	0:01:30	0:00:06	0:08:55	0:00:16	0:00:00
	Partición 2	0:00:38	0:00:05	0:08:17	0:00:14	0:00:00
	Partición 3	0:00:32	0:00:04	0:08:24	0:00:13	0:00:00
	Partición 4	0:00:33	0:00:05	0:08:36	0:00:15	0:00:00
	Partición 5	0:00:35	0:00:08	0:08:26	0:00:13	0:00:00
	Suma Tiempos	0:03:48	0:00:28	0:42:38	0:01:11	0:00:00
	Tiempo total	0:48:05				
Partición interna 4	Partición 1	0:01:22	0:00:06	0:08:20	0:00:22	0:00:00
	Partición 2	0:00:32	0:00:05	0:08:09	0:00:20	0:00:00
	Partición 3	0:00:32	0:00:06	0:08:13	0:00:20	0:00:00
	Partición 4	0:00:32	0:00:06	0:08:14	0:00:20	0:00:00
	Partición 5	0:00:32	0:00:06	0:08:17	0:00:21	0:00:00
	Suma Tiempos	0:03:30	0:00:29	0:41:13	0:01:43	0:00:00
	Tiempo total	0:46:55				
Partición interna 8	Partición 1	0:01:17	0:00:05	0:08:13	0:00:18	0:00:00
	Partición 2	0:00:29	0:00:05	0:07:49	0:00:16	0:00:00
	Partición 3	0:00:29	0:00:05	0:07:58	0:00:17	0:00:00
	Partición 4	0:00:29	0:00:04	0:07:56	0:00:16	0:00:00
	Partición 5	0:00:29	0:00:05	0:08:03	0:00:18	0:00:00
	Suma Tiempos	0:03:13	0:00:24	0:39:59	0:01:25	0:00:00
	Tiempo total	0:45:01				

Tabla 7.7 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Fars.
Fuente: Propia

Como se puede visualizar en la tabla 7.7, la diferencia de tiempos, no son tan significantes como en los conjuntos de datos anteriormente mencionados.

Esto se debe a que el dataset utilizado, no es lo suficientemente grande para seguir aumentando el número de particiones en el clúster, ya que el coste de aumentar el paralelismo es mayor que beneficio que se obtiene en rendimiento. Con un conjunto de datos mucho más grande no se daría esta situación por lo que se podrían añadir muchas más particiones.

Las particiones donde son algo más significativos los tiempos es entre la partición 1 y 2, siendo aquí donde la disminución de tiempo oscila alrededor de los 8 minutos, a diferencia de las particiones entre 2 y 4, o 4 y 8, donde la diferencia de tiempo ronda los 2 minutos.

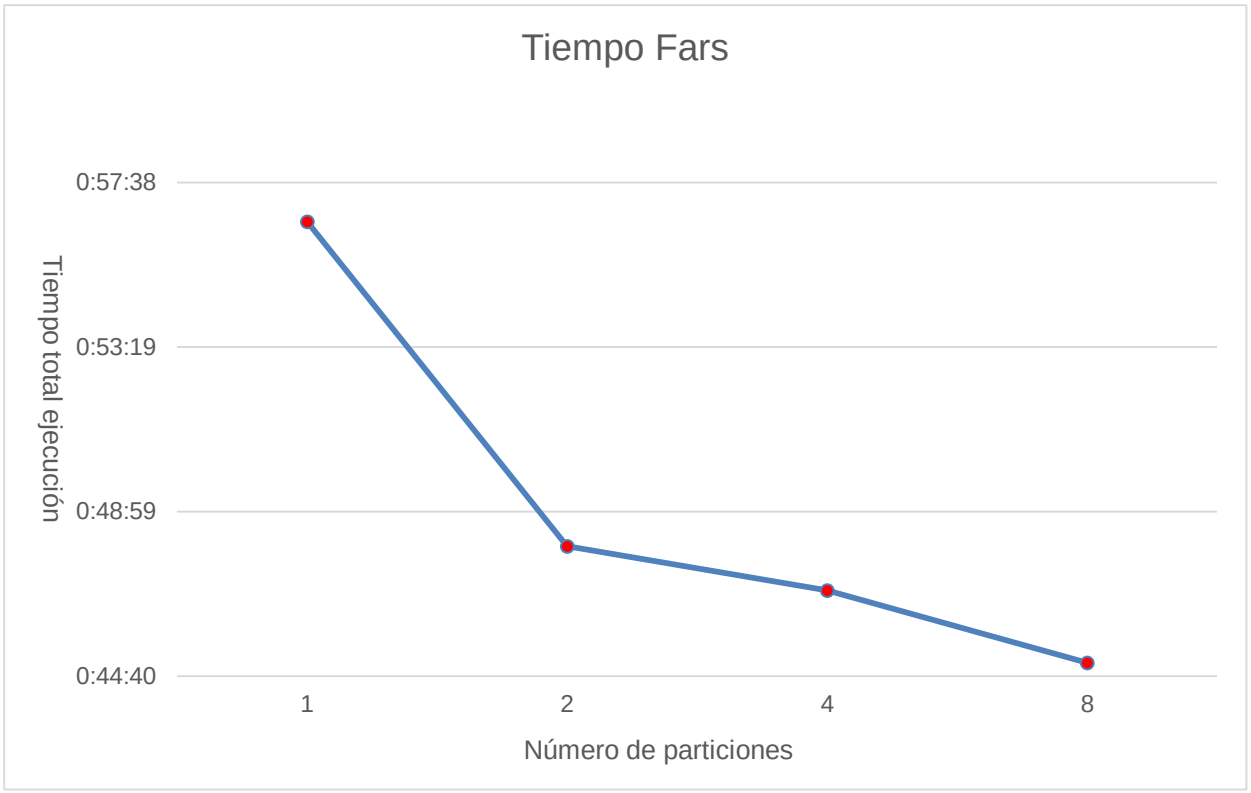


Figura 7.3 Comparación de tiempos con diferentes particiones internas para dataset Fars.
Fuente: Propia

Como se puede observar en la figura 7.3, y sobre lo que se ha hablado acerca de la tabla 7.7, gráficamente se ve de manera más sencilla como en este dataset al no ser lo suficientemente grande, las diferencias entre los tiempos de las diversas particiones no son tan apreciables como en el caso de los datasets *Poker y Census*.

Sin embargo, sigue existiendo una reducción del tiempo, aunque sea menos alarmante que en otros conjuntos de datos.

7.2.4 Tablas de resultados del dataset 'Connect-4'

Se han ejecutado las 5 particiones del conjunto de datos *Connect-4*, y se han obtenido los siguientes resultados de *Accuracy* tanto de los métodos de clasificación como el del modelo Bagging:

	Naive Bayes	MPC	Decision Tree	Random Forest	Resultado Bagging
Partición 1	0,657	0,651	0,651	0,661	0,658
Partición 2	0,657	0,686	0,699	0,682	0,679
Partición 3	0,657	0,673	0,647	0,668	0,672
Partición 4	0,657	0,664	0,643	0,658	0,661
Partición 5	0,656	0,642	0,642	0,660	0,660
Media de métodos	0,657	0,663	0,656	0,666	
Media de medias	0,661				
Media Bagging	0,666				

Tabla 7.8 Resultados de Accuracy para dataset Connect-4.
Fuente: Propia

Como se puede visualizar en la tabla 7.8, los resultados individuales de clasificación y del modelo Bagging no contienen grandes diferencias, mejorando este último la precisión de los métodos individuales de clasificación.

Estamos hablando de una media de precisión del 66,1% de los diversos clasificadores empleados, con respecto al 66,6% que se ha conseguido con el modelo Bagging, por lo que hemos conseguido mejorar la precisión del dataset habiendo usado una gran variedad de clasificadores los cuáles trabajan forma distinta.

Con esto podemos decir que la precisión del ensemble ha sido buena y el ensemble ha trabajado de forma correcta.

		Carga datos	División subdatasets	Clasificadores (Train + Test)	Unión predicciones	Voto
Partición interna 1	Partición 1	0:02:03	0:00:10	0:11:57	0:00:35	0:00:00
	Partición 2	0:00:44	0:00:10	0:11:01	0:00:29	0:00:00
	Partición 3	0:00:41	0:00:09	0:11:46	0:00:37	0:00:00
	Partición 4	0:00:51	0:00:12	0:11:43	0:00:42	0:00:00
	Partición 5	0:00:50	0:00:13	0:11:52	0:00:36	0:00:00
	Suma Tiempos	0:05:09	0:00:54	0:58:19	0:02:59	0:00:00
	Tiempo total	1:07:21				
Partición interna 2	Partición 1	0:01:43	0:00:09	0:09:40	0:00:26	0:00:00
	Partición 2	0:00:41	0:00:08	0:09:17	0:00:25	0:00:00
	Partición 3	0:00:41	0:00:08	0:09:32	0:00:32	0:00:00
	Partición 4	0:00:49	0:00:10	0:10:07	0:00:35	0:00:00
	Partición 5	0:00:51	0:00:10	0:10:18	0:00:35	0:00:00
	Suma Tiempos	0:04:45	0:00:45	0:48:54	0:02:33	0:00:00
	Tiempo total	0:56:57				
Partición interna 4	Partición 1	0:01:49	0:00:08	0:09:30	0:00:30	0:00:00
	Partición 2	0:00:42	0:00:08	0:08:29	0:00:29	0:00:00
	Partición 3	0:00:42	0:00:09	0:08:29	0:00:32	0:00:00
	Partición 4	0:00:46	0:00:09	0:08:38	0:00:33	0:00:00
	Partición 5	0:00:45	0:00:09	0:08:39	0:00:33	0:00:00
	Suma Tiempos	0:04:44	0:00:43	0:43:45	0:02:37	0:00:00
	Tiempo total	0:51:49				
Partición interna 8	Partición 1	0:01:58	0:00:08	0:08:18	0:00:28	0:00:00
	Partición 2	0:00:40	0:00:08	0:07:50	0:00:30	0:00:00
	Partición 3	0:00:42	0:00:09	0:08:02	0:00:32	0:00:00
	Partición 4	0:00:44	0:00:09	0:07:50	0:00:34	0:00:00
	Partición 5	0:00:43	0:00:09	0:07:53	0:00:31	0:00:00
	Suma Tiempos	0:04:47	0:00:43	0:39:53	0:02:35	0:00:00
	Tiempo total	0:47:58				

Tabla 7.9 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Connect-4.
Fuente: Propia

Observando la tabla 7.9, se puede comprobar como la diferencia de tiempos no es tan significativa como en conjuntos de datos de experimentaciones anteriores

Estamos tratando con un caso similar al de la tabla 7.7 sobre el dataset *Fars*, donde el inconveniente estaba en que el conjunto de datos no es lo suficientemente grande para que podamos ver una reducción de tiempo considerable al aumentar el número de particiones.

No obstante, aunque no son tan notorias como en datasets de instancias más grandes, sigue apreciándose una disminución del tiempo, aunque en menor medida, por ejemplo, los tiempos entre las particiones 1 y 2, donde oscilan sobre los 11 minutos, 2 y 4 rondando los 5 minutos, y finalmente 4 y 8, estando sobre unos 4 minutos.

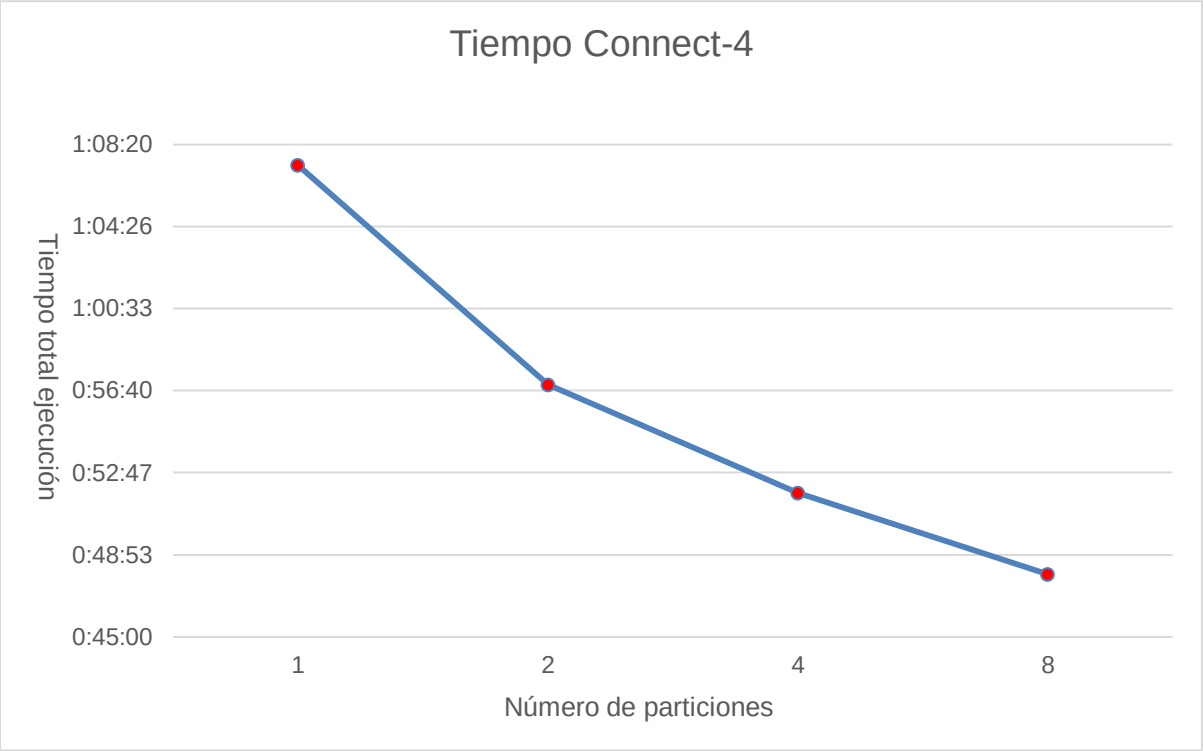


Figura 7.4 Comparación de tiempos con diferentes particiones internas para dataset Connect-4.
Fuente: Propia

Como se puede observar en la figura 7.4, y sobre lo que se ha hablado acerca de la tabla 7.9, gráficamente podemos ver que la diferencia de tiempos no es muy notoria, aunque si se puede apreciar más con las particiones 1, 2 y 4.

A medida que se aumentan el número de particiones se observa cómo va estabilizándose la línea de tiempos, pudiendo obviar particiones internas superiores a 4, puesto que es cuando comienza no notarse una diferencia considerable en los tiempos empleados.

7.2.5 Tablas de resultados del dataset 'Shuttle'

Se han ejecutado las 5 particiones del conjunto de datos *Shuttle*, y se han obtenido los siguientes resultados de *Accuracy* tanto de los métodos de clasificación como el del modelo Bagging:

	MPC	Decision Tree	Random Forest	Resultado Bagging
Partición 1	0,768	0,879	0,851	0,913
Partición 2	0,785	0,543	0,749	0,841
Partición 3	0,915	0,859	0,917	0,920
Partición 4	0,848	0,807	0,826	0,845
Partición 5	0,325	0,428	0,750	0,705
Media de métodos	0,728	0,703	0,819	
Media de medias	0,750			
Media Bagging	0,845			

Tabla 7.10 Resultados de Accuracy para dataset Shuttle.
Fuente: Propia

Como se puede visualizar en la tabla 7.10, los resultados individuales de clasificación y del modelo Bagging son diferentes, siendo los del modelo Bagging superiores a los de los clasificadores.

Se trata de una media de precisión del 75,0% de los diversos clasificadores empleados, con respecto al 84,5% que se ha conseguido con el ensemble, por lo que hemos conseguido mejorar la precisión considerablemente, siendo la diferencia de un 9,5%.

En este caso, los métodos de clasificación en algunos casos han devuelto unos resultados un poco peores con respecto a otros, pero se puede ver como el modelo los ha tratado correctamente consiguiendo una media de precisión muy mejorada.

Se puede decir que el ensemble ha trabajado de forma eficiente y ha sido eficaz, consiguiendo obtener unos resultados muy notorios frente a la utilización de métodos individuales de clasificación.

		Carga datos	División subdatasets	Clasificadores (Train + Test)	Unión predicciones	Voto
Partición interna 1	Partición 1	0:00:27	0:00:02	0:05:38	0:00:04	0:00:00
	Partición 2	0:00:02	0:00:01	0:04:21	0:00:03	0:00:00
	Partición 3	0:00:02	0:00:01	0:04:16	0:00:04	0:00:00
	Partición 4	0:00:01	0:00:01	0:04:11	0:00:05	0:00:00
	Partición 5	0:00:02	0:00:01	0:04:13	0:00:04	0:00:00
	Suma Tiempos	0:00:34	0:00:06	0:22:39	0:00:20	0:00:00
	Tiempo total	0:23:39				
Partición interna 2	Partición 1	0:00:25	0:00:06	0:05:18	0:00:05	0:00:00
	Partición 2	0:00:01	0:00:01	0:04:06	0:00:04	0:00:00
	Partición 3	0:00:02	0:00:01	0:04:04	0:00:04	0:00:00
	Partición 4	0:00:01	0:00:01	0:04:04	0:00:05	0:00:00
	Partición 5	0:00:01	0:00:01	0:04:05	0:00:05	0:00:00
	Suma Tiempos	0:00:30	0:00:10	0:21:37	0:00:23	0:00:00
	Tiempo total	0:22:40				
Partición interna 4	Partición 1	0:00:27	0:00:02	0:05:15	0:00:04	0:00:00
	Partición 2	0:00:02	0:00:01	0:04:02	0:00:05	0:00:00
	Partición 3	0:00:02	0:00:01	0:03:57	0:00:05	0:00:00
	Partición 4	0:00:02	0:00:01	0:03:53	0:00:05	0:00:00
	Partición 5	0:00:02	0:00:00	0:04:08	0:00:04	0:00:00
	Suma Tiempos	0:00:35	0:00:05	0:21:15	0:00:23	0:00:00
	Tiempo total	0:22:18				
Partición interna 8	Partición 1	0:00:25	0:00:05	0:05:01	0:00:05	0:00:00
	Partición 2	0:00:02	0:00:01	0:03:59	0:00:05	0:00:00
	Partición 3	0:00:02	0:00:01	0:03:44	0:00:06	0:00:00
	Partición 4	0:00:02	0:00:01	0:03:42	0:00:05	0:00:00
	Partición 5	0:00:02	0:00:01	0:03:49	0:00:04	0:00:00
	Suma Tiempos	0:00:33	0:00:09	0:20:15	0:00:25	0:00:00
	Tiempo total	0:21:22				

Tabla 7.11 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Shuttle.
Fuente: Propia

Al igual que ocurría con la tabla 7.9 de *Connect*-4, podemos ver en la tabla 7.11 que al tratarse de un dataset no lo suficientemente grande, aumentar el número de particiones no nos conducirá a una disminución de tiempo considerablemente aceptable.

A pesar de esto no se ha dado el caso de que el tiempo se haya quedado igual utilizando diferente número de particiones internas, por lo que sea menor la reducción del tiempo, sigue siendo importante.

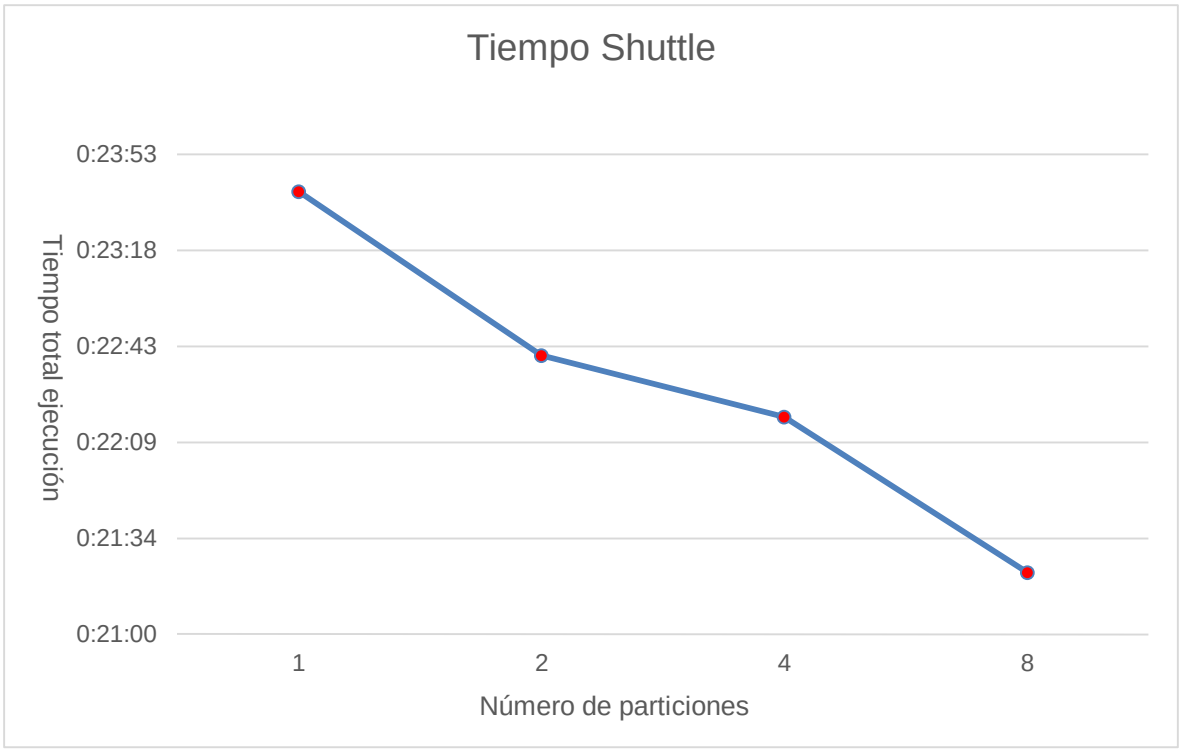


Figura 7.5 Comparación de tiempos con diferentes particiones internas para dataset Shuttle.
Fuente: Propia

Como se puede observar en la figura 7.5, y como bien hemos comentado, la diferencia entre los tiempos no es significativa al tratarse de un conjunto de datos con un número de instancias pequeño. Pero en ningún caso consigue estabilizarse el tiempo como para plantearse obviar utilizar un número de particiones determinado.

7.2.6 Tablas de resultados del dataset 'Mv'

Se han ejecutado las 5 particiones del conjunto de datos *Mv*, y se han obtenido los siguientes resultados de *RMSE, MSE, R2 y MAE* tanto de los métodos de regresión como el del modelo Bagging:

		RMSE	MSE	R2	MAE
Part 1	GBT	1,189	1,413	0,987	0,826
	Decision Tree	1,626	2,645	0,976	1,131
	Random Forest	2,102	4,421	0,960	1,439
	Media de métodos	1,639	2,826	0,974	1,132
	Resultados Bagging	1,178	1,389	0,987	0,819

		RMSE	MSE	R2	MAE
Part 2	GBT	1,152	1,333	0,987	0,805
	Decision Tree	1,583	2,508	0,974	1,101
	Random Forest	2,004	4,094	0,916	1,310
	Media de métodos	1,580	2,645	0,959	1,072
	Resultados Bagging	1,112	1,236	0,988	0,766

		RMSE	MSE	R2	MAE
Part 3	GBT	1,173	1,376	0,987	0,826
	Decision Tree	1,589	2,524	0,977	1,097
	Random Forest	2,050	4,202	0,961	1,406
	Media de métodos	1,604	2,701	0,975	1,110
	Resultados Bagging	1,130	1,277	0,988	0,795

		RMSE	MSE	R2	MAE
Part 4	GBT	1,135	1,287	0,988	0,792
	Decision Tree	1,575	2,482	0,977	1,099
	Random Forest	2,020	4,078	0,963	1,378
	Media de métodos	1,576	2,616	0,976	1,089
	Resultados Bagging	1,091	1,191	0,989	0,756

		RMSE	MSE	R2	MAE
Part 5	GBT	1,161	1,348	0,988	0,805
	Decision Tree	1,593	2,539	0,977	1,109
	Random Forest	2,117	4,482	0,959	1,445
	Media de métodos	1,624	2,790	0,974	1,120
	Resultados Bagging	1,141	1,301	0,988	0,797

Tabla 7.12 Resultados de errores para dataset Mv.
Fuente: Propia

	RMSE	MSE	R2	MAE
Partición 1	1,639	2,826	0,974	1,132
Partición 2	1,580	2,645	0,959	1,072
Particion 3	1,604	2,701	0,975	1,110
Partición 4	1,576	2,616	0,976	1,089
Partición 5	1,624	2,790	0,974	1,120
Media de particiones	1,605	2,716	0,972	1,105
Bagging	1,130	1,279	0,988	0,787

Tabla 7.13 Medias de errores de las diferentes particiones para dataset Mv.
Fuente: Propia

Visualizando la tabla 7.12 y 7.13, podemos ver como el ensemble Bagging se ha comportado correctamente obteniendo una mejora destacable en los errores frente al resto de los regresores individuales.

RSME (Raíz Cuadrada del Error Cuadrático Medio) 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, siendo una buena medida de la precisión con la que el modelo predice la respuesta. Observando la tabla 7.13 vemos como tenemos un *RMSE* del ensemble implementado de 1,130 frente a 1,605 de las distintas particiones, por lo que el ensemble contiene un mejor ajuste.

MSE (Error Cuadrático Medio) mide el error cuadrado promedio de nuestras predicciones. Para cada punto, calcula la diferencia cuadrada entre las predicciones y el objetivo y luego promedia esos valores, y cuanto mayor sea este valor, peor es el modelo, por lo que obtener un 0 sería consecuentemente un modelo perfecto. Como se puede observar en la tabla 7.13, se ha conseguido que el resultado del ensemble sea mucho mejor con respecto a los resultados obtenidos al hacer las medias de las particiones. Obteniendo del ensemble un valor de 1,279 con respecto a las particiones que han obtenido un error del 2,716.

R^2 *(R-cuadrado)* indica la bondad o la aptitud del modelo, y contiene una escala que es intuitiva, yendo de 0 a 1, con 0 indicando que el modelo propuesto no

mejora la predicción sobre el modelo medio y 1 indica una predicción perfecta. Con esto podemos decir haciendo referencia a la tabla 7.13 que la predicción es casi perfecta, puesto que el ensemble obtiene un 98,8% frente al 97,2% que obtenemos de las diversas particiones.

MAE (Error Absoluto Medio) es el promedio de la diferencia absoluta entre el valor observado y los valores predichos. El error absoluto medio o MAE es un puntaje lineal, significando esto que todas las diferencias individuales se ponderarán por igual en el promedio. En este caso el ensemble vuelve a obtener un mejor resultado con respecto a las particiones, obteniendo éstas 1,105 frente al resultado del ensemble Bagging 0,787.

Con todos los datos comentados, podemos decir que el ensemble Bagging ha vuelto a actuar correctamente, haciéndonos obtener unos resultados más fiables que empleando diversas técnicas de regresión individualmente.

		Carga datos	División subdatasets	Regresores (Train + Test)	Unión predicciones	Media
Partición interna 1	Partición 1	0:00:33	0:00:08	0:04:10	0:00:04	0:00:00
	Partición 2	0:00:02	0:00:01	0:02:51	0:00:06	0:00:00
	Partición 3	0:00:02	0:00:01	0:02:41	0:00:03	0:00:00
	Partición 4	0:00:01	0:00:01	0:02:39	0:00:05	0:00:00
	Partición 5	0:00:01	0:00:01	0:02:40	0:00:04	0:00:00
	Suma Tiempos	0:00:39	0:00:12	0:15:01	0:00:22	0:00:00
	Tiempo total	0:16:14				
Partición interna 2	Partición 1	0:00:34	0:00:06	0:03:58	0:00:04	0:00:00
	Partición 2	0:00:02	0:00:01	0:02:40	0:00:05	0:00:00
	Partición 3	0:00:02	0:00:01	0:02:43	0:00:04	0:00:00
	Partición 4	0:00:01	0:00:01	0:02:38	0:00:04	0:00:00
	Partición 5	0:00:01	0:00:01	0:02:31	0:00:05	0:00:00
	Suma Tiempos	0:00:40	0:00:10	0:14:30	0:00:22	0:00:00
	Tiempo total	0:15:42				
Partición interna 4	Partición 1	0:00:27	0:00:03	0:03:38	0:00:03	0:00:00
	Partición 2	0:00:01	0:00:01	0:02:33	0:00:03	0:00:00
	Partición 3	0:00:01	0:00:01	0:02:36	0:00:04	0:00:00
	Partición 4	0:00:01	0:00:01	0:02:31	0:00:05	0:00:00
	Partición 5	0:00:01	0:00:01	0:02:34	0:00:04	0:00:00
	Suma Tiempos	0:00:31	0:00:07	0:13:52	0:00:19	0:00:00
	Tiempo total	0:14:49				
Partición interna 8	Partición 1	0:00:24	0:00:03	0:03:08	0:00:04	0:00:00
	Partición 2	0:00:01	0:00:01	0:02:35	0:00:03	0:00:00
	Partición 3	0:00:01	0:00:00	0:02:25	0:00:03	0:00:00
	Partición 4	0:00:01	0:00:01	0:02:31	0:00:04	0:00:00
	Partición 5	0:00:02	0:00:00	0:02:29	0:00:04	0:00:00
	Suma Tiempos	0:00:29	0:00:05	0:13:08	0:00:18	0:00:00
	Tiempo total	0:14:00				

Tabla 7.14 Tiempos de ejecución del ensemble Bagging con distintas particiones internas para dataset Mv.
Fuente: Propia

En este caso estamos tratando con un conjunto de datos con un número de instancias no lo suficiente grande como para que las diferencias entre los tiempos de diversas particiones sean muy grandes.

Se puede apreciar en la tabla 7.14, ya que los tiempos totales de ejecución utilizando diversos números de particiones no son muy significativos unos con otros.

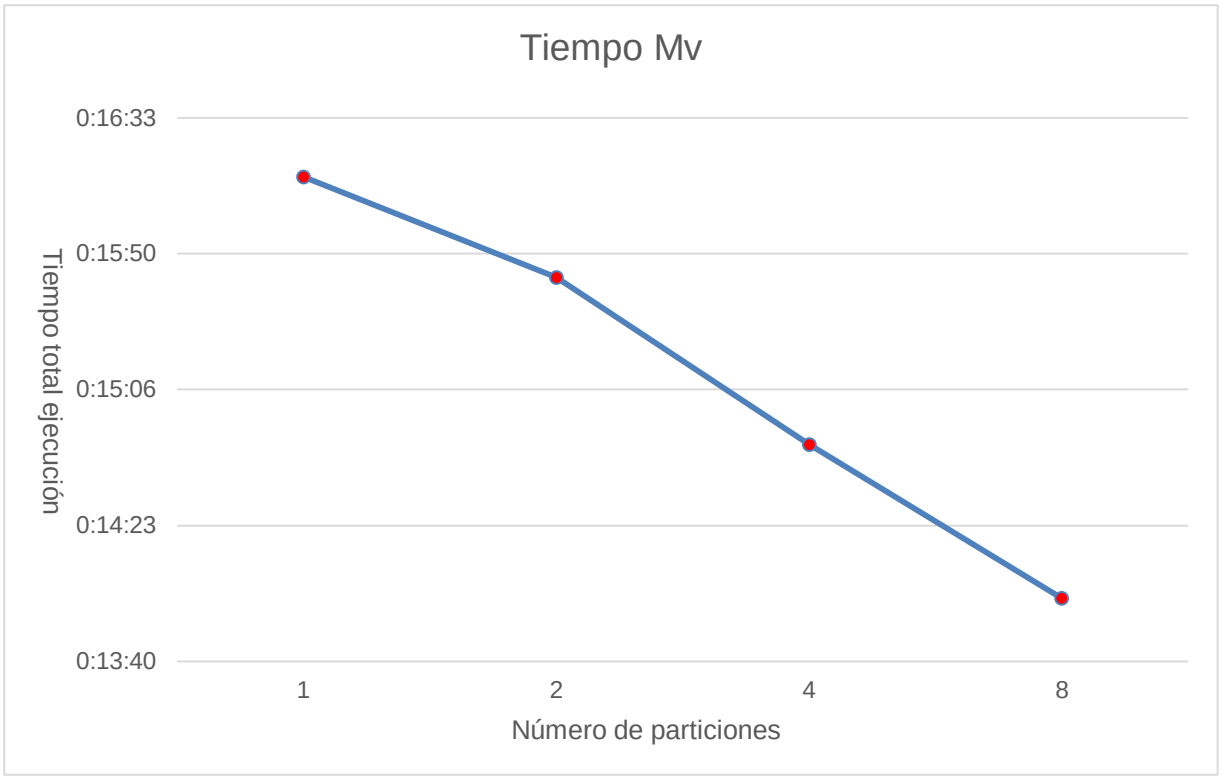


Figura 7.6 Comparación de tiempos con diferentes particiones internas para dataset Mv.
Fuente: Propia

Como se puede apreciar en la figura 7.6, y como bien hemos comentado, la diferencia entre los tiempos no es muy característico al tratarse de un conjunto de datos con un número de instancias no lo necesariamente grande.

8. CONCLUSIONES

En este último capítulo de este trabajo de final de grado, se exponen las conclusiones de realización del mismo, además de las conclusiones personales y los trabajos futuros que pueden llegar a realizarse.

8.1 Conclusiones del proyecto

En este Trabajo de Fin de Grado se ha realizado un estudio de las diferentes áreas de minería de datos, Big Data y Ciencia de Datos, así también como la teoría de ensembles, obteniendo así el marco teórico completo sobre el que se desarrolla el proyecto. También se ha realizado un estudio de las herramientas de procesamiento Big Data más conocidas, centrándonos principalmente en el procesamiento de datos con Apache Spark.

Se ha escogido Spark como herramienta base para llevar a cabo el desarrollo e implementación de ensembles, como en nuestro caso hemos realizado el ensemble Bagging. No obstante, no solo se ha utilizado Apache Spark como base para la implementación, sino que también se utiliza la librería ML de Spark, siendo ésta una biblioteca de machine learning, puesto que para el desarrollo del ensemble, se han necesitado usar diversos métodos de clasificación o regresión. Además, se ha hecho uso de un servidor de la Universidad de Jaén para realizar los diferentes experimentos expuestos en el apartado de resultados, con el objetivo de que pudieran utilizar diferentes particiones internas de Spark y así poder comprobar la eficacia de la paralelización.

Con el uso de los diferentes métodos de MLlib, e implementado el ensemble Bagging, se lleva a cabo una experimentación con diferentes conjuntos de datos públicos, obtenidos del repositorio KEEL.

El resultado de los experimentos pone de manifiesto que el uso de la técnica de ensemble Bagging distribuido, suele obtener resultados superiores con respecto a los obtenidos por los clasificadores base utilizados. Sólo en uno de los experimentos realizados se ha mantenido la precisión frente a la media de los

métodos de clasificación empleados. Por tanto, con este estudio se puede afirmar que el desarrollo de modelos de ensembles logran una mayor eficiencia que el uso individual de los diversos métodos de clasificación o regresión.

A su vez, se ha demostrado que el método paraleliza correctamente y en todos los casos experimentados se han conseguido mejores tiempos conforme se han aumentado el número de particiones que se procesan en paralelo.

Sin embargo, también se pone de manifiesto que el uso de Spark para procesar conjuntos pequeños de datos es contraproducente, puesto que el coste del paralelismo es mayor que el beneficio obtenido.

8.2 Conclusiones personales

Realizar éste proyecto me ha permitido introducirme en la minería de datos y en diferentes tecnologías de Data Science y Big Data, así como obtener un nuevo conocimiento sobre Apache Spark y su lenguaje de programación Scala y aplicarlo sobre la teoría de ensembles.

Tras la realización del mismo, he aumentado mi conocimiento sobre las áreas de las ciencias de datos y Big Data tanto de forma teórica como práctica, aunque aún me queda mucho por aprender por lo que seguiré profundizando en el estudio de ellas.

Puede que el mayor beneficio obtenido con el desarrollo del trabajo, ha sido el aprendizaje técnico de Apache Spark, ML y Scala, puesto que, en mi opinión, son herramientas muy interesantes, aunque también complejas, y con una gran proyección en el mundo profesional. Además, también he podido acceder a la utilización de un servidor donde poder lanzar mis experimentos utilizando paralelismo.

Gracias a este proyecto he descubierto mi interés relacionado con el Big Data y Data Science, haciéndome ver un nuevo camino de posibilidades de las que no era consciente hasta la realización de este proyecto.

9. REQUISITOS PARA EL DESARROLLO

9.1 IntelliJ IDEA

Se trata de un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos y es desarrollado por Jetbrains.

Tiene soporte para múltiples lenguajes como Scala, Java, PHP, entre muchos otros. Contiene dos tipos de ediciones, una para la comunidad, y una comercial, siendo la primera gratuita y la utilizada para el desarrollo del trabajo de fin de grado.

9.2 Lanzamiento de aplicación en el clúster

Debido a que tenemos acceso a un servidor de un grupo de investigación de la Universidad de Jaén, podremos lanzar nuestra aplicación a un clúster, además de disponer de un número de nodos por lo que la podremos particionar.

Spark proporciona el comando *spark-submit* que se encuentra en el directorio */bin* para lanzar aplicaciones al clúster. Este comando contiene multitud de opciones, como es el ejemplo de algunas:

```
$./SPARK_HOME/bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
   <jar de la aplicación> \
  [argumentos de la aplicación]
```

Figura 9.1 Fichero .sh de parámetros con el jar del proyecto.
Fuente: Propia

- -- class: Clase principal de la aplicación.
- --master: URL para el clúster.

- --deploy-mode: Modo de desplegar el proceso driver, bien en los nodos de trabajo (cluster), o localmente como un cliente externo (client).
- --conf: Propiedades para la ejecución del trabajo, como por ejemplo número de núcleos.
- jar de la aplicacion: Ruta hacia un jar que incluye su aplicación y todas las dependencias. Debe estrar en un directorio común, donde todos los nodos del clúster tengan acceso.
- Argumentos de la aplicación: son los argumentos que se le pasarán a la función main de la clase indicada en el parámetro --class

A continuación, se muestra un ejemplo del lanzamiento de la aplicación que realizar el ensemble Bagging, implementado en este proyecto, sobre el clúster BIGDATA:

```
#!/bin/bash

$SPARK_HOME/bin/spark-submit
    --master spark://bigdata:7077
    /home/simidat/amm00254/pruebasTFG/out/artifacts/pruebasTFG_jar/pruebasTFG.jar
```

Figura 9.2 Ejemplo de fichero .sh.
Fuente: Propia

Un aspecto fundamental a tener en cuenta previamente a lanzar nuestra aplicación en el servidor, es que las versiones de ambos Spark, tanto de nuestro proyecto como el que hay en el servidor, debe ser la misma, en caso contrario, no funcionaría.

10. MANUAL DE USUARIO

A continuación, se detallará que pasos deben seguirse para una correcta ejecución y obtención de resultados de la aplicación.

El programa consta de diversos ficheros de parámetros que deberemos configurar dependiendo de cómo queramos que sea nuestra ejecución, vamos a suponer que se realiza desde el servidor, en local habría que cambiar las rutas:

parametrosArchivosES:

- Ruta del fichero parámetros método Logistic Regression.
- Ruta del fichero parámetros método Decision Tree.
- Ruta del fichero parámetros método Random Forest.
- Ruta donde se guardarán los modelos obtenidos del ensemble Bagging.
- Variable booleana que indica si se usan particiones o no
    - False: no se usan particiones, y la siguiente ruta que se pone es únicamente la ruta del dataset que contiene interiormente Test y Train juntos.
    - True: se usan particiones, y las siguientes rutas que se ponen son las de las particiones, intercalando una ruta de train y una de test.

```
/home/simidat/amm00254/pruebasTFG/data/param/parametrosLogisticRegression.txt
/home/simidat/amm00254/pruebasTFG/data/param/parametrosDecisionTree.txt
/home/simidat/amm00254/pruebasTFG/data/param/parametrosRandomForest.txt
/home/simidat/amm00254/modelData/Bagging/BaggingMagic/BaggingMagic1/
true
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTrain1.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTest1.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTrain2.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTest2.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTrain3.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTest3.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTrain4.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTest4.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTrain5.data
hdfs://turing17.turing.ceatic.ujaen.es:8020/amm00254/datasets/magicTest5.data
```

Figura 10.1 Ejemplo de fichero de parámetros de archivos.
Fuente: Propia

parametrosModelosClass:

- Se ponen por fila los nombres de los métodos que se deseen usar: **SVM, MPC, LogisticRegression, NaiveBayes, RandomForest, DecisionTree, GBT, Isotonic.**

  A la hora de escoger métodos, debemos tener en cuenta como es nuestro dataset, ya que unos funcionan para clases binarias, y otros para multiclases. Y en cada línea del archivo se especifica un método.

```
NaiveBayes
SVM
LogisticRegression
MPC
DecisionTree
RandomForest
```

Figura 10.2 Ejemplo de fichero de parámetros de clasificadores base. Fuente: Propia

Métodos	Tipos de clase
Linear Support Vector Machine (SVM)	Binarios
Multilayer Perceptron Classifier (MPC)	Binarios / Multiclase
Logistic Regression	Binarios
Naive Bayes	Binarios / Multiclase / Reales
Decision Tree	Binarios / Multiclase / Reales
Gradient Boosted Tree (GBT)	Reales
Isotonic	Reales
Random Forest	Binarios / Multiclase

Figura 10.3 Clasificación métodos ML. Fuente: Propia

parametrosNumericos:

- Parámetro que indica cuánto porcentaje del dataset real se quedará para Train. Teniendo en cuenta que usemos un dataset con Test y Train juntos.
- Parámetro que indica cuánto porcentaje del dataset real se quedará para Test. Teniendo en cuenta que usemos un dataset con Test y Train juntos.
- Número de clasificadores que deseamos que se ejecuten.
- Porcentaje del dataset de Train que tendrán nuestros subdatasets que se enviarán a cada clasificador.
- Número de particiones internas de Spark.

```
0.7
0.3
3
0.6
1
```

Figura 10.4 Ejemplo de fichero de parámetros númericos.
Fuente: Propia

parametrosCR:

- Parámetro que indicará si queremos que se realice una clasificación o una regresión en nuestro ensemble. Se debe poner: **clasificacion o regresion**.

  En caso de regresion, los métodos del fichero parametrosModelosClass únicamente pueden ser: DecisionTree y RandomForest, ya que son los implementados para este tipo de datasets.

parametrosLogisticRegression:

- Número máximo de iteraciones
- Parámetro de regularización

- Parámetro de mezcla ElasticNet. Para alfa = 0, la penalización es una penalización L2. Para alfa = 1, es una penalización L1. Para alfa en (0,1), la penalización es una combinación de L1 y L2. El valor predeterminado es 0.0, que es una penalización L2

parametrosRandomForest:

- Número de categorías máximas Vector Indexer
- Número de árboles

parametrosDecisionTree:

- Número de categorías máximas Vector Indexer

Si no modificamos los ficheros de parámetros de los métodos, se quedarán por defecto como están, pero los que si son importantes cambiar son:

- Rutas de los datasets a ejecutar.
- Métodos de modelos que se quieran ejecutar.
- Número de clasificadores.
- Porcentaje que se cogerá del conjunto de datos de entrenamiento para los subconjuntos de datos.
- Número de particiones.

Dentro de IntelliJ IDEA, seleccionaremos nuestro proyecto el cual contiene nuestra aplicación, y deberemos generar un **jar** para que pueda ser lanzado desde el servidor: **File → Project Structure → Artifacts → + → jar → From modules with dependencies.. → Seleccionamos nuestra clase → Apply/Ok**

Y debemos construir el **jar**: **Build → Build Artifacts → Seleccionamos jar → Build**.

Tras haber hecho esto, ya estará nuestro proyecto listo para ser importado dentro del servidor.

Ahora dentro del servidor deberemos generar dos ficheros, que nos proporcionarán una correcta ejecución de la aplicación: un fichero **.sh** y otro **.sbs**, siendo este último el que se lanzará:

- **.sh**

```
#!/bin/bash

$SPARK_HOME/bin/spark-submit
    --master spark://bigdata:7077
    /home/simidat/amm00254/pruebasTFG/out/artifacts/pruebasTFG_jar/pruebasTFG.jar
```

Figura 10.5 Ejemplo de fichero .sh.
Fuente: Propia

- **.sbs**

```
#!/bin/bash

#SBATCH --job-name=pruebaEjTFG
#SBATCH --partition=spark
#SBATCH --output=Resultados/salidaPruebaTFGRegressionElevator8_%j.out
#SBATCH --error=Errores/salidaPruebaTFGRegressionElevator8_%j.err
#SBATCH --mail-user=amm00254@red.ujaen.es

/home/simidat/amm00254/prueba.sh
```

Figura 10.6 Ejemplo de fichero .sbs.
Fuente: Propia

Este último fichero es el que se ejecutará dentro de la Shell y consta de las siguientes opciones:

- **--job-name**: Será el nombre que aparecerá en ejecución dentro de Spark
- **--partition**: Particiones internas a través de Spark
- **--output**: Ruta donde se nos generará un fichero con el nombre deseado con las salidas de nuestra aplicación, en el caso de este proyecto mostrará los resultados de los modelos individuales y del ensemble al final, así como los tiempos empleados.
- **--error**: Ruta donde se nos generará un fichero con el nombre deseado con los errores que puedan surgir durante la ejecución del programa o posibles WARNINGS. Muy útil cuando no somos conscientes de terminados bugs o errores que puedan aparecer.

Y finalmente lanzaremos a ejecutar nuestra aplicación, con el comando **sbatch ejecucion.sbs**

BIBLIOGRAFÍA

Aggarwal, C. C. (2015). *Data classification: Algorithms and applications.* Yorktown Heights: Chapman & Hall / CRC.

Aishwarya, S. (2018, Junio 18). *A Comprehensive Guide to Ensemble Learning*. Retrieved from https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. In *Journal of Multiple-Valued Logic & Soft Computing* (p. 17).

Alex, W. (2019, Marzo 8). *Figura 3.5 Estructura de Spark*. Retrieved from https://www.datanami.com/2019/03/08/a-decade-later-apache-spark-still-going-strong/

Apache. (Sin fecha). *Apache spark. Guía de programación de RDD y Shuffle*. Retrieved from https://spark.apache.org/docs/latest/rdd-programming-guide.html

Apache. (Última actualización 2020). *Apache Spark*. Retrieved from https://spark.apache.org/

Arthur Asuncion, & David Newman. (2007). *UCI - MACHINE LEARNING REPOSITORY*. Retrieved from https://archive.ics.uci.edu/ml/datasets.php

Azevedo, A., & Santos, M. F. (2008). KDD, SEMMA and CRISP-DM: A parallel overview.

Berzal, F. (2014). *Árboles de decisión.*

Borja, R. P. (2017). *Figura 3.10 Nº de desarrolladores de ambas plataformas. Figura 3.11 Comparativa de récords*. Retrieved from http://oa.upm.es/47243/1/TFG_BORJA_RODRIGUEZ_PANOS.pdf (Página 11) y (Página 15)

Charles, E. (2011, Enero 18). *Evaluating Classifiers. University of California.* Retrieved from https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada#cite_note-loocv2-7

Cleveland, W. S. (2001). Data science: an action plan for expanding the technical areas of the field of statistics. In *International Statistical Review / Revue Internationale de Statistique* (pp. 21-26).

Datacentric. (2018). *Netflix: Las claves del éxito basado en Big Data*. Retrieved from https://www.datacentric.es/blog/insight/exito-netflix-datos/

Dataflair. (2018). *DAG*. Retrieved from https://data-flair.training/blogs/dag-in-apache-spark/

Data-flair.training. (2020, Marzo 2). *Figura 3.3 Arquitectura HDFS*. Retrieved from https://data-flair.training/blogs/hadoop-hdfs-architecture/

Datahack. (2019). *MLlib, la biblioteca de Machine Learning de Spark*. Retrieved from https://www.datahack.es/mllib-machine-learning-spark/

Devijver, P. A., & Kittler, J. (1982). *Pattern Recognition: A Statistical Approach.* Pretince-Hall, Londres.

Doug, C., & Mike, C. (2002). *Nutch, buscador Web de código abierto*. Retrieved from https://www.sas.com/es_es/insights/big-data/hadoop.html#hadoopimportance

Edward, M. (2019, Junio 11). *Figura 2.1 Disciplinas del Data Science*. Retrieved from https://www.icemd.com/digital-knowledge/articulos/spark-vs-hadoop-cual-es-mejor/

Fravak, B. (2017, Julio 12). *Figura 3.6 Operaciones sobre un RDD*. Retrieved from https://medium.com/@fravak.bharucha/understanding-spark-ii-rdd-operations-41493395ad6a

Gartner, D. L. (2001).

Ghahramani, Z. (2004). *Unsupervised Learning.*

Hamidah, J., Abdul, R. H., & Zulaiha, A. o. (2009). *Figura 2.3 Proceso de clasificación*. Retrieved from https://www.semanticscholar.org/paper/Potential-Data-Mining-Classification-Techniques-for-Jantan-Hamdan/c2a24a3c9e1dc9913a543f090aa6d928b2fdf1a9/figure/2

IDC. (2017). *diarioti.com*. Retrieved from https://diarioti.com/el-volumen-de-datos-total-a-nivel-mundial-aumentara-en-10-veces-para-2025/104972

Jean-Philippe, L. (2014, Enero 3). *BioInformatic Department*. Retrieved from https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada#cite_note-loocv2-7

Joan, D. (2011, Diciembre 8). *Figura 2.4 Esquema k-fold cross validation*. Retrieved from https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada#/media/Archivo:Esquema_castell%C3%A0.jpg

Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning Spark. Lightning-fast data analysis (1ª edición).* O'Reilly.

Kaspersky. (2019, Abril 5). *Data Mining*. Retrieved from https://latam.kaspersky.com/resource-center/definitions/data-mining

Kotsiantis, S. B. (2007). *Supervised Machine Learning: A review of classification techniques.*

Kusnetzky, D. (2010). *What is "Big Data"?* Retrieved from https://web.archive.org/web/20100221024502/http://blogs.zdnet.com/virtualization/?p=1708

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. META Group Research Note ,6 ,70.

Marilín, G. (2013). *Los datos masivos (o big data) son el nuevo oro*. Retrieved from https://www.eldiario.es/turing/Big-data_0_161334397.html

Mashey, J. R. (1998). *Big Data ... and the Next Wave of InfraStress. Usenix.* Retrieved from https://es.wikipedia.org/wiki/Macrodatos#cite_ref-43

Mesa, A. R. (2018). *Qué es Apache Spark. Características*. Retrieved from https://openwebinars.net/blog/que-es-apache-spark/

Mohammed, E. (2016). *Figura 3.4 Map Reduce*. Retrieved from https://www.researchgate.net/figure/The-MapReduce-architecture-MapReduce-Algorithm-There-are-four-steps-to-implement_fig2_305489358

Monnapa, A. (2020, Marzo 11). *Simplilearn*. Retrieved from https://www.simplilearn.com/data-science-vs-big-data-vs-data-analytics-article

Necati, D. (Sin fecha). *Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results*. Retrieved from https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

Oded, M., & Lior, R. (2010). *Data Mining and Knowledge Discovery Handbook.* Springer, New York. ISBN 978-0-387-09823-4.

Odersky, M., A., P., C., V., E., B., M., S., M., . . . & Zenger, M. (2004). An overview of the Scala programming language. *(No. LAMP-REPORT-2004-006)*.

O'Reilly. (2014). *Bid Data now: 2014 edition.*

Paloma, R. d. (2017). *Hadoop por dentro(II): HDFS y MapReduce*. Retrieved from https://empresas.blogthinkbig.com/hadoop-por-dentro-ii-hdfs-y-mapreduce/

Payam, R., Lei, T., & Huan, L. (2008).

Raynel, B. (Sin fecha). *Figura 2.2 Proceso del KDD*. Retrieved from https://www.researchgate.net/figure/Etapas-del-Descubrimiento-de-Conocimiento-en-Base-de-Datos-KDD-por-sus-siglas-en-Ingles_fig4_316212474

Researchgate. (2018). *Figura 3.2 Tres Vs en Big Data*. Retrieved from https://www.researchgate.net/profile/Alp_Kiziltan/publication/334204077/figure/fig4/AS:776588660572165@1562164573544/3-The-three-Vs-of-big-data-Psannis-et-al-2018.ppm

SAS. (n.d.). *Información sobre Hadoop*. Retrieved from https://www.sas.com/es_es/insights/big-data/hadoop.html#hadoopimportance

Scott, F. (2015). *Figura 4.2 Variaciones en los errores. Figura 4.3 Complejidad del modelo y error*. Retrieved from https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

SoHPC-Team. (2015, Diciembre 23). *Figura 3.8 DAG*. Retrieved from https://summerofhpc.prace-ri.eu/apache-spark-bridge-between-hpc-and-big-data/

Tavish, S. S. (2015, Agosto 2). *Basics of Ensemble Learning Explained in Simple English. Figura 4.1 Error en Ensemble Learning*. Retrieved from https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

Tecnología-e-innovación. (2012, Noviembre 28). *Figura 3.1 Datos creados vs Almacenamiento en Big Data*. Retrieved from https://www.tecnologiaeinnovacion.defensa.gob.es/es-es/Contenido/Paginas/detallereferencia.aspx?referenciaID=33

Usuario-StackOverflow. (2016, Mayo 30). *Figura 3.7 Aplicación de Shuffle*. Retrieved from https://stackoverflow.com/questions/37528047/how-are-stages-split-into-tasks-in-spark

Validación cruzada. (n.d.). Retrieved from https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada#cite_note-loocv2-7

Vance, A. (2009, Marzo 17). *Hadoop, a Free Software Program, Finds Uses Beyond Search*. Retrieved from https://www.nytimes.com/2009/03/17/technology/business-computing/17cloud.html

Victor, V. F. (2019, Julio 26). *Spark vs Hadoop, ¿cuál es mejor?* Retrieved from https://www.icemd.com/digital-knowledge/articulos/spark-vs-hadoop-cual-es-mejor/

Wikipedia. (Última actualizacion 2019). *Apache hadoop*. Retrieved from https://es.wikipedia.org/wiki/Apache_Hadoop#cite_ref-6

Zaharia, M., Chowdhury, M., Franklin, M. J., & Shenker, S. S. (2010). *Spark: Cluster Computing with Working Sets.* University of California, Berkeley.

Zhu, X., & Goldberg, A. B. (2009). *Introduction to Semi-Supervised learning.*

Zikopoulos, P. C., Eaton, C., deRoos, D., Deutsch, T., & Lapis, G. (2011). Understanding Big Data - Analytics for entreprise class haddop and streaming data.