ISSN 1870-4069

Entornos de trabajo para procesamiento de datos masivos y aprendizaje automático

Angélica Guzmán Ponce[1], Rosa María Valdovinos Rosas[1], José Raymundo Marcial Romero[1], Roberto Alejo Eleuterio[2]

[1] Universidad Autónoma del Estado de México, Facultad de Ingeniería, Toluca, Estado de México, México

[2] Instituto Tecnológico de Toluca, Estado de México, México

aguzmanp@alumni.uaemex.mx, {rvaldovinosr,jrmarcialr}@uaemex.mx, ralejoll@hotmail.com

Resumen. *Big Data*, ha surgido como un concepto tanto en ámbitos públicos como privados por la creciente demanda de información, ya sea desde la web en redes sociales hasta por medio de los sensores de un teléfono móvil. La creciente demanda de datos ha implicado el buscar alternativas en técnicas de aprendizaje clásico que aborden los desafíos que *Big Data* presenta en sus características, el éxito depende de las metodologías de análisis y rapidez computacional que se utilicen. Para la aplicación de *Big Data*, recientemente han surgido entornos de trabajo que prometen un adecuado funcionamiento en contexto de grandes volúmenes de datos. Para identificar las bondades que cada uno ofrece, en este artículo se presenta un análisis desde tres perspectivas, sus características generales, los algoritmos de aprendizaje automático con los que disponen y la complejidad de su implementación y operación, de este modo determinar el mejor entorno de trabajo para *Big Data*.

Palabras clave: Big data, hadoop, spark, flink, aprendizaje automático, entornos de trabajo.

Frameworks for Big Data and Machine Learning

Abstract. *Big Data* has emerged as a concept in public and private spheres due to the growing demand for information of sources like web on social networks or through the sensors of a mobile phone. The growing demand for data has involved looking for alternatives in classical learning techniques that try to solve the challenges that *Big Data* has in its characteristics, the success depends on the methodologies of analysis and computational speed that are used. To apply *Big Data*, there are frameworks that have recently emerged and promise an adequate functioning in context of large volumes of data. To identify the benefits that each

one offers, in this paper, we present an analysis from three perspectives, the first one in general characteristics, the second one are algorithms of Machine Learning are implemented and the last one, the complexity of their implementation and operation, in this way to determine the best environment to *Big Data*.

Keywords: Big data, hadoop, spark, flink, machine learning, frameworks.

1. Introducción

Con la creciente demanda de información, el término *Big Data* ha tomado importancia en diversos ámbitos ya sean públicos o privados. De acuerdo a Morabito [19], Edd Dumbill define a *Big Data* como *el dato que excede la capacidad de procesamiento de sistemas de base de datos convencionales, siendo tan grande que los movimientos de datos son rápidos*. Es decir que, el volumen de los datos es tal que para tecnologías de bases de datos actuales, el procesamiento de estos se vuelve complejo y en ocasiones imposible de procesar. Derivado de esto, las principales características de *Big Data* de acuerdo a la literatura impactan en tres conceptos fundamentales [18]:

- *Volumen*: Se refiere a la cantidad de datos disponibles para su procesamiento. Por ejemplo, la creciente interacción de usuarios en redes sociales como Twitter, con el uso de hashtags (etiquetas) en reacción a eventos causan una enorme producción de datos en cortos periodos de tiempo.
- *Variedad*: Se refiere a la diversidad de fuentes de datos o también en los formatos de datos. Por ejemplo social media, tweets, correos, información de sensores, producen diferentes formatos así como también son diversas maneras de obtener información.
- *Velocidad*: Se refiere a la rapidez del flujo de datos que tienen que ser manejados por el sistema, es decir, el proceso de generar datos en cortos periodos de tiempo implica una agilidad del proceso en captarlos y procesarlos.

Adicionalmente a estas características, algunas investigaciones consideran dos más, las cuales impactan más en los resultados de un análisis que en la generación de datos son la *Veracidad* y *Valor*:

- *Veracidad*: Se refiere a la calidad y confianza de los datos disponibles en un grado incomparable de volumen, velocidad y variedad [19]. Es decir, los datos que se están utilizando son los adecuados para analizar un problema, en pocas palabras se habla de calidad de los datos.
- *Valor*: Se refiere al proceso de extraer información oculta de datos emergentes [18]. En pocas palabras, se hace referencia al resultado deseado del proceso de análisis de *Big Data*.

Como se muestra en la Tabla 1, para enfrentar los grandes retos de *Big Data* algunas estrategias son: 1. Diseño de ambientes escalables. 2. Proveer Tolerancia a fallos. 3. Diseño de soluciones eficientes. [8].

Tabla 1. Retos de Big Data [21].

Característica	**Reto**
Volumen	Gran escala: El volumen de datos a procesar puede ser tal que los recursos disponibles para ello no lo permitan.
Variedad	Alta dimensionalidad con tipos de datos mezclados con distribución lineal.
Velocidad	Tiempo de llegada de datos: real data streams y alta velocidad.
Veracidad	El contenido de los datos puede ser incierto o incompleto.
Valor	Bajo valor de densidad, Significado diverso de datos.

Debido a la gran cantidad de datos que se manejan en tiempo real, el éxito en *Big Data* depende de metodologías de análisis y rapidez computacional que se utilicen. Para ello, se buscan algoritmos de optimización que no sólo trabajen rápido sino que también reduzcan el uso de memoria [21].

De igual modo, la mayoría de las herramientas utilizadas en minería de datos para almacenamiento, procesamiento y análisis de datos se han vuelto poco eficientes para el tratamiento masivo de datos heterogéneos. En consecuencia se tiene la necesidad de encontrar o proponer alternativas que solventen la demanda de *Big Data*.

Algunas de estas propuestas son consideradas por algunos de los principales entornos de trabajo disponibles para *Big Data* en los que estudios como en [8,12,14,22] hace uso de Spark [4], Hadoop [3], Flink [2], Google Cloud [6], Amazon [1], entre otros, considerando la tolerancia a fallos, así como también el rendimiento de la solución hasta el diseño de la arquitectura. De estos entornos de trabajo resulta de interés identificar las bondades que ofrecen en la implementación, uso y manejo de algoritmos de aprendizaje automático.

En este artículo, se presenta un estudio comparativo de tres de los entornos de trabajo más usados en el área científica para el tratamiento de grandes volúmenes de información con características propias de *Big Data* y la integración de algoritmos de aprendizaje automático. El resto del artículo está organizado como sigue: En la Sección 1.1 se introduce la importancia de aprendizaje automático en *Big Data*, mientras que en la Sección 3 se describen tres de los entornos de trabajo que ayudan en tareas propias de *Big Data*. En la Sección 4 se describen características de cada entorno de trabajo, así como también los algoritmos de aprendizaje automático implementados, por último las conclusiones se presentan en la Sección 5.

1.1. Aprendizaje automático

Como ya se mencionó, una de las áreas que están apoyando fuertemente la realización de *Big Data* es el aprendizaje automático. En este sentido los algoritmos de aprendizaje automático se utilizan para encontrar patrones o analizar datos, por medio de la toma inteligente de decisiones, de manera similar *Big Data* maneja una gran cantidad de datos que pueden contener patrones a descubrir por medio del análisis, la situación que ha popularizado su uso.

Los campos de investigación y desarrollo de aprendizaje automático son diversos, no obstante en la literatura se consideran básicamente tres enfoques de proceso de aprendizaje [13]: aprendizaje supervisado, aprendizaje no supervisado y semi-supervisado. En el aprendizaje supervisado o inductivo, se realiza el entrenamiento con datos de los cuales se conoce a priori la salida deseada o esperada por el clasificador, es decir, los conjuntos de patrones utilizados se presentan al sistema ya etiquetados por un experto humano en el área de estudio [17].

Por otro lado, el aprendizaje no supervisado o deductivo, permite la construcción libre del agrupamiento de los patrones que carecen de las etiquetas de clases, donde en ocasiones no hay información a priori sobre la cantidad de etiquetas, el enfoque no supervisado intenta encontrar una hipótesis útil, basada en únicamente relaciones de similitud en el espacio de representación de los datos [10].

Por último, el aprendizaje semi-supervisado, el cual utiliza una cantidad mínima de objetos etiquetados que son representativos del espacio de representación de los datos combinando con la mayor parte de objetos no etiquetados realiza la clasificación de todos, de está manera se reduce el costo de etiquetado de patrones por un experto humano en el área, mejorando la exactitud del aprendizaje [16].

Para el logro de su cometido, en el aprendizaje automático se encuentra una amplia gama de algoritmos, algunos orientados a tareas de clasificación, regresión o agrupamiento. El contexto de la investigación se centra en los algoritmos de clasificación con paradigma de aprendizaje supervisado. Dentro de estos algoritmos se pueden distinguir árboles de decisión, memorias asociativas, Máquinas de vector soporte, redes neuronales, métodos Bayesianos y muchos más [11].

2. Trabajos relacionados

Existen trabajos comparativos de entornos de trabajo para *Big Data* en los cuales se muestran en resumen características de cada entorno de trabajo, por ejemplo, Inoubil et al. [9] presentan una revisión de los entornos de trabajo más populares y ampliamente usados en *Big Data* (Hadoop y Spark), en el cual se realizó una comparativa considerando el rendimiento para el procesamiento por lotes y el procesamiento en linea.

Por otro lado, Singh et al. [22] proporcionan un análisis de las plataformas Hadoop y Spark para realizar análisis de *Big Data* evaluando las ventajas e inconvenientes de cada plataforma en base a diversas medidas tales como escalabilidad, velocidad de datos, tolerancia a fallas, procesamiento en linea, tamaño

de datos y tareas iterativas. De las plataformas describen los puntos fuertes y desventajas de cada uno.

Landset et al. [12] realizan un trabajo de evaluación de herramientas tales como Spark, Hadoop, Flink, Storm y H_2O para *Big Data* bajo criterios de ventajas y desventajas de cada herramienta, así como también realizan una comparativa de los paradigmas de procesamiento y las librerías de aprendizaje automático que se pueden implementar. Sin embargo, este análisis no incluye un enfoque de instalación e implementación de algoritmos propios.

Otro estudio de tecnologías de código abierto como Spark, Hadoop, Kafka, Scribe, S4, HStreaming, All-RiTE e Impala, para el procesamiento de grandes volúmenes de datos en tiempo real, pero bajo el contexto del algoritmo *MapReduce* [15].

Para cada entorno o herramienta de trabajo bajo el contexto de *Big Data* el cambio de paradigma de programación y manejo de recursos computacionales que cada uno tiene han generado dificultades tales como el incremento de manejo de archivos como es el caso de *MapReduce*, hasta el escalamiento de algoritmos de aprendizaje automático en algún entorno de trabajo.

3. Entornos de trabajo para datos masivos

En esta sección se presenta un resumen de los entornos de trabajo más usados, destacando propiedad clave tales como el modelo de programación, los lenguajes de programación que se pueden emplear, así como el tipo de fuente de datos.

3.1. Apache Hadoop

Desde el año 2008, Hadoop ha sido una tecnología destacada en el área de *Big Data*, fue de los primeros entornos de trabajo que dieron solución a las cuatro $v's$. Es de código libre, desarrollado para ser usado de manera distribuida y escalable, así como también la administración de grandes cantidades de datos [20]. Se divide en dos componentes principales 1. HDFS (Hadoop Distributed File System). 2. Cómputo distribuido, basado en la idea *MapReduce*.

3.1.1. HDFS

HDFS es una implementación de código libre del sistema de archivos distribuido de Google (*Google File System, GFS*), este sistema de archivos se encarga de almacenar los archivos a lo largo del clúster, diseñado para obtener un acceso rápido para grandes archivos o conjuntos de datos grandes, es escalable y tolerante a fallas.

A pesar de las deficiencias detectadas a lo largo de los diferentes estudios [8], el sistema de archivos que maneja es el predilecto para otros entornos de trabajo debido a su funcionamiento y de implementación de código libre.

Como se muestra en la Figura 1 el sistema de archivos HDFS está estructurado por bloques, es decir que HDFS divide los archivos en bloques de tamaño fijo (128 MB), de ser posible HDFS difunde los bloques de un archivo por las

diferentes máquinas que se encuentran en clúster, de este modo se paraleliza la lectura y escritura del archivo, contribuyendo a la rápidez de solo leer o escribir en un sólo disco, sin embargo se aumenta el riesgo de no tener disponible un archivo en caso de falla, HDFS tiene en consideración esto, debido a que mitiga el riesgo al replicar cada bloque de archivos en varias computadoras, siendo un principio básico de la arquitectura [7].

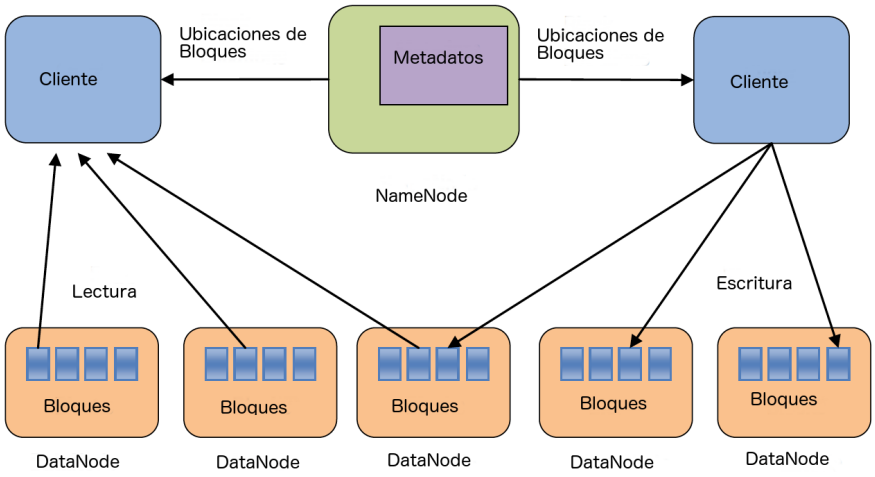

Fig. 1. Arquitectura HDFS.

HDFS introduce dos conceptos, por una parte *NameNode* el cual gestiona el espacio de nombres del sistema de archivos, almacenando en memoria los metadatos para un acceso rápido a estos. Por otro lado, *DataNode* se encarga de almacenar el contenido de un archivo completo en bloques de archivos.

3.1.2. MapReduce MapReduce es un algoritmo de cómputo distribuido, funciona para el procesamiento en paralelo de grandes volúmenes de datos, permitiendo realizar códigos de manera distribuida o paralela. La idea principal es dividir los datos masivos en pequeñas partes y éstas de manera paralela y distribuida generen resultados parciales, los cuales de forma conjunta den una solución global [5]. En general analiza cada registro al leer una entrada [8], cuenta con dos funciones importantes, *Map y Reduce*. La función *Map* considera un par $clave - valor$ de entrada y produce una lista de pares intermedios $clave - valor$, los valores intermedios están agrupados bajo la misma *clave* y son enviados a la función *Reduce*, la cual además de las claves intermedias, recibe el conjunto de valores por clave, estos son mezclados.

3.2. Apache Spark

Cuenta con diversas ventajas sobre el resto de los entornos de trabajo, hoy en día es utilizado por empresas como Yahoo, Baidu, entre otras [8]. De igual manera que Hadoop, es de código libre, para procesos de manera distribuida, basado en el mejoramiento del rendimiento de memoria.

La arquitectura que implementa Spark se basa en el uso de *RDD* (*Resilient Distributed Dataset*) (Figura 2), básicamente es una inmutable colección de objetos por todo el clúster de Spark. Es importante destacar que en Spark se tienen dos tipos de operaciones sobre los RDD, las *transformaciones* que consisten en crear nuevos RDD de uno ya existente aplicando funciones como *map, filter, union y join*, es decir, recorrido de los datos, filtrado de datos, union conjuntos de datos, union a pares respectivamente. Por otro lado, las *acciones*, las cuales son el resultado final de realizar transformaciones a los RDD.

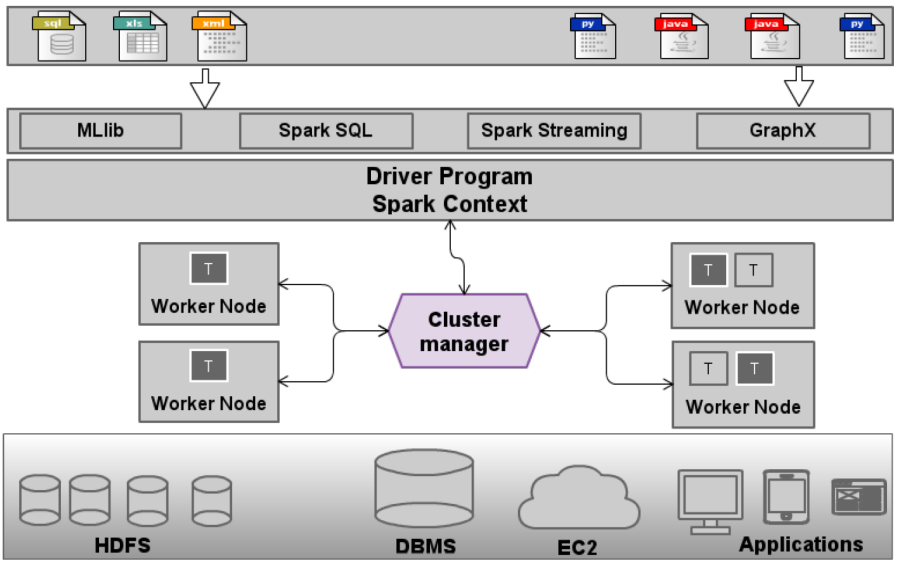

Fig. 2. Arquitectura Spark [8].

Como se muestra en la Figura 2 la arquitectura que presenta Spark está basada en cliente-servidor (maestro-exclavo). El apartado de *Driver Program* es el nodo cliente o esclavo, el cual tiene un objeto de tipo SparkContext, el cual administra la ejecución de las aplicaciones, en tanto que, el *Cluster Manager* gestiona el flujo de la aplicación en cuanto a recursos disponibles. Durante la ejecución de una aplicación en Spark, mantienen contenedores de operaciones denominados *Worker Nodes*. Una de las características que hace atractivo el uso

de Spark es que provee de API's que facilitan la programación e interacción con el entorno de trabajo [8], entre ellas están:

- *Spark Core*: Proporciona la gestión de memoria, así como también un modelo generalizado de ejecución que soporta una variedad de aplicaciones, así como también códigos de Java, Scala y Python.
- *Spark Stream*: Proporciona aplicaciones interactivas y analíticas en transmisión de datos, tolerante a fallas y puede ser usado para diferentes fuentes de datos.
- *Spark SQL*: Permite el manejo de SQL en datos estructurados de diversas fuentes.
- *Spark MLLib*: Proporciona un conjunto de algoritmos de aprendizaje automático, de mayor velocidad en comparación que MapReduce.
- *GraphX*: Es una librería para computo paralelo de gráficas, soporta operaciones sobre gráficas tales como unión de vértices, obtención de subgráficas, entre otras.

3.3. Apache Flink

Flink [2] es un entorno de trabajo de código libre, utilizado para el procesamiento de datos tanto en tiempo real como por lotes, capaz de ser aprovechado por las características de ser distribuido, alto rendimiento, alta disponibilidad y preciso. Una ventaja competitiva es que las aplicaciones pueden mantener una agregación o resumen de los datos procesados, asegurando el estado de una aplicación en caso de falla.

De acuerdo a Inoubil et. al. [8], el modelo de programación empleado en Flink es similar a *MapReduce*, sin embargo se tienen propiedades adicionales como el alto nivel de funciones tales como *join, filter* y *agregation*.

Como se observa en la Figura 3, Flink ofrece una arquitectura compuesta por modos de despliegue ya sean de manera local, en cluster o en la nube (*cloud*), el *core* de Flink es la manera distribuida procesando los datos como un evento a la vez en lugar de una serie de lotes, siendo esta de suma importancia y distintiva con respecto al rendimiento. Por último, en un nivel superior y abstracto se cuentan con las API's y librerías que permiten a los usuarios el computo distribuido y de manera transparente el uso.

Para las API's a nivel de procesamiento en línea, se implementan las transformaciones de flujo de datos (por ejemplo, filtrado, actualización de estado, definición de ventanas, agregación), mientras que para el procesamiento por lotes se aplican transformaciones en conjuntos de datos (por ejemplo, filtrado, mapeo, unión, agrupamiento). La API *Table* indica el uso del lenguaje SQL.

Como se muestra en la Figura 3, se cuenta con una librería de aprendizaje automático denominada *FlinkML* así como también una librería para el procesamiento de gráficas denominada *Gelly*.

De manera similar que Hadoop y Spark, Flink hace uso de sistema de archivos *HDFS*, así como también de archivos locales.

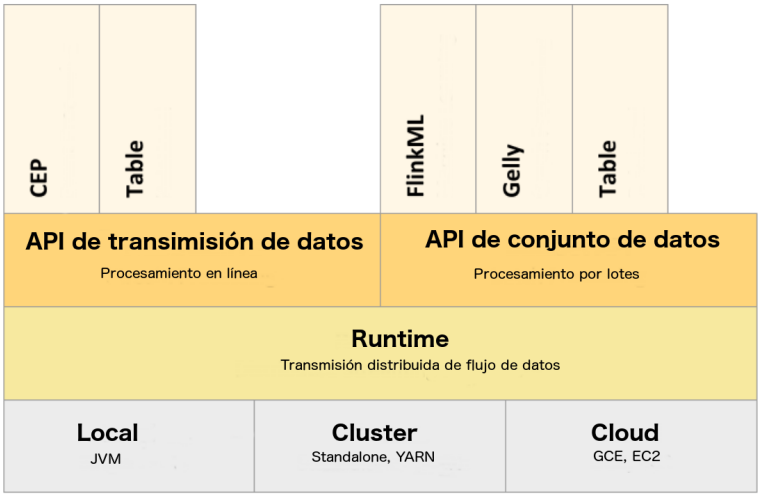

Fig. 3. Arquitectura Flink [2].

4. Análisis de entornos de trabajo

El objetivo de este artículo es presentar una comparativa de características que los entornos de trabajo más usados para *Big Data* tienen en la implementación de algoritmos de aprendizaje automático.

4.1. Características generales

La Tabla 2 muestra las características generales a considerar que tienen los entornos de trabajo en cuanto a lenguajes de programación, formato de datos, procesamiento de datos, entre otros.

Como se observa en la Tabla 2 Spark tiene un mayor abanico de posibilidades en términos de lenguajes de programación, y la inclusión de librerías de aprendizaje automático facilitan la implementación. Debido a que Hadoop, no cuenta con una integración de algoritmos de aprendizaje automático se tiene que incorporar la API necesaria para poder realizar una implementación en Java. Mientras que para Spark, se puede elegir el lenguaje de programación con el que se esté mayormente familiarizado.

Por otro lado, Flink, puede ser una alternativa viable de uso, debido a que cuenta con algoritmos de aprendizaje automático, así como también el modelo de programación aumentado de MapReduce a través de Transformaciones, sin embargo, en estudios como el de Inoubli [8] indica que el rendimiento depende del estado de la red, debido a que los resultados de las funciones MapReduce son enviados a través del cluster.

Tabla 2. Comparativa de entornos de trabajo.

	Hadoop	**Spark**	**Flink**
Formato de datos	clave-valor	RDD	clave-valor
Procesamiento de datos	Por lotes	Por lotes y en línea	Por lotes y en línea
Fuentes de datos	HDFS	HDFS, DBMS y KAFKA	KAFKA, mensajes, archivos
Modelo de Programación	MapReduce	Transformaciones y Acciones	Transformaciones
Lenguajes de Programación	Java	Java, Python y Scala	Java
Librerías de aprendizaje automático	No	Si	Si

4.2. Aprendizaje automático

Como anteriormente se mencionó la implementación de algoritmos propios de aprendizaje automático han sido escalados para dar solución a las características de *Big Data*, en consecuencia, es importante iniciar a considerar los entornos de trabajo que faciliten el uso.

Como se observa en la Tabla 2, Hadoop, Spark y Flink proporcionan una librería de código libre para el uso de algoritmos de aprendizaje automático, no obstante el resto de características hacen atractivo el uso de Hadoop, ya que aún cuando no incluye librerías propias del aprendizaje automático, es posible realizar una integración de código.

En la Tabla 3 se muestra los algoritmos de aprendizaje automático que Spark y Flink implementan. Como puede verse la cantidad de algoritmos de Spark es amplia y variada. Spark en comparación de Flink, cuenta con una mayor gama de algoritmos para implementar aprendizaje automático, adicionalmente, las bondades de distribuir el trabajo, hace que Spark sea una solución viable como herramienta para *Big Data*, permitiendo la implementación de algoritmos propios del entorno de trabajo, así como también el desarrollo e implementación de código.

Es importante resaltar la importancia de la ausencia de librerías de aprendizaje automático en Hadoop, ya que como se menciona anteriormente es un área que apoya fuertemente a *Big Data* en el análisis de datos para toma de decisiones.

Por otro lado, para el pre-procesamiento de datos Flink ofrece las siguientes funciones: Transformador de características polinomiales, el cual mapea un vector en el espacio de característica polinomial de grado d; Estándar de escalas, el cual escala el conjunto de datos, tal que todas las características tengan una medida y varianza especificadas; Escalas *MinMax*, el cual escala el conjunto de datos en un rango especificado por el usuario y *Validación cruzada* con las siguientes estrategias: 1. K-Fold 2. Train-Test 3. Multi-Random

Mientras que Spark, cuenta con una amplia gama de funciones para realizar extracción, transformación y selección de características como por ejemplo TF-

Tabla 3. Aprendizaje automático en entornos de trabajo.

Flink (*FlinkML*)	**Spark (*MLib*)**
▪ Aprendizaje Supervisado • SVM implementando el algoritmo de ascenso de coordenadas duales distribuidas. • Regresión lineal múltiple. ▪ Aprendizaje No Supervisado • KNN	▪ Clasificación: • Regresión logistica • Árbol de decisión. • Random forest. • Perceptron multicapa. • SVM lineal. • Naive Bayes ▪ Regresión: • Regresión lineal. • Árbol de decisión. • Random forest. ▪ Clustering: • K-means • LDA (*Latent Dirichlet allocation*).

IDF, Word2Vec, entre otras, para extracción; Tokenizer, StopWordsRemover, VectorIndexer, Normalizer, entre otras, para transformación y para selección esta VectorSlicer, RFormula.

4.3. Implementación de algoritmos

Parte importante del estudio aquí presentado es el análisis de funcionalidad que los entornos de trabajo ofrecen al momento de su instalación y al incorporar algoritmos desarrollados con código propio para su operación. Para ello, se utilizaron los equipos con las siguientes características: MacBook Pro, procesador Intel Core i5 a 2.6 GHz con memoria RAM de 8 GB DDR3 de 1600 MHz con sistema operativo OS X El capitán y MacBook Pro, procesador Inter Core i5 a 2.4 GHz con memoria RAM de 4 GB DDR3 con sistema operativo OS X Sierra.

La instalación de los entornos de trabajo no son engorrosas, para el sistema operativo de Mac OS, existen manuales y gestores de paquetes que lo facilitan. Sin embargo, para Hadoop en la versión más reciente 3.0.0, los puertos de acceso cambiaron, esto se puede consultar en `https://issues.apache.org/jira/browse/HDFS-9427`, es de suma importancia considerar esto para lograr tener acceso visual al sistema de archivos HDFS. Por otro lado, para desarrollar código, existen consideraciones importantes, debido a que la API que permite realizar trabajos para Hadoop cambió en el uso del objeto *Job*, así como también para el acceso de archivos, los cuales se encuentran en estado *deprecated*.

En cuanto a Spark, la implementación de algoritmos en el lenguaje de programación Python hace que sea entendible, la carga del conjunto de datos es por medio de una instancia del contexto de trabajo de Spark. Sin embargo hay que tener presente el formato del archivo de datos a cargar, debido a que

algoritmos como un árbol de decisión o una red neuronal pueden cargar datos en formato libsvm, pero no limita el uso de otros formatos de archivos de datos. La modificación de ejemplos incluidos en la instalación de Spark es sencilla en la carga de datos, sin embargo para modificar el algoritmo ya no es intuitivo, debido a que se puede integrar el trabajo de Java y Python.

5. Conclusión

El objetivo del estudio aquí presentado es exponer las bondades y facilidades que los entornos de trabajo tienen en la implementación de algoritmos de aprendizaje automático.

Como se mencionó anteriormente, los beneficios que ofrece Spark, hacen que este sea el entorno de trabajo de preferencia, debido a que cuenta con un mayor número de algoritmos de aprendizaje automático ya programados y disponibles para ser usados, así como también el uso de lenguajes de programación como Java, Scala o Python, amplían los beneficios, sin olvidar que por sus características el modelo de programación logra la optimización de recursos en cuanto a memoria, así como también el tipo de procesamiento de datos que ofrece en no sólo realizarlo por lotes, sino que también acepta el procesamiento en línea.

Sin embargo no se puede dejar atrás a Flink, debido a que también cuenta con algoritmos de aprendizaje automático que si bien la gama no es tan amplia, se tienen los más usados en el área. Al igual que Spark, permite el procesamiento de datos por lotes y en línea, siendo parte importante para tratar de dar solución a la velocidad de llegada en los datos. Como linea abierta de estudio, se encuentra la implementación de más algoritmos con los que no cuente Spark, entre ellos mezcla de expertos, reglas de decisión e inclusive el algoritmo KNN.

Otra linea abierta de estudio es realizar pruebas con grandes volúmenes de datos en cada entorno de trabajo realizando un estudio comparativo de optimización de recursos tanto de memoria como de tiempos de ejecución.

Para cada uno de estos entornos de trabajo se necesita conocimiento de las API's que logran la realización del trabajo distribuido de tareas, para lograr un aprovechamiento de recursos y cubrir con las características de *Big Data*, por lo que se requiere un tiempo de estudio para lograr resultados favorables en implementaciones propias de código.

Referencias

1. Amazon: Amazon aws. Obtenido de: https://aws.amazon.com/es/machine-learning/, Último acceso el 27 de Marzo del 2018
2. Apache: Apache flink. Obtenido de: https://flink.apache.org/, Último acceso el 20 de Marzo del 2018
3. Apache: Apache hadoop. Obtenido de: http://hadoop.apache.org/, Último acceso el 12 de Marzo del 2018
4. Apache: Apache spark. Obtenido de: https://spark.apache.org/, Último acceso el 11 de Febrero del 2018

5. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
6. Google: Google cloud. Obtenido de: https://cloud.google.com/?hl=es, Último acceso el 27 de Marzo del 2018
7. Guller, M.: Big data analytics with Spark: A practitioner's guide to using Spark for large scale data analysis. Springer (2015)
8. Inoubli, W., Aridhi, S., Mezni, H., Jung, A.: Big data frameworks: A comparative study. CoRR abs/1610.09962 (2016), http://arxiv.org/abs/1610.09962
9. Inoubli, W., Aridhi, S., Mezni, H., Maddouri, M., Nguifo, E.M.: An experimental survey on big data frameworks. Future Generation Computer Systems (2018), http://www.sciencedirect.com/science/article/pii/S0167739X17327450
10. Jebara, T.: Machine learning: discriminative and generative, vol. 755. Springer Science & Business Media (2012)
11. Kuncheva, L.I.: Using diversity measures for generating error-correcting output codes in classifier ensembles. Pattern Recognition Letters 26(1), 83–90 (2005)
12. Landset, S., Khoshgoftaar, T.M., Richter, A.N., Hasanin, T.: A survey of open source tools for machine learning with big data in the hadoop ecosystem. Journal of Big Data 2(1), 24 (Nov 2015), https://doi.org/10.1186/s40537-015-0032-1
13. L'Heureux, A., Grolinger, K., ElYamany, H.F., Capretz, M.: Machine learning with big data: Challenges and approaches. IEEE Access (2017)
14. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In: Proceedings of the 18th International Database Engineering & Applications Symposium. pp. 356–361. IDEAS '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2628194.2628251
15. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In: Proceedings of the 18th International Database Engineering & Applications Symposium. pp. 356–361. IDEAS '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2628194.2628251
16. Michalski, R.S., Carbonell, J.G., Mitchell, T.M.: Machine learning: An artificial intelligence approach. Springer Science & Business Media (2013)
17. Mitchell, T.M., Carbonell, J.G., Michalski, R.S.: Machine learning: a guide to current research, vol. 12. Springer Science & Business Media (1986)
18. Mohanty, H., Bhuyan, P., Chenthati, D.: Big Data: A Primer. Springer India, 1 edn. (2015)
19. Morabito, V.: Big Data and Analytics: Strategic and Organizational Impacts. Springer International Publishing, 1 edn. (2015)
20. Prasad, B.R., Agarwal, S.: Comparative study of big data computing and storage tools: a review. International Journal of Database Theory and Application 9(1), 45–66 (2016)
21. Qiu, J., Wu, Q., Ding, G., Xu, Y., Feng, S.: A survey of machine learning for big data processing. EURASIP Journal on Advances in Signal Processing 2016(1), 67 (May 2016), https://doi.org/10.1186/s13634-016-0355-x
22. Singh, D., Reddy, C.K.: A survey on platforms for big data analytics. Journal of Big Data 2(1), 8 (Oct 2014), https://doi.org/10.1186/s40537-014-0008-6