Detección de anomalías en grandes volúmenes de datos

Omar Torres-Domínguez[1]
Samuel Sabater-Fernández[2]
Lisandra Bravo-Ilisatigui[3]
Diana Martin-Rodríguez[4]
Milton García-Borroto[5]

Fecha de recepción: 2 de septiembre de 2018
Fecha de aprobación: 28 de noviembre de 2018

Resumen

El desarrollo de la era digital ha traído como consecuencia un incremento considerable de los volúmenes de datos. A estos grandes volúmenes de datos se les ha denominado *big data* ya que exceden la capacidad de procesamiento de sistemas de bases de datos convencionales. Diversos sectores consideran varias oportunidades y aplicaciones en la detección de anomalías en problemas de *big data*. Para realizar este tipo de análisis puede resultar muy útil el empleo de técnicas de minería de datos porque permiten extraer patrones y relaciones desde grandes cantidades de datos. El procesamiento y análisis de estos volúmenes de datos, necesitan de herramientas capaces de procesarlos como Apache Spark y Hadoop. Estas herramientas no cuentan con algoritmos específicos para la detección de anomalías. El objetivo del trabajo es presentar un nuevo algoritmo para la detección de anomalías basado en vecindad para de problemas *big data*. A partir de un estudio comparativo se seleccionó el algoritmo KNNW por sus resultados, con el fin de diseñar una variante *big data*. La implementación del algoritmo *big data* se realizó en la herramienta Apache Spark, utilizando el paradigma de programación paralela MapReduce. Posteriormente se realizaron diferentes experimentos para analizar el comportamiento del algoritmo con distintas configuraciones. Dentro de los experimentos se compararon los tiempos de ejecución y calidad de los resultados entre la variante secuencial y la variante *big data*. La variante *big data* obtuvo mejores resultados con diferencia significativa. Logrando que la variante *big data*, KNNW-Big Data, pueda procesar grandes volúmenes de datos.

Palabras clave: *big data*; detección de anomalías; MapReduce; minería de datos.

Anomalies detection for big data

Abstract

The development of the digital age has resulted in a considerable increase in data volumes. These large volumes of data have been called big data since they exceed the processing capacity of conventional database systems. Several sectors consider various opportunities and applications in the detection of anomalies in big data problems. This type of analysis can be very useful the use of data mining techniques because it allows

[1] Universidad Tecnológica de la Habana "José A. Echeverría" (La Habana, Cuba). otorres@ceis.cujae.edu.cu. ORCID: 0000-0003-4582-7549.
[2] Universidad Tecnológica de la Habana "José A. Echeverría" (La Habana, Cuba). ORCID: 0000-0003-4552-9103.
[3] M.Sc. Universidad Tecnológica de la Habana "José A. Echeverría" (La Habana, Cuba). lbravo@ceis.cujae.edu.cu. ORCID: 0000-0002-8209-4121.
[4] Ph. D. Universidad Tecnológica de la Habana "José A. Echeverría" (La Habana, Cuba). dmartin@ceis.cujae.edu.cu.
[5] Ph. D. Universidad Tecnológica de la Habana "José A. Echeverría" (La Habana, Cuba). mgarcia@ceis.cujae.edu.cu. ORCID: 0000-0002-3154-177X.

extracting patterns and relationships from large amounts of data. The processing and analysis of these data volumes need tools capable of processing them as Apache Spark and Hadoop. These tools do not have specific algorithms for detecting anomalies. The general objective of the work is to develop a new algorithm for the detection of neighborhood-based anomalies in big data problems. From a comparative study, the KNNW algorithm was selected by its results, in order to design a big data variant. The implementation of the big data algorithm was done in the Apache Spark tool, using the parallel programming paradigm MapReduce. Subsequently different experiments were performed to analyze the behavior of the algorithm with different configurations. Within the experiments, the execution times and the quality of the results were compared between the sequential variant and the big data variant. Getting better results, the big data variant with significant difference. Getting the big data variant, KNNW-Big Data, can process large volumes of data.

Keywords: big data; data mining; detecting anomalies; MapReduce.

Detecção de anomalias em grandes volumes de dados

Resumo

O desenvolvimento da era digital tem trazido como consequência um incremento considerável dos volumes de dados. Estes grandes volumes de dados têm sido chamados de *big data* já que excedem a capacidade de processamento de sistemas de bases de dados convencionais. Diversos setores consideram várias oportunidades e aplicações na detecção de anomalias em problemas de *big data*. Para realizar este tipo de análise pode resultar muito útil o emprego de técnicas de mineração de dados porque permitem extrair padrões e relações desde grandes quantidades de dados. O processamento e análise destes volumes de dados, necessitam de ferramentas capazes de processá-los como Apache Spark e Hadoop. Estas ferramentas não contam com algoritmos específicos para a detecção de anomalias. O objetivo do trabalho é apresentar um novo algoritmo para a detecção de anomalias baseado em vizinhança para problemas de *big data*. A partir de um estudo comparativo selecionou-se o algoritmo KNNW por seus resultados, com o fim de desenhar uma variante *big data*. A implementação do algoritmo *big data* realizou-se na ferramenta Apache Spark, utilizando o paradigma de programação paralela MapReduce. Posteriormente realizaram-se diferentes experimentos para analisar o comportamento do algoritmo com distintas configurações. Dentro dos experimentos compararam-se os tempos de execução e qualidade dos resultados entre a variante sequencial e a variante *big data*. A variante *big data* obteve melhores resultados com diferença significativa. Logrando que a variante *big data*, KNNW-Big Data, possa processar grandes volumes de dados.

Palavras chave: *big data*; detecção de anomalias; MapReduce; mineração de dados.

Para citar este artículo:

O. Torres-Domínguez, S. Sabater-Fernández, L. Bravo-Ilisatigui, D. Martin-Rodríguez, and M. García-Borroto, “Detección de anomalías en grandes volúmenes de datos,” *Revista Facultad de Ingeniería*, vol. 28 (50), pp. 62-76, Ene. 2019. DOI: https://doi.org/10.19053/01211129.v28.n50.2019.8793.

I. Introducción

Actualmente muchas compañías en sus sistemas de información registran todas las transacciones que se realizan. La gran cantidad de datos almacenada sobrepasa con creces las capacidades humanas para su procesamiento y análisis manual; limitando las capacidades de detección de fraude en la institución. Una de las soluciones planteadas para el apoyo a la detección de fraude ha sido la identificación de anomalías o datos atípicos para analizar aquellas transacciones de los clientes que no corresponden a lo que habitualmente este hace [1-3]. Para lograr este objetivo se hace necesario la utilización de herramientas informáticas, que permitan identificar dentro de miles o millones de transacciones y registros, patrones de comportamiento que son inusuales y corresponden a actividades potencialmente fraudulentas. En [1] se presenta un breve estudio de las diferentes técnicas que han sido aplicadas. Las técnicas utilizadas se enfocan fundamentalmente en la detección de anomalías [4, 5], las cuales pueden depender de factores como la naturaleza de los datos, la disponibilidad de los datos etiquetados y el tipo de anomalías que se desee detectar.

Cuando los datos no están etiquetados se han utilizado técnicas descriptivas basadas en algoritmos de agrupamiento [6, 7] y técnicas basadas en vecino más cercano . Dentro de las técnicas basadas en el vecino más cercano [8], se encuentran los algoritmos cuyo cálculo del índice de anomalía se basa en el cálculo de las distancias y en el de las densidades. El incremento del uso de la tecnología representa un reto para los algoritmos tradicionales de detección de anomalías. El reto se debe a que los datos se han caracterizados por el incremento de su volumen, la velocidad con la que se generan y la variedad en su estructura y formato. Estas tres características continúan creciendo, lo que imposibilita que sistemas tradicionales almacenen y procesen los datos. No existe un tamaño específico que determine los datos en big data. A partir de lo anterior surge como concepto “*big data*”. La fortaleza de *big data* radica en la capacidad de procesar grandes volúmenes de información en un tiempo razonable [9-12].

Entre las diversas herramientas que aplican la minería de datos y ejecutan sus técnicas sobre grandes volúmenes de información se encuentra Spark. Una plataforma de clúster de computadoras diseñada para ser rápida y de propósito general [13-15]. La cual no presenta algoritmos para la detección de anomalías. En este trabajo se propone un diseño MapReduce de un algoritmo de detección de anomalías para grandes volúmenes de datos. Por esta razón, el resto del trabajo está organizado de la siguiente manera. En la sección 2 se aborda toda la explicación referente a las diferentes técnicas de detección de anomalías y fu funcionamiento además de un estudio comparativo de los diferentes algoritmos de detección de anomalías basados en vecindad. En la sección 3 se demuestra un diseño MapReduce para la detección de anomalías en grandes volúmenes de datos donde se explican las diferentes etapas del algoritmo propuesto. Posteriormente, en la sección 4 se exponen los resultados obtenidos de la comparación del algoritmo para *big data* y el algoritmo secuencial para detección de anomalías. Finalmente, en la sección 5 se muestran las principales conclusiones alcanzadas en este trabajo.

II. Técnicas de detección de anomalías

Dentro de las técnicas de detección de anomalías puntuales se encuentran las basadas en agrupamiento, en el vecino cercano, en estadísticas y en la teoría de la información. Las técnicas de detección de anomalías basadas en la vecindad plantean que las instancias normales ocurren en vecindades densas, mientras las anomalías ocurren lejos de sus vecinos más cercanos. Estas técnicas requieren de una distancia o una medida similar definida entre dos instancias de datos [4, 5]. La distancia que se emplean en los algoritmos de vecinos más cercanos es la distancia Euclidiana [4, 5].

Las técnicas de detección de anomalías basadas en la vecindad se pueden agrupar en dos categorías:

- Técnicas que emplean las distancia a su k vecino cercano como índice de anomalía.
- Técnicas que determinan el índice de anomalía a partir del cálculo de densidad de cada instancia.

El índice de anomalía es una puntuación o valor numérico que se le asigna a cada instancia de los datos en dependencia del grado de anomalía en que es considerada dicha instancia[4, 5]. Por lo que la salida de las técnicas de detección de anomalías basada en la vecindad es un listado de posiciones de acuerdo con el índice de anomalía. Para ambos casos se define una etapa inicial común que trata de determinar la k vecindad de cada instancia. La k vecindad está dada por la k_distancia del objeto [4, 5] . La cual cumple con la siguiente definición:

Definición 1. Dado un conjunto de datos D la $k\_distancia$ de un objeto p para cualquier entero positivo k, es definida como la distancia $d(p,o)$ entre p y el objeto $o \in D$. Esta distancia debe cumplir las siguientes restricciones [16]:

- Para al menos k objetos $o' \in D \backslash \{p\}$ se mantiene que la $d(p,o') \leq d(p,o)$.
- Para lo sumo $k-1$ objetos $o' \in D \backslash \{p\}$ se mantiene que la $d(p,o') < d(p,o)$.

Por lo que la $k\_distancia$ de la vecindad de p no es más que aquella que se define a continuación:

Definición 2. Dada la *k_distancia* de *p*, la *k_vecindad* de p contiene cada objeto cuya distancia desde p no es mayor que la *k_distancia*. De esta forma los objetos q son llamados los k vecinos más cercanos de *p*.

$$S_k(p) = \{q \in D \backslash \{p\} | d(p,q) \leq k\_distancia(p)\} \quad (1)$$

La Figura 1 muestra la k vecindad para la instancia P definida por *k_distancia(P)=d (P, O_1)* para k=4. Donde O_1, O_2, O_3 y O_4 son los k vecinos cercanos de P. En el caso que el objeto presenta duplicados se puede incorporar una nueva restricción en la que debe haber como mínimo k objetos con diferentes coordenadas espaciales. Esto se debe a que los duplicados pueden traer consecuencias graves para el cálculo de las densidades como la obtención de valores infinitos.

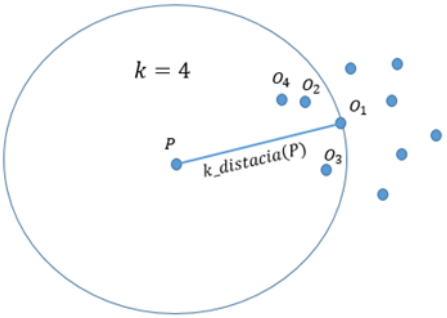

Fig. 1. Ejemplo de la k-distancia para k=4.

Para el cálculo del índice de anomalía basado en la densidad se realizan los siguientes pasos:

1. Se determina la densidad de cada instancia.
2. Se obtiene el índice de anomalía a raíz de la comparación de la densidad de la instancia con las densidades de los otros vecinos cercanos.

Para el caso en que el índice está basado en la distancia:

- Se determina el índice de anomalía con una operación sobre las distancias de la instancia a sus vecinos cercanos.

Ejemplo de ello es el KNNW que el índice de anomalía de una instancia es la suma de las distancias a sus k vecinos cercanos. Estudio comparativo de los algoritmos de detección de anomalía basada en vecindad

Las comparaciones se realizaron con distintas configuraciones de los algoritmos y empleando distintas bases de datos clasificadas en anómalo o normal. Los algoritmos se compilaron con distintos valores para la búsqueda del vecino cercano con k = {1,2,3,4,5,6,7,8,9,10,15,20}. Las bases de datos que se emplearon para la compilación de cada algoritmo presentan atributos numéricos y un atributo que es la etiqueta que clasifica al objeto como anómalo o normal. Estas bases de datos se describen en la Tabla1.

Tabla 1. Bases de datos utilizadas para la comparación de los algoritmos secuenciales.

Bases de datos	Cantidad de objetos	Cantidad de objetos anómalos	Cantidad de objetos normales
Aloi-unsupervised-ad	49534	1508	48026
Annthyroid-unsupervised-ad	6845	250	6595
Breast-cancer-unsupervised-ad	365	10	357
Letter-unsupervised-ad	1598	100	1498
Pen-global-unsupervised-ad	809	90	719
Pen-local-unsupervised-ad	6724	10	6714
Satellite-unsupervised-ad	5100	75	5025
Shuttle-unsupervised-ad	46464	878	45586
Speech-unsupervised-ad	3686	61	3625

Para evaluar cada resultado arrojado por los algoritmos por cada valor de k y la comparación entre ellos se empleó la métrica de evaluación área bajo la curva ROC. La comparación de los resultados se realizó con la herramienta Keel utilizando el procedimiento esta dístico Friedman 1xN y los métodos post hoc Holm y Finner. La Tabla 2 muestra los lugares que toman los algoritmos de acuerdo con las pruebas de Friedman, y se destaca que el KNNW es el algoritmo con mejor resultados.

Tabla 2. Promedio del lugar obtenido por los algoritmos en las pruebas de Friedman.

Algoritmo	Lugar
KNNW	4.1019
KNNO	4.6713
SLOF	4.8472
LoOP	5.2778
LOF	5.3704
INFLO	5.6065
ODIN	5.6204
COF	6.0556
LDOF	6.7407
LDF	7.5046
KDEOS	10.2037

En la Tabla 3 se puede apreciar la comparación del algoritmo KNNW con respecto a los restantes algoritmos empleando la prueba estadística de Friedman, en la que se fija un α=0.05. En la columna marcada con “P” se muestra el p-valor ajustado para determina si existe diferencia significativa. En el caso del KNNW existe una diferencia significativa con los algoritmos cuyo valor P sea menor que 0.05. Por lo que los algoritmos que se comportan semejante al KNNW son el SLOF y el KNNO. A partir de los resultados de las pruebas se determina emplear el KNNW como base para la detección de anomalías en *big data.*

Tabla 3. Comparación post hoc para α = 0.05 con respecto al algoritmo KNNW.

Lugar	Algoritmo	P	Holm	Finner
10	KDEOS	0	0.005	0.005116
9	LDF	0	0.005556	0.010206
8	LDOF	0	0.00625	0.01527
7	COF	0.000015	0.007143	0.020308
6	ODIN	0.000767	0.008333	0.025321
5	INFLO	0.000857	0.01	0.030307
4	LOF	0.004945	0.0125	0.035268
3	LoOP	0.009176	0.016667	0.040204
2	SLOF	0.098641	0.025	0.045115
1	KNNO	0.207061	0.05	0.05

III. Diseño MapReduce para la detección de anomalías en big data

La información almacenada que genera una empresa o un sistema informático crece exponencialmente. Esta información almacenada puede contener

conocimientos valiosos que le permitan a la entidad dueña obtener ventajas. Pero cuando los datos almacenados crecen día a día exponencialmente, implica un reto en el procesamiento y extracción de conocimiento para los sistemas informáticos. Las herramientas tradicionales que comúnmente analizan la información no son capaces de trabajar con grandes volúmenes de datos. Estos datos son conocidos actualmente con el término *big data.* Se definen como *big data* a datos cuyo volumen, diversidad y complejidad requieren nueva arquitectura, técnicas, algoritmos y análisis para gestionar y extraer conocimiento.

A raíz de esta situación surgen diversas herramientas y framework que proponen una solución a esta problemática. Una de las soluciones más populares para abordar algunos problemas de *big data* es MapReduce, un modelo de programación utilizado por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras. Este paradigma procesa datos estructurados en pares clave-valor y consta de dos fases: map y reduce.

En términos generales, los datos se dividen en sub-problemas más pequeños que son distribuidos por los nodos de un clúster para ser procesados de forma independiente en la fase Map. Las soluciones parciales obtenidas por cada proceso Map se combinan de alguna forma en la fase Reduce para obtener una solución. Muchas son las plataformas que han adoptado MapReduce como una solución para problemas de big data, como ejemplo se encuentran Hadoop y Apache Spark.

A. Diseño MapReduce del algoritmo KNNW

En esta sección se muestran y describen las distintas fases del diseño del algoritmo KNNW para problemas de *big data*. El algoritmo KNNW mantiene la estructura de los algoritmos basado en distancias planteados anteriormente. La función que calcula el índice de anomalía del KNNW no es más que las sumas de las distancias de cada objeto a sus k vecinos más cercanos como se muestra a continuación:

$$IA = \sum_{i}^{n} d_i \qquad (2)$$

El diseño MapReduce del algoritmo KNNW consta de dos fases. La primera fase se encarga del particionado de los datos y del cálculo de los valores de anomalía local (con respecto a los puntos de su partición) de todas las tuplas que se analizan. Como muestra la Figura 2 inicialmente distribuyen los datos en distintas particiones. Posteriormente a cada partición se le aplica una función map para obtener de cada elemento de la partición las distancias de sus k vecinos más cercanos en la partición. Luego se realiza un map para obtener el índice de anomalía normalizado. El resultado final de esta fase son los elementos como llave y el índice de anomalía lo mismos.

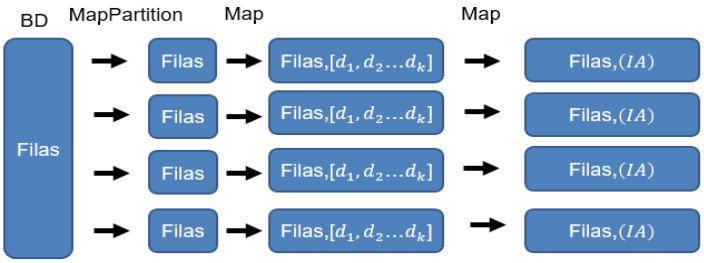

Fig. 2. Primera fase particionado y cálculo de índice de anomalía.

Un índice de anomalía elevado luego de la primera fase puede indicar dos cosas: que estos objetos son anómalos o que en su partición no se encontraban sus vecinos cercanos. Es por ello que en la segunda fase se seleccionan el P porciento de los objetos con mayor valor de anomalía de cada partición (generalmente el 1%) para reajustar el índice de anomalías. De esta manera se pueden evitar gran cantidad de falso positivos y es el objetivo principal de la segunda fase del algoritmo. Para ello se diseña una etapa MapReduce calcula las distancias a todos los objetos de la base de datos de los elementos seleccionados en cada partición y luego determina los vecinos cercanos a estos.

Por último, se realiza una etapa de map para calcular los nuevos valores de anomalía de los elementos seleccionados al principio de la segunda fase. De esa forma aquel objeto que no es anómalo corrige su índice de anomalía y el que es anómalo su índice quedara ajusto al valor real. Los resultados del reduce se combinan con el resto de los elementos que no fueron filtrados en la primera etapa y se devuelve la lista ordenada de los objetos con respecto a su valor de anomalía. En las Figuras 3 y 4 se muestra este proceso.

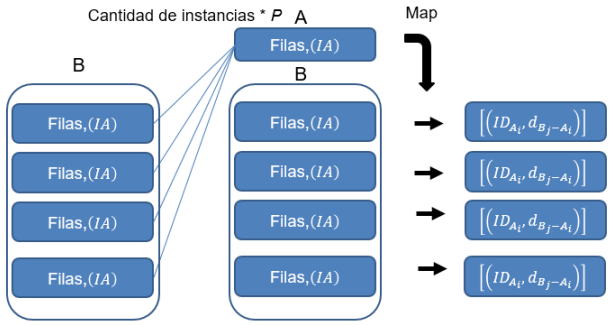

Fig. 3. Selección de los P objetos anómalos en la segunda fase.

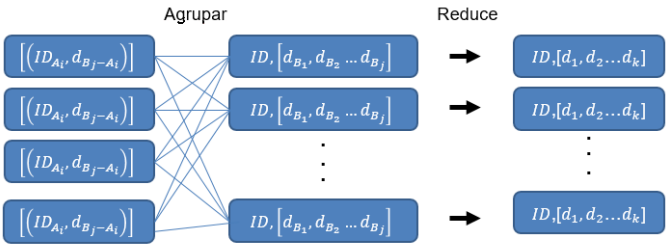

Fig. 4. Agrupamiento y Reduce del final de la segunda fase del algoritmo.

B. Análisis de la eficiencia y eficacia del diseño MapReduce para KNNW - Diseño de los experimentos

En este caso de documento se aborda cómo se comporta la implementación *big data* del algoritmo KNNW frente a su versión secuencial. El diseño los experimentos se encuentran en la Tabla 4, la cual se muestra a continuación:

Tabla 4. Diseño de los experimentos para el caso de estudio.

Base de datos	Parámetro
Kdd99	Este experimento es para probar la efectividad y eficiencia del algoritmo big data respecto al secuencial. Parámetros K=10, P=0.01. Se define cada partición de datos para un total de 25000 elementos.
Dataset de Operaciones de una Empresa de Telecomunicaciones	Utilizando datos reales de operaciones de una Empresa de Telecomunicaciones se ejecutaron varias pruebas sobre el algoritmo *big data*. Parámetros K=10, P=0.01. Se define cada partición de datos para un total de 25000 elementos.

Para validar los resultados se realizaron experimentos con una base de datos clasificadas del Kdd99 de 500000 transacciones aproximadamente. Esta base de datos se divide en distintos subconjuntos de 100000, 200000, 300000 y 400000 manteniendo para cada subconjunto la cantidad de instancias anómalas. Donde se compara el tiempo de ejecución y la eficiencia de cada algoritmo para las distintas cantidades de datos. Para los algoritmos se definió una k = 10. En caso del KNNW_Big Data se establece por ciento para la segunda fase igual a 0.01 y para cada aumento de los datos en 100000 se aumenta la cantidad de particiones en 4.

IV. Resultados experimentales

A. Comparación entre KNNW-Big Data con KNWW secuencial

Para los algoritmos KNNW y KNNW-Big Data se asignó para la variable de entrada K=10 Y p=0.01 la cual representa a los K vecinos más cercanos. Los experimentos de los algoritmos se realizaron con un total de 40 cores y 40 GB de memoria RAM. En caso del algoritmo secuencial los experimentos se realizaron en una PC donde se utilizó 1 core y 8 GB RAM.

A demás se muestran en la Tabla 5 las características de la base de datos que se empleó para los posteriores experimentos. Esta base de datos se extrajo del sitio web de la universidad de Harvard. Esta base de datos cuenta con una etiqueta que clasifica si a una instancia en normal y anómala. Esta etiqueta es el único atributo nominal que presenta y que para la comparación entre los algoritmos de los algoritmos se excluyó.

Tabla 5. Base de datos original.

BD	Cantidad de instancias	Cantidad de atributos	Atributos numéricos reales	Atributos numéricos enteros	Atributos nominales
Kdd99	578764	31	16	14	1

El propósito del primer experimento es evaluar la calidad de los resultados y el tiempo de ejecución de los algoritmos KNNW y KNNW-Big Data a medida que

aumenta el volumen de los datos. Para ello se dividió en distintos subconjuntos la base de datos clasificada kdd99. A demás se definió para cada partición de datos de la variante KNNW-Big Data, alrededor de 25000 instancias a procesar en cada iteración de las distintas bases de datos. Partiendo de 4 particiones para 100000 instancias hasta 20 map para la base de datos original de 578764 instancias. En la Tabla 6 se muestra la calidad de los resultados obtenidos por ambos algoritmos, donde a mayor valor mejor calidad obtenida. En la tabla se puede apreciar como la variante big data obtiene mayor área bajo la curva que el KNNW para cada base de datos. Esto se debe a que las instancias anómalas que no pasaron a la segunda fase del algoritmo obtuvieron un mayor valor de índice de anomalía. Ocasionando que las anomalías ocuparan un mejor puesto en la lista de posiciones. El alto índice de anomalía se debe a que en las particiones donde se encuentran las instancias, no aparecen sus vecinos cercanos o algunos de ellos.

Tabla 6. Área bajo la curva ROC conforme aumenta el número de instancias a procesar.

BD	Particiones para KNNW-Big Data	KNNW-Big Data	KNNW
Kdd99_100000	4	0.288	0.219
Kdd99_200000	8	0.468	0.260
Kdd99_300000	12	0.833	0.286
Kdd99_400000	16	0.721	0.321
Kdd99_500000	20	0.799	0.368

En la Figura 5 se muestra la relación entre el área bajo la curva ROC y el número de instancias para los algoritmos KNNW y KNNW-Big Data en cada base de datos. Donde se puede apreciar una mejor representación de la comparación de ambos algoritmos con respecto al área bajo la curva ROC. El área del algoritmo KNNW-Big Data aumenta a la par que se incrementan la cantidad de instancias. Pero disminuye para la base de datos kdd99_400000, lo que se puede interpretar que las nuevas instancias normales que se incorporaron pueden presentar un mayor índice de anomalía que instancias anómalas. Esto se debe a que en las particiones de estas instancias no se encuentran sus vecinos cercanos. Además, su índice de anomalía puede no ser ajustado en la segunda fase porque no fueron seleccionados. Lo que influye en la disminución del área bajo la curva ROC.

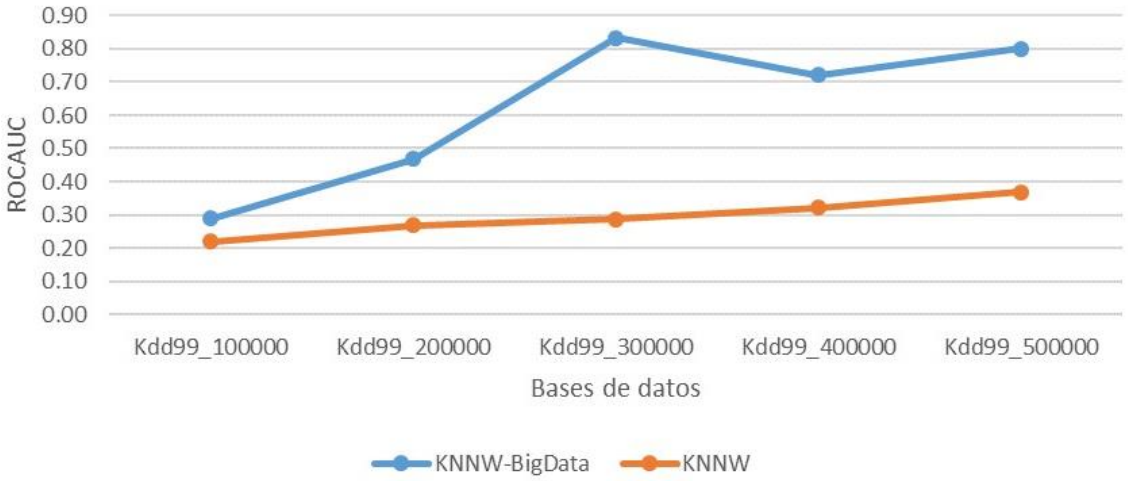

Fig. 5. Relación entre el área bajo la curva ROC y el número de instancias para los algoritmos KNNW y KNNW-Big Data.

La Tabla 7 describe los resultados de ambos algoritmos con respecto al tiempo de ejecución en cada base de datos. Donde se puede observar a primera vista que los resultados del algoritmo KNNW-Big Data ofrecen menores tiempos de ejecución que los resultados del KNNW. También se aprecia que a medida que aumenta la cantidad de registros, los tiempos de ejecución en ambos algoritmos se incrementan. Por lo que los tiempos de ejecución se corresponden con la cantidad de instancias de las bases de datos de forma proporcional. Se muestran valores de medidas de rendimiento del tiempo de ejecución del algoritmo big data respecto al algoritmo KNNW.

A pesar de que ambos algoritmos no sean equivalentes, se utilizó al KNNW como referencia para determinar los valores de medida de esta tabla. Con el fin de tener una visión más amplia del comportamiento del tiempo de ejecución del algoritmo KNNW_Big Data. Una de las medidas que se describe es la aceleración, que demuestra que tan rápido es el KNNW_Big Data, llegando a ser hasta 31 veces más rápida. Por último, se observa la eficiencia del algoritmo MapReduce, demostrando hasta un 155% del uso de los recursos de los procesadores.

Tabla 7. Tiempo de ejecución conforme aumenta el número de instancias.

BD	Particiones de KNNW-Big Data	Tiempo KNNW-Big Data	Tiempo KNNW	Aceleración	Eficiencia
Kdd99_100000	4	00:03:12	00:10:10	3.17	0.79
Kdd99_200000	8	00:06:13	00:35:56	5.77	0.72
Kdd99_300000	12	00:05:12	01:22:16	15.81	1.31
Kdd99_400000	16	00:07:01	02:33:24	21.82	1.36
Kdd99_500000	20	00:08:56	04:38:06	31.08	1.55

En el caso de la Figura 6 se representa la relación grafica entre el tiempo de ejecución en milisegundos y el número de instancias para los algoritmos KNNW y KNNW-Big Data que se muestran en la Tabla 7. En esta representación se puede observar como el crecimiento del tiempo del algoritmo KNNW tiende a ser

acelerado. Por otra parte, se puede comprobar como la diferencia de tiempo de ejecución entre KNNW-Big Data y KNNW es mayor conforme el tamaño de los datos aumentan aumenta.

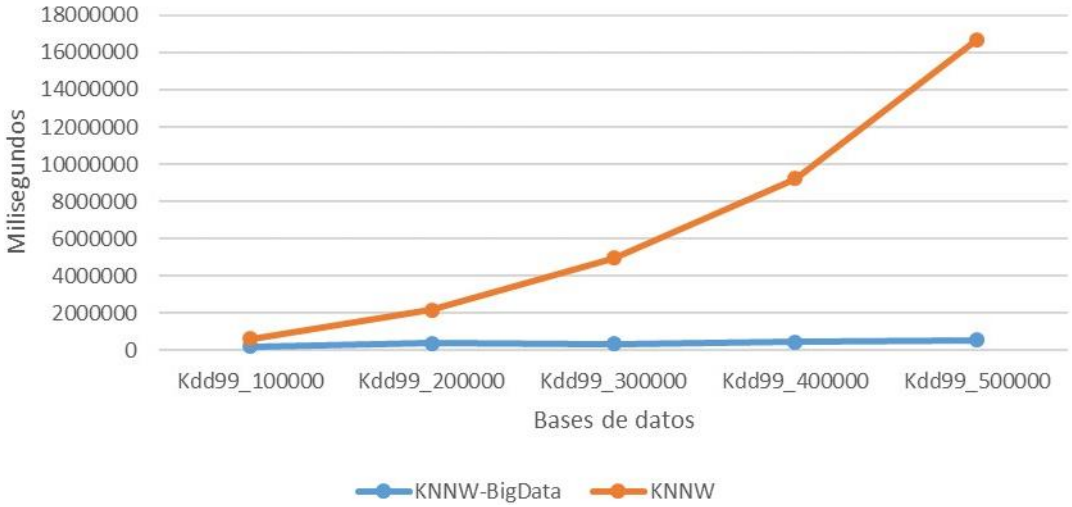

Fig. 6. Relación entre el tiempo de ejecución y el número de instancias para los algoritmos KNNW y KNNW-Big Data.

B. Experimentos con datos de una compañía de Telecomunicaciones

Para este experimento con KNNW-Big Data se asignó para la variable de entrada K=10 Y p=0.01 la cual representa a los K vecinos más cercanos. Las características de los recursos utilizados es un clúster de computadoras que está montado sobre el sistema operativo CentOS7 sobre el cual se encuentra instalado DCOS 1.9. Respecto a recursos se posee 64 cores y 40GB RAM, de los cuales están disponible 32 cores y 20 GB de RAM.

Se utilizó como fuente de datos primaria aproximadamente 500000 transacciones de operaciones de la compañía, la Tabla 8 muestra los datos primarios utilizados

Tabla 8. Datos primarios de la compañía de Telecomunicaciones.

BD	Cantidad de instancias	Cantidad de atributos	Atributos numéricos reales	Atributos numéricos enteros	Atributos nominales
Operaciones	450236	7	3	3	1

Se realizó un experimento sobre las transacciones de la compañía que tenían presente múltiples operaciones, para un total de 450236 instancias. Los datos se cargaron desde un sistema de archivos distribuidos y el algoritmo se configuro con el valor de k=10 y un umbral del 0.01.

Para las 450236 instancias analizadas se determinaron 513 anomalías en un tiempo de 25 minutos aproximadamente (1519 segundos). La calidad del algoritmo fue validada por un experto en el problema donde se obtuvo como resultado que:

- Las 10 instancias más anómalas obtenidas por el algoritmo coincidían con instancia de elementos que eran sospechosos de realizar operaciones anómalas y estaban marcados como posibles instancias fraudulentas.
- En las primeras 100 instancias de las anomalías obtenidas se determinó un nivel de efectividad de aproximadamente el 70%.
- Se obtuvieron más instancias anómalas por el algoritmo, pero quedaron pendiente de verificar su estado anómalo por un experto debido a que se desconocía el comportamiento de dichas instancias.
- El algoritmo demostró alta eficacia y eficiencia en el análisis de los datos de operaciones.

C. Optimización del algoritmo

Para la mejora del algoritmo KNNW-Big Data respecto a los tiempos de ejecución y de la capacidad del tamaño de datos que se podía analizar se realizaron diferentes cambios sobre el mismo para mejorar sus tiempos de ejecución.

1. Después de cada fase de map del algoritmo se especificó un nivel de persistencia en disco y memoria.
2. Se cambiaron las estructuras de datos necesarias a arreglos que posean un tiempo de acceso constante a cada elemento en la estructura.
3. Se determinó que ejecutar el algoritmo con menor cantidad de datos por partición disminuye significativamente el tiempo de ejecución, se recomienda aproximadamente 10000 elementos por partición.

Con los nuevos cambios y los datos de la compañía también se realizaron pruebas de rendimiento donde se replicaron los datos originales hasta alcanzar los 15 millones de transacciones. El experimento de rendimiento utilizo K=10, P=0.01 y el particionado se determinó un total de 10000 elementos por partición. La siguiente tabla muestra para cada replica de datos realizada los tiempos de ejecución alcanzados por el algoritmo.

Tabla 9. Resultados obtenidos en el experimento de rendimiento con los datos de la compañía.

Cantidad de datos de la replica	Cantidad de particiones	Tiempo de ejecución
1 000 000	100	10 minutos (624 segundos)
2 000 000	200	23 minutos (1403 segundos)
5 000 000	500	1hora 32 minutos (5564 segundos)
10 000 000	1000	3 horas 43 minutos (13418 segundos)
15 000 000	1500	5 horas 57 minutos (21475 segundos)

La tabla anterior nos permite observar que las mejoras realizadas al algoritmo big data anteriormente expuestas nos permite procesar aproximadamente 4 veces más datos que la versión inicial del algoritmo *big data*. Inicialmente con 450 000 instancias tardaba 25 minutos y con las mejoras analiza 2 millones en 23 minutos lo que significa una mejora de 4 veces la velocidad de análisis.

V. Conclusiones

Con los experimentos anteriores se confirma que la versión big data obtiene mejores resultados que la versión secuencial en cuanto a calidad de los resultados y tiempo de ejecución. Esta versión big data también permite el análisis de volúmenes de datos que anteriormente era imposible su análisis en

un ordenador común. Con los resultados obtenidos se mejoró el tiempo de ejecución del algoritmo KNNW. Para ello se debe tener en cuenta el valor del umbral que puede afectar en el tiempo de ejecución del algoritmo, así como en la calidad de los resultados. Dado que el umbral puede dejar sin reajustar si se selecciona un valor muy bajo a posibles candidatos anómalos, por eso se recomienda utilizar 0.01. El algoritmo propuesto demuestra que en entornos reales tiene una alta eficacia en la detección de anomalías en datos no etiquetados y que posee un tiempo de ejecución acorde al volumen de datos que se analiza. En los experimentos realizados se obtuvo como resultado que especificar 10000 elementos por partición mejoraba considerablemente los tiempos de ejecución del algoritmo sin incidir en la calidad de los datos.

Referencias

[1] R. Bolton, and D. Hand, "Statistical fraud detection: A review," *Statistical science,* pp. 235-249, 2002.

[2] K. Chitra, and B. Subashini, "Data mining techniques and its applications in banking sector," *International Journal of Emerging Technology and Advanced Engineering,* vol. 3, pp. 219-226, 2013.

[3] S.-H. Li, D. C. Yen, W.-H. Lu, and C. Wang, "Identifying the signs of fraudulent accounts using data mining techniques," *Computers in Human Behavior,* vol. 28 (3), pp. 1002-1013, May. 2012. DOI: https://doi.org/10.1016/j.chb.2012.01.002.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR),* vol. 41 (3), pp. 1-15, Jul. 2009. DOI: https://doi.org/10.1145/1541880.1541882.

[5] J. Zhang, "Advancements of outlier detection: A survey," *ICST Transactions on Scalable Information Systems,* vol. 13 (1), pp. 1-26, Feb. 2013. DOI: https://doi.org/10.4108/trans.sis.2013.01-03.e2.

[6] L. M. Cruz-Quispe, and M. T. Rantes-García, "Detección de fraudes usando técnicas de clustering," 2010.

[7] M. Vadoodparast, and A. R. Hamdan, "Fraudulent Electronic Transaction Detection using dynamic KDA model," *International Journal of Computer Science and Information Security,* vol. 13, p. 90, 2015.

[8] M.Zhang, J.Salerno, and P.Yu, "Applying data mining in investigating money laundering crimes," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 747-752. DOI: https://doi.org/10.1145/956750.956851.

[9] B. Baesens, *Analytics in a big data world: The essential guide to data science and its applications*. New Jersey: John Wiley & Sons, 2014.

[10] J. Coumaros, S. D. Roys, L. Chretien, J. Buvat, S. KVJ, V. Clerk*, et al.*, "Big Data Alchemy: How can Banks Maximize the Value of their Customer Data?," Capgemini Consulting, 2014.

[11] V. Mayer-Schönberger, and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. New York: Houghton Mifflin Harcourt, 2013.

[12] N. Marz, and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[13] S. Ryza, U. Laserson, S. Owen, and J. Wills, *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*. O'Reilly Media, Inc., 2015.

[14] H. Karau, *Fast Data Processing with Spark*: Packt Publishing Ltd, 2013.

[15] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning spark: lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

[16] M. Breungi, P. Kriegel, R. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *ACM sigmod record*, 2000, pp. 93-104. DOI: https://doi.org/10.1145/335191.335388.