DISEÑO DE EXPERIMENTOS

Análisis Apriori vs. BruteForce vs. Clustering

1. Planeación y realización

Allers Group requiere hacer un análisis sobre todos sus datos almacenados, aplicando técnicas de DataMining con el objetivo de explorar sus registros de ventas de forma semiautomática, encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto, y con base en estos resultados realizar proyecciones a futuro, mejorar su toma de decisiones y aumentar sus ventas.

Se considera fundamental que el tiempo en que tarda el programa en generar estas tendencias, patrones y dependencias sea mínimo, ya que sus resultados deben pasar por un proceso de análisis y aprobación por parte de un grupo de analistas de la misma empresa, por lo cual se requiere implementar los algoritmos que tengan bajo tiempo de respuesta y alta relevancia (y exactitud en lo posible) en sus resultados.

1.1. Fase 1: Delimitación del problema

La delimitación debe establecerse los límites de la investigación en términos de espacio, tiempo, universo y del contenido. Según el detalle:

Delimitación Espacial.
La investigación se va a realizar en instalaciones de la Universidad Icesi. Utilizando computadores portátiles de cada integrante.

Delimitación Temporal.
El periodo seleccionado para realizar la investigación es de 4 días.

Delimitación del Universo.
Se deben definir la variable de respuesta, los factores (estudiados, no controlables, controlables).

La población de interés es el conjunto de datos transaccionales de la empresa Allers Group, que incluya datos de clientes, transacciones y productos.

La muestra que se selecciona son los datos recopilados durante los últimos 6 meses.

1.2. Fase 2: Elección de las variables de respuesta

La variable de respuesta para este experimento es el tiempo en que el algoritmo tarda en generar los determinados itemset ya que el cálculo de dependencias lo genera un solo algoritmo a partir de estos conjuntos de ítems. Esta es la variable se ve directamente afectada por la cantidad de ítems que se encuentran en la base de datos.

1.3. Fase 3: Determinación de los factores

1.3.1 Factores estudiados

Los factores estudiados que influyen en la variable de respuesta son:

- Número ítems con los que se va a trabajar.
- Algoritmo utilizado.

1.3.2 Factores no controlables

Los factores que no pueden ser controlados dentro de este experimento y, por tal, alteran el resultado de las pruebas, son las características propias del equipo o computador donde se ejecutan las pruebas, entre estas se encuentran:

- Fragmentación de la memoria: disco duro y memoria principal.
- Tareas realizadas en paralelo con la prueba.

1.3.3 Factores controlables

Los factores que se pueden controlar dentro del presente experimento son:

- Cantidad de transacciones que se dejaran constantes durante el experimento
- Número de elementos de entrada
- Orden inicial de los elementos de entrada
- Lenguaje de programación
- Sistema operativo
- Tamaño de disco del procesador
- Memoria RAM
- Núcleos de equipo

1.4. Fase 4: Niveles de cada factor y diseño del experimento

Se utilizará la cantidad de ítems como unidad experimental. Para cada cantidad especificada de ítems se tomarán 1000 muestras con los tiempos de cada uno de los algoritmos en este caso Apriori, BruteForce y Clustering, los ítems que se utilizarán para las pruebas serán los proporcionados por Allers Group al igual que las transacciones

Para el experimento se generarán itemset con un porcentaje de aparición mayor o igual a 95% y similitud de 80%, cabe resaltar que se hará uso de todas las transacciones (21843) para cada medición de tiempo.

1.4.1 Niveles

Número de elementos de entrada
10
20
30
40
50

Tabla 1.1: Niveles de entrada

Algoritmos
Apriori
BruteForce
Clustering

Tabla 1.2: Niveles de los Algoritmos

1.4.2 Tratamientos

Algoritmo	Nivel de elementos	Tratamiento
Apriori	10	1
	20	2
	30	3
	40	4
	50	5
BruteForce	10	6
	20	7
	30	8
	40	9
	50	10
Clustering	10	11
	20	12
	30	13
	40	14
	50	15

Tabla 1.3 Definición y separación de los Tratamientos

1.4.3 Diseño de pruebas del experimento

Clase: Apriori			
Caso #	Descripción de la prueba	Valores de entrada	Resultado
1	Dado 20 Ítems se calcula el tiempo en que tarda el algoritmo Apriori en generar todos los itemsets frecuentes	Itemsets = 20 transactions = 21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Apriori
2	Dado 40 Ítems se calcula el tiempo en que tarda el algoritmo Apriori en generar todos los itemsets frecuentes	Itemsets = 40 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Apriori
3	Dado 60 Ítems se calcula el tiempo en que tarda el algoritmo Apriori en generar todos los itemsets frecuentes	Itemsets = 60 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Apriori
4	Dado 80 Ítems se calcula el tiempo en que tarda el algoritmo Apriori en generar todos los itemsets frecuentes	Itemsets = 80 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Apriori
5	Dado 100 Ítems se calcula el tiempo en que tarda el algoritmo Apriori en generar todos los itemsets frecuentes	Itemsets = 100 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Apriori

Clase: BruteForce			
Caso #	Descripción de la prueba	Valores de entrada	Resultado
6	Dado 20 Ítems se calcula el tiempo en que tarda el algoritmo BruteForce en generar todos los itemsets frecuentes	Itemsets = 20 transactions = 21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo BruteForce
7	Dado 40 Ítems se calcula el tiempo en que tarda el algoritmo BruteForce en generar todos los itemsets frecuentes	Itemsets = 40 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo BruteForce
8	Dado 60 Ítems se calcula el tiempo en que tarda el algoritmo BruteForce en generar todos los itemsets frecuentes	Itemsets = 60 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo BruteForce
9	Dado 80 Ítems se calcula el tiempo en que tarda el algoritmo BruteForce en generar todos los itemsets frecuentes	Itemsets = 80 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo BruteForce
10	Dado 100 Ítems se calcula el tiempo en que tarda el algoritmo BruteForce en generar todos los itemsets frecuentes	Itemsets = 100 transactions=21843 threshold = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo BruteForce

Clase: Clustering			
Caso #	Descripción de la prueba	Valores de entrada	Resultado
11	Dado 20 Ítems se calcula el tiempo en que tarda el algoritmo Clustering en generar todos los itemsets	Itemsets = 20 Transactions=21843 Relación = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Clustering
12	Dado 40 Ítems se calcula el tiempo en que tarda el algoritmo Clustering en generar todos los itemsets	Itemsets = 40 Transactions=21843 Relación = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Clustering
13	Dado 60 Ítems se calcula el tiempo en que tarda el algoritmo Clustering en generar todos los itemsets	Itemsets = 60 Transactions=21843 Relación = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Clustering
14	Dado 80 Ítems se calcula el tiempo en que tarda el algoritmo Clustering en generar todos los itemsets	Itemsets = 80 Transactions=21843 Relación = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Clustering
15	Dado 100 Ítems se calcula el tiempo en que tarda el algoritmo Clustering en generar todos los itemsets	Itemsets = 100 Transactions=21843 Relación = 0.95	Tiempo que tarda en generar todos los itemsets frecuentes el algoritmo Clustering

1.5. Fase 5: Planeación y organización del trabajo experimental

La organización y planeación del experimento se encuentra en el repositorio de Github del proyecto[1] en un archivo de Microsoft Project. Este archivo cuenta con la asignación de responsabilidades y la forma operativa en que se realizan las actividades.

1.6. Fase 6: Realización del experimento

Las métricas obtenidas al realizar el experimento encuentran en el repositorio de Github del proyecto[1].

[1] Link del repositorio: https://github.com/saradrada/AllersGroup_IntegradorI

2. Análisis de Complejidad

2.1 Brute Force

Es una técnica de búsqueda también llamada búsqueda exhaustiva que consiste en observar todos los posibles candidatos a una solución.

Complejidad Temporal: O(NMw), M = 2n

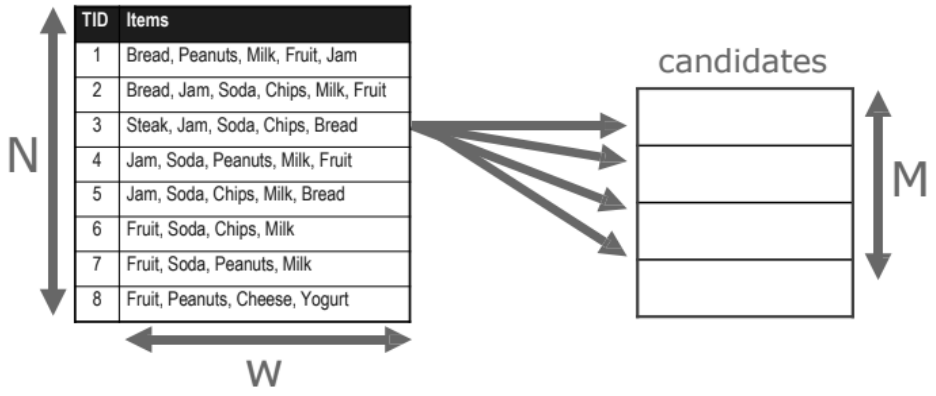

Figura 2.1: Representación Complejidad Temporal [1].

Pseudocódigo.

Algorithm 1: COMBINATIONS(c, S)

Input: c, S
Output: All subsets of size c of S

```
1 root ← ∅
2 s ← |S|
3 COMBINATIONSREC(c, 0, root, s)
4 icombinations ←EXTRACTFROMTREE(root)
5 combinations ←MAPPOSITIONSTOELEMENTS(icombinations)
6 return combinations
```

Figura 2.2: Pseudocódigo Combinaciones [1].

2.2 A priori

Permite encontrar de forma eficiente conjuntos de ítems frecuentes, los cuales sirven de base para generar reglas de asociación

Complejidad Temporal: O(NMW)

N: Cantidad de ítems en un itemset.
M: Cantidad de transacciones.
W: Cantidad de itemsets.

Pseudocódigo.

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: $C_k = \text{apriori-gen}(F_{k-1})$. {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: $C_t = \text{subset}(C_k, t)$. {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

Figura 2.3: Pseudocódigo de Generación de Itemsets Frecuentes para A priori [2].

Algorithm 6.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets
2: Let $k_{\max}$ denote the maximum size of closed frequent itemsets
3: $F_{k_{\max}} = \{f | f \in C, \ |f| = k_{\max}\}$ {Find all frequent itemsets of size $k_{\max}$.}
4: **for** $k = k_{\max} - 1$ downto 1 **do**
5: $F_k = \{f | f \subset F_{k+1}, \ |f| = k\}$ {Find all frequent itemsets of size k.}
6: **for** each $f \in F_k$ **do**
7: **if** $f \notin C$ **then**
8: $f.support = \max\{f'.support | f' \in F_{k+1}, \ f \subset f'\}$
9: **end if**
10: **end for**
11: **end for**

Figura 2.4: Pseudocódigo Algoritmo de Support Count utilizando itemsets frecuentes [2].

2.3 Clustering

Agrupa un conjunto de objetos de tal manera que los miembros del mismo grupo (llamado clúster) sean más similares.

Complejidad Temporal: O(NM^3)

N: cantidad máxima de ítems que compra un cliente
M: cantidad de clientes

Pseudocódigo.

Algorithm 1 The k-Medoid Algorithm

Input: k : The total number of clusters to generate.
Input: D : The collection of input documents.
Input: T : Total number of iterations allowed.
Output: P : The array of k clusters of the partition.

```
1:  Function k-Medoidsk, D
2:      iterCount ← 0
3:      repeat
4:          {m_1, m_2, ..., m_k} ← InitializeMedoids(k, D)
5:          P ← [{m_1}, {m_2}, ..., {m_k}]
6:          repeat
7:              D̂ ← {d_i : d_i ∈ D ∧ d_i ∉ {m_1, m_2, ..., m_k}}
8:              for each: d_i ∈ D̂ do
9:                  for j = 1 to k do
10:                     sim_{i,j} ← CosineSimilarity d_i, m_j
11:                 end for
12:                 imax ← argmax_{1≤j≤k}(sim_{i,j})
13:                 P[imax] ← P[imax] ∪ {d_i}
14:             end for
15:             oldMedoids ← {m_1, m_2, ..., m_k}
16:             {m_1, m_2, ..., m_k} ← UpdateMedoids P
17:             iterCount ← iterCount + 1
18:         until (oldMedoids = {m_1, m_2, ..., m_k}) ∨ (iterCount = T)
19:         NoTinyClusters ← CheckSizeOfClusters (C)
20:     until (NoTinyClusters = True) ∨ (iterCount = T)
21:     return P
22: end function
```

Figura 2.5: Pseudocódigo del algoritmo K-Means [3]

3. Análisis ANOVA

Para realizar el análisis ANOVA de este experimento, se tienen que tener en cuenta dos tipos de hipótesis: la hipótesis nula establece que todas las medias de la población (medias de los niveles de los factores) son iguales mientras que la hipótesis alternativa establece que al menos una es diferente. Para el análisis ANOVA se establece un nivel de confianza del 95%.

Tamaño 10

Análisis de varianza de un factor

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
A priori	1000	1975,5532	1,9755532	3,692566027
Brute Force	1000	2718,653	2,718653	3,055762155
Cloustering	1000	2290,1924	2,2901924	2,374538554

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	278,2578749	2	139,1289374	45,75171647	0	2,998728735
Dentro de los grupos	9113,743869	2997	3,040955579			
Total	9392,001744	2999				

Tamaño 20

Análisis de varianza de un factor

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
A priori	1000	22803,8452	22,8038452	384,2864544
Brute Force	1000	19786,928	19,786928	47,70587204
Cloustering	1000	10402,9723	10,4029723	38,27316614

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	83647,35422	2	41823,67711	266,8089267	2,295E-107	2,998728735
Dentro de los grupos	469795,2271	2997	156,7551642			
Total	553442,5813	2999				

Tamaño 30

Análisis de varianza de un factor

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
A priori	1000	288251,976	288,251976	20952,89031
Brute Force	1000	351022,5765	351,0225765	2685,27336
Cloustering	1000	42994,6461	42,9946461	297,3269151

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	52990837,35	2	26495418,68	3320,85343	0	2,998728735
Dentro de los grupos	23911555,1	2997	7978,496863			
Total	76902392,45	2999				

Tamaño 40

Análisis de varianza de un factor

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
A priori	1000	1078153,707	1079,23294	267924,9095
Brute Force	1000	2837917,487	2840,758245	1334759,426
Cloustering	1000	95274,8375	95,37020771	410,0562494

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	3865501547	2	1932750774	3616,91261	0	2,99873174
Dentro de los grupos	1599888203	2994	534364,7972			
Total	5465389750	2996				

Tamaño 50

Análisis de varianza de un factor

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
A priori	1000	2304959,72	2307,266987	1055031,001
Brute Force	1000	25573391,41	25598,9904	598626763,5
Cloustering	1000	689926,4394	690,6170565	64747,51513

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Probabilidad*	*Valor crítico para F*
Entre grupos	3,88126E+11	2	1,94063E+11	970,7262759	0	2,99873174
Dentro de los grupos	5,98547E+11	2994	199915514			
Total	9,86674E+11	2996				

4. Interpretación

Debido a que en todas las ANOVAs se encontró que la probabilidad calculada es menor a 0,5% y que el estadístico F es mayor al F critico se puede rechazar la Hipótesis nula con un nivel de confianza del 95% es decir que no hay similitud entre los tiempos de los algoritmos.

Teniendo en cuenta los datos el promedio y la varianza el algoritmo más eficiente es Clustering debido a que tiene el mejor rendimiento en tiempo y menor desviación entre sus tiempos de respuesta.

5. Conclusiones

A partir del experimento anterior se utilizará en mayor medida el algoritmo de Clustering para el desarrollo del software ya que da solución al problema de Aller y es el más eficiente temporalmente, sin embargo, se implementará el algoritmo Apriori para el desarrollo de algunos requerimientos específicos.

Bibliografía

[1] Wodarz, N. (2018). Recuperado de https://www4.uwsp.edu/math/nwodarz/Math209Files/209-0809F-L10-Section06_03-AlgorithmsForGeneratingPermutationsAndCombinations-Notes.pdf

[2] Lahti, L. (2018). Recuperado de http://www.cis.hut.fi/Opinnot/T-61.6020/2008/apriori.pdfAlgoritmos de minería de datos (Analysis Services: minería de datos). (2018). Recuperado de https://docs.microsoft.com

[3] G. Soni, K., & Patel, A. (2017). Comparative Analysis of K-means and K-medoids Algorithm on IRIS Data. Recuperado de https://www.ripublication.com/ijcir17/ijcirv13n5_21.pdf