# Diseño de pruebas

Generación de Reglas y itemSets frecuentes con el algoritmo apriori

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo | Propósito | Escenario | Entrada | Resultado |
| ApGenRules(long kItemSet , List<long> itemSets) | Probar la correcta generación de reglas donde consecuentes y antecedentes constan de un solo ítem. | itemsToEvaluate=3;  minConfidence=0.35;  rules={}  support(101)=0.2;  support(100)=0.5;  support(001)=0.6; | Argumentos:  (101,{100,001}) | El estado de rules al final del método será:  rules={[100,001]} |
| AprioriRuleGeneration(List<List<long>> frequentItemSets) | Probar la correcta generación de reglas donde consecuentes y antecedentes constan de uno o dos ítems. | itemsToEvaluate=4;  minConfidence=0.35;  rules={}  support(1000)=0.7;  support(1100)=0.5;  support(0100)=0.8;  support(1001)=0.55;  support(0010)=0.83;  support(1110)=0.3;  support(0001)=0.91;  support(1011)=0.1;  support(1010)=0.58;  support(0110)=0.66;  support(0011)=0.6; | Argumentos:  ({{1010,0011},{1110,1011}}) | rules={[0010,1000],[1000,0010],[0001,0010],[0010,0001],[0110,1000],[1010,0100],[1100,0010],[0010,1100],[0100,1010],[1000,0110]} |
| RemoveNonFrequentItemSetsFromCandidateSet(List<long> candidateSet) | Probar que se descartan correctamente los itemsets que no son frecuentes | itemsToEvaluate=6;  minSupport=0.3;  Transacciones:  n=14  100111  111100  001000  111000  111000  111000  100011  101010  110011  011011  110110  111111  000001  100000 | Argumentos:  ({100100,011000,10100}) | Lista resultante:  {011000,10100} |
| AprioriGen(List<long> frequentItemSets) | Probar el correcto funcionamiento de la generación de candidatos | itemsToEvaluate=6; | Argumentos:  ({101100,011100,110001,101001,011010,110010}) | Resultado:  {101101,011110,110011} |
| GenerateFrequentItemSetsApriori(Item[] frequentOneItemSets) | Probar la correcta generación de ítemSets frecuentes dados unos ítems a evaluar y transacciones | itemsToEvaluate=4;  maxItemSetSize=3;  minSupport=0.5;  i1=new Item(“120”,”Alcohol”);  i2=new Item(“130”,”Manzana”);  i3=new Item(“200”,”Pera”)  i4=new Item(“300”,”Papa”)  Transacciones:  n=10  1234  0011  1100  0110  0001  1000  1111  1100  1110  1001  1110 | Argumentos: | Resultado:  {{1100}} |
|  |  |  |  |  |

**PRUEBAS PARA EL MÉTODO BRUTEFORCE**

**Escenario 1:**

No ítems= 3

Transacciones:

{100}, {110},{110},{011},{111},{111},{100},{100},{010}

**Escenario 2:**

No ítems= 4

Transacciones:

{1111},{1101},{1101},{1011},{0111},{0001},{0101},{0010},{1110},{0111},{0111},{1010},{0101},{1110},{1110}

**Escenario 3:**

No ítems= 3

Transacciones:

{101}

**Escenario 4:**

No ítems= 3

Transacciones:

{110},{110},{011},{011}

**PRUEBAS PARA ARBOL FP**

**Escenario1:**

{{“b”,”a”,”c”}, {“c”,”a”,”d”}, {“a”,”d”}, {“c”, “e”}, {“b”, “f”}, {“d”, “c”}}

minSup = 0.5

Se inicializa el Árbol llamando al constructor que recibe una lista.

**Escenario2:**

{[{“b”,”a”,”c”}, 1], [{“c”,”a”,”d”}, 1], [{“a”,”d”}, 1], [{“c”, “e”}, 1], [{“b”, “f”}, 1], [{“d”, “c”}, 1]}

minSup = 3

Se inicializa el Árbol llamando al constructor que recibe un diccionario

**Escenario 3:**

{{“b”,”a”,”c”}, {“c”,”a”,”d”}, {“a”,”d”}, {“c”, “e”}, {“b”, “f”}, {“d”, “c”}}

minSup = 0.1

Se inicializa el Árbol llamando al constructor que recibe una lista.

**Escenario 4:**

{[{“a”,”c”,”b”}, 2], [{“d”, “e”}, 3], [{“b”,”c”}, 2], [{“b”,”d”}, 3]}

minSup = 4

Se inicializa el Árbol llamando al constructor que recibe un diccionario

**Escenario 5:**

{ {“b”, “c”, “a”, “e”}, {“c”, “d”}, {“a”, “c”, “f”, “b”}, {“c”, “a”, “b”}, {“f”, “e”, “d”}, {“a”, “d”, “c”} }

minSup = 0.5

Se inicializa el Árbol llamando al constructor que recibe una lista.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo | Propósito | Escenario | Entrada | Resultado |
| ConstructFPTree (List<List<string>> trans, double minsup) | Demostrar que el árbol se construye correctamente cuando hay items no frecuentes | 1 | Las definidas por el escenario. |  |
| ConstructFPTree (Dictionary<List<string>, int> trans, double minsup) | Demostrar que el árbol se construye correctamente cuando hay items no frecuentes, utilizando un diccionario con todas las transacciones repitidienose una sola vez | 2 | Las definidas por el escenario | C:\Users\WINDOWS 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\escenario1.png |
| ConstructFPTree (List<List<string>> trans, double minsup) | Demostrar que el árbol se construye correctamente cuando todos los ítems son frecuentes | 3 | Las definidas por el escenario | C:\Users\WINDOWS 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\767b2e7c-3011-42f2-a433-f3a4d61f33be.jpg |
| ConstructFPTree (Dictionary<List<string>, int> trans, double minsup) | Demostrar que el árbol se construye correctamente cuando hay items no frecuentes, utilizando un diccionario con las transacciones repitidienose más de una sola vez | 4 | Las definidas por el escenario | C:\Users\WINDOWS 10\Desktop\e744c6f7-36b6-4e6c-a08d-bd3e062f4b3d.jpg |
| FindFrequentItemSets<List<List<string>>() | Demostrar que el algoritmo funciona correctamente cuando no hay itemsets frecuentes mayores a 1 | 1 | N.A. | Lista vacía |
| FindFrequentItemSets<List<List<string>>() | Demostrar que el algoritmo funciona correctamente cuando todas las transacciones son frecuentes | 3 | N.A. | { {“b”,”c”}, {“a”,”b”}, {“c”, “a”, “b”}, {“c”, “e”}, {“b”, “c”} , {“d”, “a”}, {“d”, “a”, “c”}, {“a”, “c”}, {“d”, “c”} } |
| FindFrequentItemSets<List<List<string>>() | Demostrar que el algoritmo funciona correctamente cuando hay ítems y transacciones que no son frecuentes. | 5 | N.A. | { {“b”, “a”}, {“b”, “c”}, {“b”, “a”, “c”}, {“a”, “c”} } |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Valores Entrada | Resultado |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario1  MaxItemSetSize=2  Minsup=0,4 | Todos los ítems | {110},{100},{010} |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario1  MaxItemSetSize=2  Minsup= 0,3 | Todos los items | {110},{011},{100},{010},{001} |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario2  MaxItemSetSize=2  Minsup= 0,3 | Todos los items | {1100},{1010},{0010},{0110},{0101}  {0001},{0011},{1000},{0100} |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario2  MaxItemSetSize=3  Minsup= 0,2 | Todos los items | {1110},{1000},{0001},{0011},{1001}  {0111},{0100},{1100},{1010}  {0010},{0110},{0101} |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario3  MaxItemSetSize=2  Minsup= 0,1 | Todo los items | {101} |
| AssociationAnalyzer | GenerateFrequentItemSets() | Escenario4  MaxItemSetSize=2  Minsup= 0,2 | Todos los items | {110},{011} |
| AssociationAnalyzer | CommonItems() | Escenario1  Top=2 |  | Debe retornar el ítem 1 y 2 |
| AssociationAnalyzer | CommonItems() | Escenario2  Top=3 |  | Debe retornar el ítem 2 3 y 4 |
| AssociationAnalyzer | CommonItems() | Escenario4  Top=3 |  | Retorna todos los items |
| AssociationAnalyzer | CountSetBits() |  | 15 | 4 |
| AssociationAnalyzer | CountSetBits() |  | 12 | 3 |
| AssociationAnalyzer | CountSetBits() |  | 6 | 2 |
| AssociationAnalyzer | CountSetBits() |  | 2 | 1 |
|  |  |  |  |  |