DIAGNÓSTICO DE CALIDAD DE DATOS EN SISTEMAS DE MANTENIMIENTO. COMPARACION ENTRE MAPAS DE KARNAUGHT Y ALGORITMOS DE INDUCCIÓN

G. Cuello[1], P. Britos[2] y R. García-Martínez[2]

[1]Petrobras Energía, UTE Puesto Hernandez, Ruta Prov. 6, Km 20 (8319)
[2]Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. Instituto Tecnológico de Buenos Aires

Resumen: La calidad de la información de mantenimiento de equipos de superficie depende principalmente de la calidad de los reportes diarios provenientes de los distintos frentes de trabajo. La calidad de un parte diario implica aprobación en tiempo y contenido. Para facilitar el análisis de tiempos de aprobación de partes diarios relacionados con el mantenimiento de equipamiento de superficie, en el ámbito de la operación de un área petrolera, se describe y se compara la aplicación de dos algoritmos, uno de inducción y otro basado en mapas de Karnaught. La aplicación intenta ser un enfoque alternativo al clásico análisis de tendencias temporales, donde las reglas obtenidas no son utilizadas para aplicarse en un clasificador de clases, sino para describir la información almacenada en los registros de aprobación de partes diarios. Los aprobadores son clasificados considerando todas las cuadrillas de operarios por la que son responsables y el tiempo que dedican desde que se entrega cada reporte hasta que se lo controla y aprueba; las reglas son traducidas a conceptos de resumen que se utilizan en las reuniones del área de mantenimiento para realimentar a los aprobadores con información acerca de su desempeño como controladores del flujo de información de mantenimiento.

1. Introducción

El análisis de la calidad de un sistema de gestión del mantenimiento se relaciona con la calidad de la información reportada en las diferentes etapas del proceso total. El sistema de control de mantenimiento de las instalaciones constituye la principal fuente de datos de confiabilidad y mantenimiento de los equipos. La calidad de los datos obtenidos de esta fuente depende, en primera instancia, de la manera en que se reportan; en segundo lugar cuándo son revisados y aprobados los reportes. Por lo tanto es imperioso aplicar un lenguaje consistente y único tanto en la recolección y registro de fallas en cualquier operación, así como para describir los plazos de aprobación de cada reporte. Relacionado con esto último se exponen dos algoritmos para describir la información. Utilizando en primer lugar un conjunto reducido de datos tipo, se presenta en orden, el tipo de información a analizar, los resultados de clasificación por inducción sobre este tipo de datos, los resultados utilizando una forma alternativa de generar reglas mediante mapas de Karnaught. Finalmente se hace lo propio sobre el conjunto de datos real y se exponen pros y contras de cada método.

2. El tipo de información a procesar

2.1. Reporte de mantenimiento

El reporte de mantenimiento tiene asociadas dos tablas de registros, la primera que almacena el detalle del trabajo y es cargada por cada cuadrilla de trabajo en campo mediante colectores de datos; la segunda almacena datos relativos a las fechas hito del reporte: cuándo se realizó la tarea en campo, cuándo se

subió al sistema la información de los colectores de campo, y cuándo fue aprobado por el supervisor. Mensualmente se realizan reuniones con los supervisores de mantenimiento para discutir indicadores de calidad de la gestión del mantenimiento. Uno de estos indicadores es el "tiempo de aprobación de partes diarios". Cada supervisor es controlado con respecto a la demora en días que existe entre la fecha de aprobación y la fecha de alta en el sistema. Normalmente se analiza la respectiva tendencia temporal, que posee un umbral con el límite máximo permitido, comentando aspectos cualitativos de la misma. La propuesta del trabajo consistió en evaluar si fuese posible la aplicación de un algoritmo de inducción tal que devuelva reglas en forma automática de modo de simplificar la tipificación del comportamiento de cada supervisor, agilizando la devolución de perfomance a cada miembro del equipo. Mediante la aplicación de esta técnica se esperaba contar con el árbol de decisión o las reglas, con las cuales se pudiera describir fácilmente la conducta de cada uno de los supervisores durante una reunión de seguimiento.

2.2. Datos de partida

La figura 1 muestra los datos básicos con los que se contó para este trabajo.

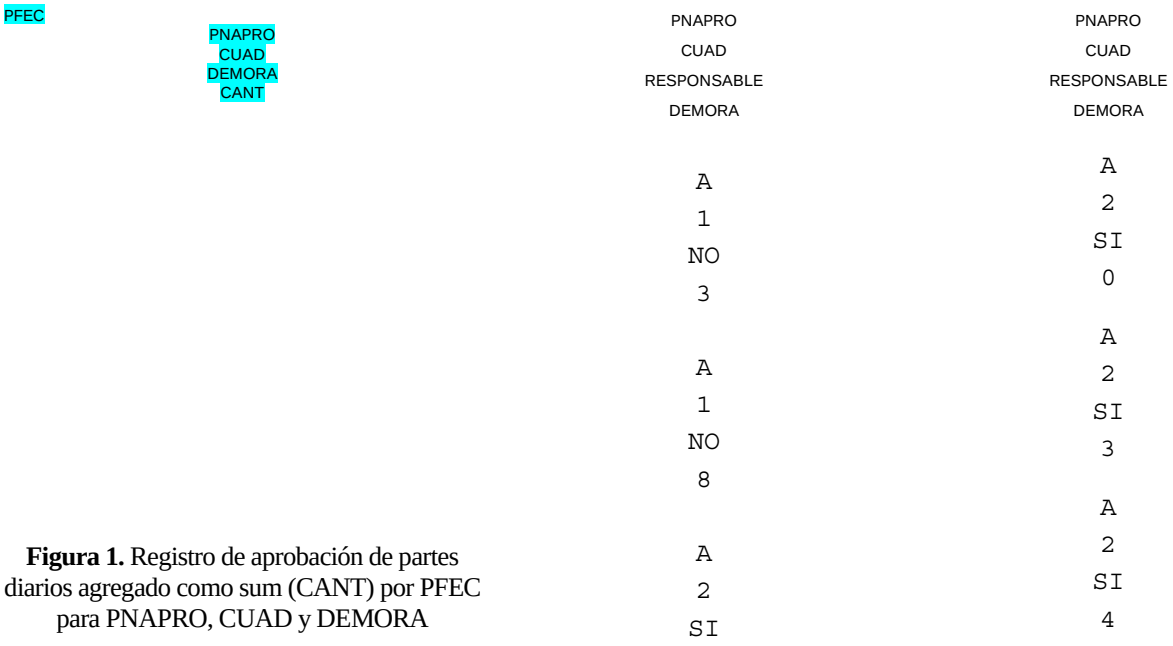


Figura 1. Registro de aprobación de partes diarios agregado como sum (CANT) por PFEC para PNAPRO, CUAD y DEMORA

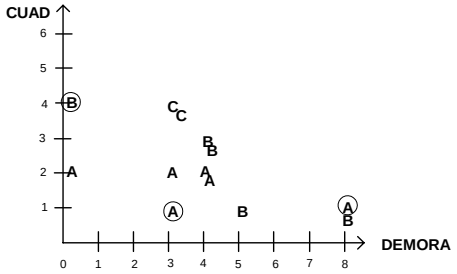


Figura 2. Gráfico de dispersión de los datos iniciales

0				
A				
2				
SI				
3				
A				
2				
SI				
4				
A				
2				
SI				
4				
B				
1				
SI				
5				
B				
1				
SI				
8				
B				
3				
SI				
4				
B				
3				
SI				
4				
B				
4				
NO				
0				
C				
4				
SI				
3				
C				
4				
SI				
3				

Tabla I. Ejemplo

A				
2				
SI				
4				
B				
1				
SI				
5				
B				
1				
SI				
8				
B				
3				
SI				
4				
B				
3				
SI				
4				
C				
4				
SI				
3				
C				
4				
SI				
3				

Tabla II. tabla final para efectuar el análisis

2.2.1. Condiciones de borde que debe respetar la información a procesar

Para aplicar el análisis de inducción, es preciso que se satisfagan los siguientes requisitos:

a) toda la información referida a una muestra sea expresable en términos de una colección fija de atributos o propiedades,

b) cada atributo puede contener tanto valores numéricos como valores discretos, pero los atributos utilizados para describir las muestras, no varían de una muestra a otra,

c) las categorías a las que se asignarán las muestras deben estar pre-establecidas (si se pretende implementar un aprendizaje supervisado), y,

d) las clases deberán estar perfectamente delineadas, de modo que una muestra pertenezca a una sola clase.

Con respecto a la figura 2, se muestra un tipo de caso frecuente que se da a la hora de realizar los análisis de mantenimiento: un supervisor que cubre a otro por ausencia, se convierte en aprobador de cuadrillas no usuales, por lo que violaría la condición de borde (d). Antes de aplicar el algoritmo, se puede optar:

- eliminar las tuplas con clase superpuesta (en la figura 2 es el caso de la clase A y B, que se ubican en CUAD=1 y DEMORA=8); se incluye en el análisis los relevos de supervisión, o,
- eliminar de la tabla I, correspondiente a la figura 2, aquellas tuplas con registro de aprobadores que no estén a cargo de una cuadrilla determinada; es decir, para el análisis no se tienen en cuenta los reemplazos temporarios entre supervisores.

2.3. Característica general del problema

Teniendo en cuenta solo los atributos PNAPRO, CUAD, DEMORA y realizando un gráfico de dispersión con ejes CUAD-DEMORA, es posible indicar a todos los aprobadores. En el ejemplo (ver Tabla I), A es responsable de aprobar la cuadrilla 2; B es responsable por la 1 y la 3; y C por la 4. Con un círculo se indican las tuplas que se eliminarán. En el presente trabajo, se opta por analizar a los supervisores, sin tener en cuenta los reemplazos temporarios. En la tabla II se muestra la tabla final para efectuar el análisis. CUAD y DEMORA son categorías, discretas y continuas, respectivamente; PNAPRO es la clase que se desea clasificar.

2.4. Análisis algorítmico

Aplicando el algoritmo C4.5 a este ejemplo, tal como se describe en [Kantardzic2003; Huenerfauth2005; Pei2003; Littman et al. 2003] sin poda de ramas y considerando todos los datos como datos para entrenamiento, resulta un árbol de decisión del cual se infieren las regals descriptas en la Tabla III cuyo soporte, confianza y grado de captura viene dado por la Tabla IV.

```
Rule0

PNAPRO = a

Rule1
IF
CUAD = 1

THEN
PNAPRO = b
```

RULE ID	0	1	2
CLASS	a	b	
SUPPORT	100.0%	20.0%	
CONFIDENCE	40.0%	100.0%	
CAPTURE	100.0%	50.0%	

Rule2 IF CUAD = 2	a 40.0% 100.0% 100.0%
THEN PNAPRO = a	3 b 20.0% 100.0% 50.0%
Rule3 IF CUAD = 3	4 c 20.0%
THEN PNAPRO = b	100.0% 100.0%
Rule4 IF CUAD = 4	
THEN PNAPRO = c	

Tabla IV. Soporte, confianza y grado de captura de las reglas

Tabla III. Reglas inferidas del árbol de decisión

Aún eliminando la opción de poda de ramas del árbol y considerando el atributo DEMORA como continuo, el algoritmo no llega a tomar en cuenta la información del atributo DEMORA.

2.5. Tipificación de los datos mediante las reglas lógicas obtenidas

Teniendo en cuenta las reglas lógicas resultantes, se las puede describir en términos de cubrimientos en el diagrama de dispersión [Kantardzic2003]. Un enfoque formal de los problemas de clasificación está dado si consideramos su interpretación gráfica. Un conjunto de datos con N clases puede pensarse como una colección de puntos discretos (uno por muestra) en un espacio n-dimensional. Una regla de clasificación es un hipercubo en este espacio; el hipercubo puede contener una o más de estos puntos. Cuando existen dos o más cubos para una clase dada, todos los cubos se unen mediante el OR, es decir si para una clase X dada, existe cubo1 para regla-1, cubo2 para regla-2, etc., entonces la clase es descripta mediante:

```
IF [regla-1 OR regla-2 OR … OR regla-n]  THEN clase=CLASEx.
```

Dentro de cada cubo, se efectúa la intersección de las dimensiones mediante el operador AND; así por ejemplo, para regla-1 podría ser

```
IF (Atributo1=valor1 AND Atributo2=valor2 AND…AND Atributo-n=valor-n) THEN clase=CLASEx
```

El tamaño del cubo indica su generalidad, tanto más grande, más hipervértices posee y cubre potencialmente un mayor número de muestras. Con respecto a la figura 3, se observa que si bien la reglas describen bien cada clase, el cubrimiento no tiene en cuenta el atributo DEMORA.

2.6. Pros y contras del algoritmo de inducción para la descripción de la información

El algoritmo C4.5 requiere como mínimo dos clases diferentes. Por lo tanto no podría aplicarse al análisis de un solo supervisor o si se considerara un sólo supervisor por vez, considerando los valores de demora

como clases, tampoco sería aplicable porque exigiría que se cuenten por lo menos con dos valores por cada valor del atributo DEMORA.

3. Solución conceptual

3.1. Propuesta de transformación de los datos a una mapa de Karnaugh

Dado que la necesidad es clasificar adecuadamente a cada supervisor —cada una de las clases—, se explora otra forma adicional de descripción, mucho más rígida en términos de reglas, que pueda aplicarse o bien a todas las clases como conjunto único, o a una sola clase por vez.

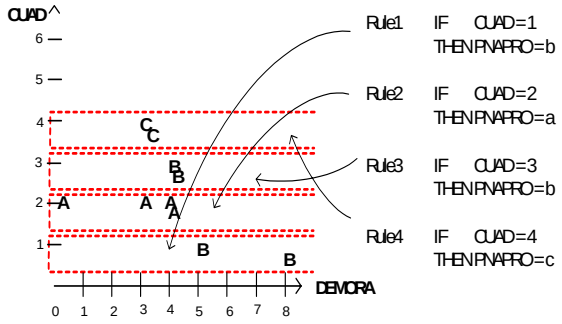


Figura 3. Cubrimiento de las reglas lógicas obtenidas por algoritmo de inducción

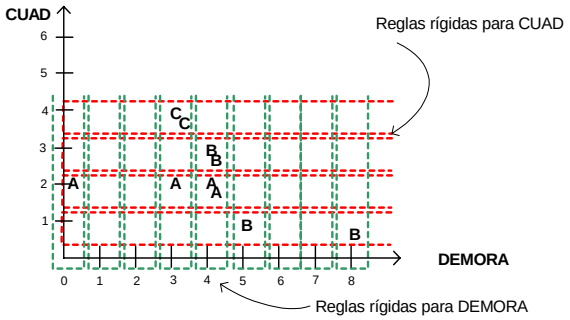


Figura 4. Reglas rígidas para descripción de clases de la figura 3

En la figura 4 se han considerando los atributos como discretos y se han incluido reglas que cubren los valores para DEMORA. Así por ejemplo, se tendría que, para describir la clase C, la regla debería ser:

```
IF (DEMORA=3 .and. CUAD=4) THEN clase=C
```

o bien para describr la clase B,

```
IF    (DEMORA=5 .and. CUAD=1 )  .or.
      (DEMORA=4 .and. CUAD=3) .or.
      (DEMORA=8 .and. CUAD=1)
THEN clase=B.
```

Dada la necesidad de describir el detalle DEMORA para cada supervisor, es posible automatizar un análisis similar al descrito, a fin de determinar todas las reglas que describen la clase. Para ello, si la figura 4 es interpretada como un mapa de Karnaugh, se pueden aprovechar las mismas técnicas de minimización para funciones lógicas que son utilizadas para optimizar los circuitos digitales. Se basa en el postulado siguiente: dado que un algoritmo de inducción genera reglas en forma automática y teniendo en cuenta la función de cubrimiento que éstas cubren en el espacio de atributos categóricos, como clasificador el árbol de decisión no resultaría ser mas que un conjunto de reglas lógicas. Si existe otra forma equivalente de generarlas, que sea mucho más rígida, para asegurar la correcta descripción de los datos, el conjunto resultante de reglas podría seguir siendo utilizado como clasificador. Es decir, la propuesta parte de plantear los cubrimientos básicos (reglas) que son necesarios para describir la información y encontrar una forma óptima de generar las reglas que describan cada clase.

3.2. Aplicación de la transformación propuesta al ejemplo

Teniendo en cuenta que el problema que se estudia es de carácter bi-dimensional y considerando la figura 4, ésta se interpreta como parte de un mapa de Karnaugh que la abarca, donde cada clase se examina por separado. Convirtiendo los valores del par de atributos a forma binaria, con tantos bits como sean necesarios para incluir la grilla en el mapa, y considerando el ordenamiento de los bits tal y como se ordena en un mapa de Karnaugh, se mapea cada atributo a una serie de bits (las entradas del mapa),

figura 5. Cada clase se cubre con mapas de tamaño adecuado al rango de valores. Por ejemplo, para la clase B, el atributo DEMORA se codificará con cuatro bits en vez de tres, dado que uno de los valores para DEMORA es ocho. Se incluye la codificación binaria como un atributo adicional a cada atributo. Así, de la tabla utilizada para el análisis con el algoritmo C4.5, la clase A se convierte a lo mostrado en la tabla V.

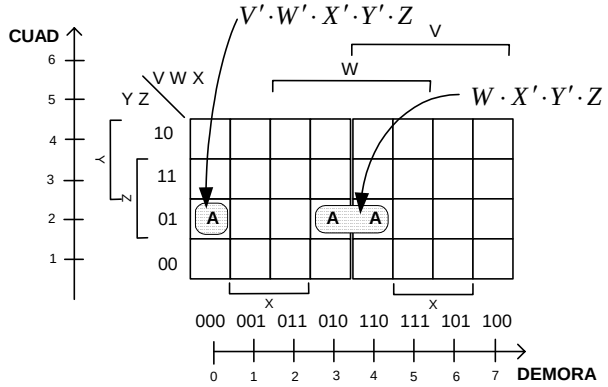


Figura 5. Mapeo de los datos de una clase al mapa de Karnaugh

PNAPRO
CUAD
YZ
DEMORA
V W X
A
2
01
0
000
A
2
01
3
010
A
2
01
4
110
A
2
01
4
110

Tabla V. Conversión de la clase A

Aplicando las técnicas de agrupación utilizadas en los mapas de Karnaugh, se minimiza la función lógica $F = V' \cdot W' \cdot X' \cdot Y' \cdot Z + W \cdot X' \cdot Y' \cdot Z$, que describe la clase A. Para que F sea verdadera, uno o ambos términos deben ser verdaderos; para el primer término, debe ser V=0, W=0, X=0, Y=0 y Z=1; para el segundo, W=1, X=0, Y=0 y Z=1. Una vez obtenida *F* y el conjunto de entradas que la hacen verdadera, se convierten las entradas digitales a los valores de atributos. Este proceso asegura que la función lógica que describe los datos sea mínima porque así lo es *F* y, por tanto, no existirán reglas redundantes para describir las clases. Como para el primer término de F, debe cumplirse que VWX=000 e YZ=01, ingresando a la tabla por las columnas VWX e YZ resulta VWX=000 $\Rightarrow$ {DEMORA=0}, e YZ=01 $\Rightarrow$ {CUAD=2}; entonces la regla para este término descripta con los atributos sería:

```
IF {DEMORA=0} and {CUAD=2} THEN clase=A;
```

para el segundo término, debe cumplirse que VWX=x00 y YZ=01, (V se incluye por estructura; la 'x' indica que puede ser 1 ó 0 indistintamente por lo que se tienen en cuenta los dos casos: VWX=000 o

VWX=100) o sea VWX = x00 $\Rightarrow$ {DEMORA=3 or DEMORA=4}, e YZ = 01 $\Rightarrow$ {CUAD=2}; la regla para el segundo término sería entonces

```
IF {DEMORA=3 or DEMORA=4} and {CUAD=2} THEN clase = A.
```

La unión de las dos reglas da la regla total para determinar la clase A:

```
IF {DEMORA=0} and {CUAD=2} THEN clase=A
                                                    OR
IF {DEMORA=3 or DEMORA=4} and {CUAD=2} THEN clase = A.
```

3.2.1. Algoritmo

Se describe a continuación el algoritmo conceptual.

Algoritmo conceptual

Tabla II

```
1. Obtener las clases
2. Para cada clase
      2.1.  encontrar un mapa cuyo tamaño cubra los rangos
            de valores de los atributos CUAD y DEMORA
      2.2.  insertar las columnas de representación binaria
            para los valores del atributo (construye Tabla
            como la V)
      2.3.  ejecutar técnica de minimización digital y
            obtener la expresión de F
      2.4.  para cada término de F,
                  2.4.1.  obtener el valor binario que lo hace
                          verdadero
                  2.4.2.  partir este valor en los grupos de
                          bits correspondientes a cada atributo
                  2.4.3.  ubicar en la Tabla V el valor binario
                          del atributo y extraer el string de
                          descripción del atributo (parte
                          condicional del IF)
                  2.4.4.  construir y almacenar el string (IF
                          THEN) de la regla para este término
      2.5.  repetir para TERMINO
3. repetir para CLASE
4. presentar conjunto de reglas
```

3.3. Pros y contras del algoritmo

Si bien el algoritmo detallará para cada clase y para cada cubrimiento en el mapa respectivo a una regla, tiene la desventaja de que si el cubrimiento es demasiado abarcativo, generará reglas con muchos "or" tornándolas abrumadoramente detalladas. Por el contrario, si los datos en el mapa están dispersos, el algoritmo genera reglas puntuales concretas.

4. Aplicación al caso real

4.1. Preparación de los datos

La tabla final a utilizar por el algoritmo se simplificó considerando los días de demora divididos en grupos de cuatro días para limitar las descripciones conceptuales del atributo DEMORA y para simplificar la aplicación de los mapas de Karnaugh (esto último puede liberarse si se aplicaran métodos

algorítmicos de minimización tales como los basados en el algoritmo de Quine-McClusckey). En la tabla VI, se ejemplifica para una porción de los datos cómo queda el atributo DEMORA así como cuáles fueron las categorías consideradas para aplicar el algoritmo de inducción.

CLASE	DIAS DEMORA _LIDER	DEMORA	CUAD	RANGO_ DEMORA
CLASE		CAT	CAT	
D	4	1	5	[4,8)
D	1	0	5	[0,4)
D	0	0	5	[0,4)
D	13	3	5	[12,16)
D	13	3	5	[12,16)
D	11	2	5	[8,12)

TABLA VI. generación del atributo DEMORA

La tabla completa tiene una longitud de 4592 registros para los supervisores de mantenimiento y alrededor de 8500 registros adicionales para los supervisores de producción.

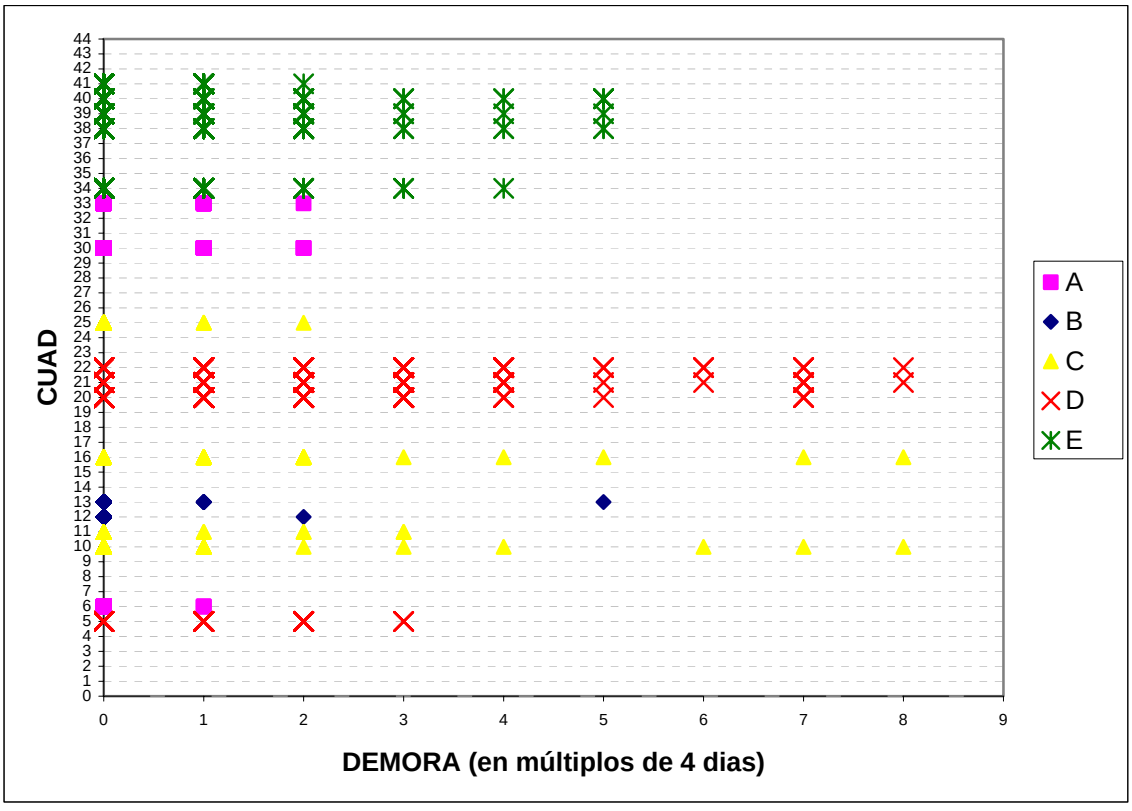


Figura 6. Grafica de todo el conjunto de clases

4.2. Resultados

4.2.1. Algoritmo de inducción

Aplicando el algoritmo C4.5 a la tabla VI sólo para los supervisores de mantenimiento; se obtuvieron resultados cualitativos similares al presentado en el caso conceptual (ver conjunto de reglas resultantes en Tabla VII); la regla 0 se descarta por su bajo grado de confianza. Nótese que las demás reglas poseen el 100% de confianza (ver Tabla VIII).

```
Rule0

CLASE = e

Rule1
IF
CUAD = 5

THEN
CLASE = d

Rule2
IF
CUAD = 6

THEN
CLASE = a

Rule3
IF
CUAD = 10

THEN
CLASE = c

Rule4
IF
CUAD = 11

THEN
CLASE = c

Rule5
IF
CUAD = 12

THEN
CLASE = b

Rule6
IF
CUAD = 13

THEN
CLASE = b

Rule7
IF
CUAD = 16
```

```
Rule10
IF
CUAD = 22

THEN
CLASE = d

Rule11
IF
CUAD = 25

THEN
CLASE = c

Rule12
IF
CUAD = 30

THEN
CLASE = a

Rule13
IF
CUAD = 33

THEN
CLASE = a

Rule14
IF
CUAD = 34

THEN
CLASE = e

Rule15
IF
CUAD = 38

THEN
CLASE = e

Rule16
IF
CUAD = 39

THEN
CLASE = e
```

```
RULE ID
CLASS
SUPPORT
CONFIDENCE
CAPTURE

0
e
100.0%
27.5%
100.0%

1
d
1.4%
100.0%
8.1%

2
a
4.7%
100.0%
29.4%

3
c
5.4%
100.0%
20.7%

4
c
6.7%
100.0%
25.9%

5
b
6.7%
100.0%
50.5%

6
b
6.6%
100.0%
49.5%

7
c
7.0%
100.0%
27.0%

8
d
5.5%
100.0%
```

```
THEN
CLASE = c

Rule8
IF
CUAD = 20

THEN
CLASE = d

Rule9
IF
CUAD = 21

THEN
CLASE = d

Rule17
IF
CUAD = 40

THEN
CLASE = e

Rule18
IF
CUAD = 41

THEN
CLASE = e
```

Tabla VII. Reglas inferidas del árbol de decisión

```
32.1%

 9
 d
5.6%
100.0%
32.8%

 10
 d
4.6%
100.0%
27.0%

 11
 c
6.8%
100.0%
26.5%

 12
 a
6.2%
100.0%
38.5%

 13
 a
5.2%
100.0%
32.1%

 14
 e
5.9%
100.0%
21.3%

 15
 e
5.5%
100.0%
20.1%

 16
 e
5.7%
100.0%
20.7%

 17
 e
5.4%
100.0%
19.6%

 18
 e
5.1%
100.0%
18.4%
```

Tabla VIII. Soporte, confianza y grado de captura de las reglas

4.2.2. Método de los mapas de Karnaugh

Las figuras 7 a 11 muestran los mapeos realizados para cada una de las clases a las que se aplicó un algoritmo similar al descrito en el punto 3.2.1.; en las expresiones lógicas, '*d*' significa *don't care* significando que ambos valores de ese bit deben ser tenidos en cuenta. De los mapeos realizados resultó el siguiente conjunto de reglas:

CLASE A:

```
IF {DEMORA=0 or DEMORA=1}
and {CUAD=30 or CUAD=33}
THEN clase=A
IF {DEMORA=2}
and {CUAD=30 or CUAD=33}
THEN clase=A
IF {DEMORA=0 or DEMORA=1}
and {CUAD=6}
THEN clase=A
```

CLASE B:

```
IF {DEMORA=0 }
and {CUAD=12 or CUAD=13}
THEN clase=B
IF {DEMORA=1 }
and {CUAD=13}
THEN clase=B
IF {DEMORA=2 }
and {CUAD=12 }
THEN clase=B
IF {DEMORA=5 }
and {CUAD=13}
THEN clase=B
```

CLASE C:

```
IF {DEMORA=0 or DEMORA=1 }
and {CUAD=10 or CUAD=11 or CUAD=16
or CUAD=25}
THEN clase=C
IF {DEMORA=2 }
and {CUAD=10 or CUAD=11 or CUAD=16
or CUAD=25}
THEN clase=C
IF {DEMORA=3 } and {CUAD=16}
THEN clase=C
IF {DEMORA=3 }
and {CUAD=10 or CUAD=11 }
THEN clase=C
IF {DEMORA=4 }
and {CUAD=10}
THEN clase=C
IF {DEMORA=4 or DEMORA=5 }
and {CUAD=16}
THEN clase=C
IF {DEMORA=6 or DEMORA=7 }
and {CUAD=10}
THEN clase=C
IF {DEMORA=8 } and {CUAD=10}
THEN clase=C
IF {DEMORA=7 or DEMORA=8 }
and {CUAD=16}
THEN clase=C
```

Tabla IX. Reglas resultantes de los mapeos realizados.

CLASE D:

```
IF {DEMORA=0 or DEMORA=1 or DEMORA=2 or DEMORA=3 }
and {CUAD=5 or CUAD=20 or CUAD=21 or CUAD=22}
THEN clase=D
IF {DEMORA=4 or DEMORA=5 or DEMORA=6 or DEMORA=7 }
and {CUAD=21 or CUAD=22}
THEN clase=D
IF {DEMORA=4 or DEMORA=5 } and {CUAD=20 } THEN clase=D
IF {DEMORA=7 } and {CUAD=20 } THEN clase=D
IF {DEMORA=8 } and {CUAD=21 or CUAD=22} THEN clase=D
```

CLASE E:

```
IF {DEMORA=0 or DEMORA=1  }
and {CUAD=41 } THEN clase=E
IF {DEMORA=2   } and {CUAD=41 }
THEN clase=E
IF {DEMORA=0 or DEMORA=1 or DEMORA=2 or DEMORA=3 }
and {CUAD=34 or CUAD=38 or CUAD=39 or CUAD=40}
THEN clase=E
IF {DEMORA=4 or DEMORA=5  } and {CUAD=39 or CUAD=40 }
THEN clase=E
IF {DEMORA=4 or DEMORA=5  } and {CUAD=38 }
THEN clase=E
IF {DEMORA=4  } and {CUAD=34 }
THEN clase=E
```

Tabla IX. Reglas resultantes de los mapeos realizados (continuación).

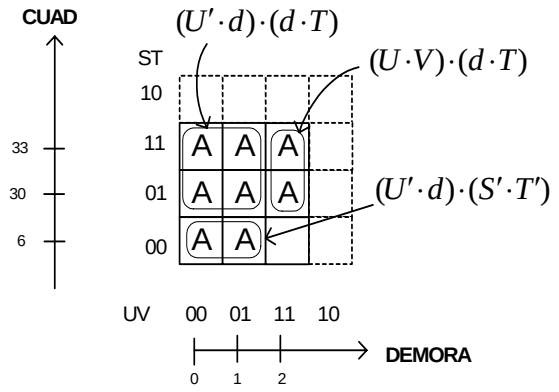


Figura 7. Clase A

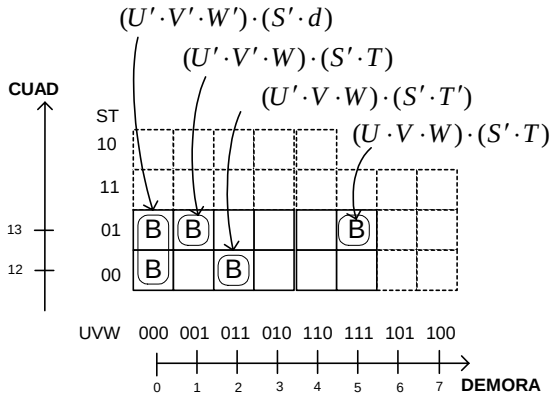


Figura 8. Clase B

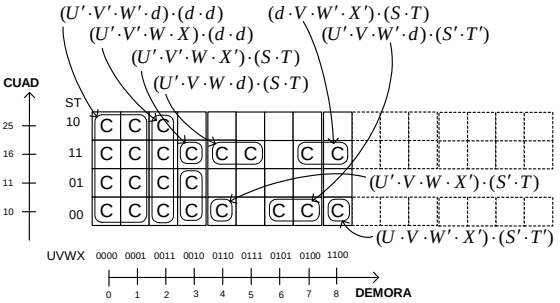


Figura 9. Clase C

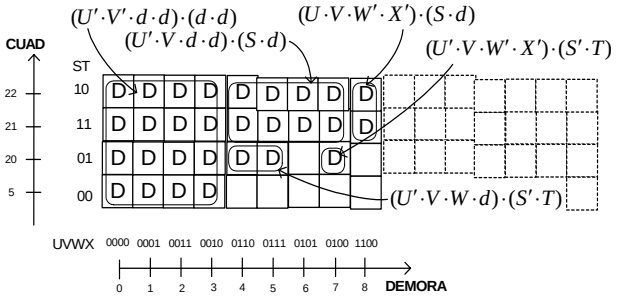


Figura 10. Clase D

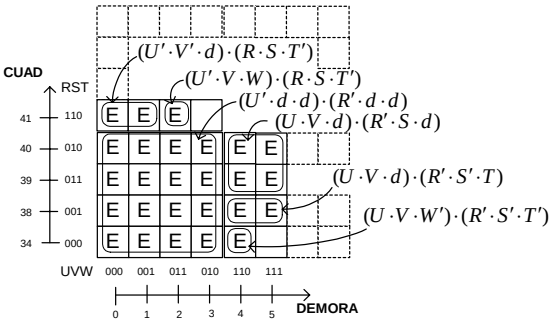


Figura 11. Clase E

5. Futuros trabajos

5.1. Métodos de minimización programada

La mayoría de las funciones lógicas "reales" son demasiado grandes como para minimizarse mediante mapas de Karnaugh. No obstante, éstos constituyeron el punto de partida para métodos tabulares, algorítmicos y que pueden implementarse en computadoras. Tal es el caso del algoritmo de Quine-McCluskey o el algoritmo de concenso iterativo [Wakerly1992] que minimiza funciones para un número arbitrario de variables de entrada. Al igual que el método de los mapas, el algoritmo consta de dos pasos principales: (a) encontrar todos los implicantes primos de la función y, (b) seleccionar un conjunto mínimo de ellos que cubra toda la función. Dado que existen métodos programables para una gran escala de mintérminos, es posible aplicarlo a un gran volumen de información.

5.2. Descripción de clases con más de dos atributos

Queda por explorar si sería factible aplicar los métodos de minimización programada a casos de clasificación multidimensionales, encontrando un mapeo adecuado entre los valores de los atributos y las variables digitales para luego encontrar la función *F* mínima.

6. Conclusiones

Con respecto al análisis previo de la información, una abstracción conceptual acerca de lo que se busca y cómo se lo buscará resulta útil para evitar pitfalls en los algoritmos y así asegurar su efectividad. La visión simplificada y gráfica de cómo operan las reglas de clasificación en un espacio bi-dimensional resulta útil para preparar los datos de partida. El algoritmo C4.5 manifestó su capacidad generalizadora tanto en el ejemplo conceptual como en el caso real, prescindiendo de los detalles en lo que respecta al atributo DEMORA. Tal como se describió en el punto 3.3., el algoritmo con mapas de Karnaught, generó reglas complejas cuando los cubrimientos en el mapa resultaban demasiado abarcativos. Sin embargo, debe considerarse que este análisis abarcó un período de un año, mientras que en el análisis de un mes estas situaciones tan abarcativas no se dan en la práctica y la matriz de datos del tipo de la figura 6, resulta bastante dispersa por lo que el algoritmo genera reglas simples con suficiente detalle.

7. Agradecimientos

A Petrobras Energía S.A., que me han permitido estudiar la especialidad en explotación de datos, abriendo un nuevo horizonte en el ámbito del análisis de la información relacionada con la gestión de la explotación de hidrocarburos.

8. Referencias

Huenerfauth M., 2005, *Decision Tree Problems*, CSE-391: Artificial Intelligence Lecture Notes, University of Pennsylvania. Visitar: http://www.cis.upenn.edu/~cse391/.

Kantardzic M., 2003, *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley & Sons, ISBN: 0471228524, 2003.

Littman M. L., Yihua W., 2003, *Chapter 3: Decision Tree Learning*, CS 536 lecture notes, Rutgers University, Fall 2003. Visitar: http://www.cs.rutgers.edu/~mlittman/courses/ml03/.

Microsoft SQL Server OLAP Servicies 7.0: *OLAP Manager Tutorial*.

Pei J., 2003, Data preprocessing, CSE 626 – Data Mining Lecture Notes, University at Buffalo. Visitar: http://www.cse.buffalo.edu/faculty/jianpei/teaching/datamining/.

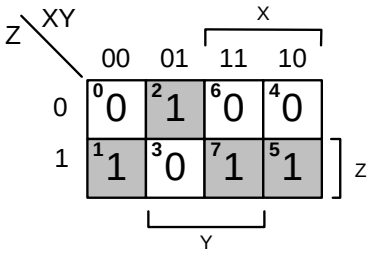
Shannon C. E., 1948, A Mathematical Theory of Communication, Reprinted with corrections from The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656.

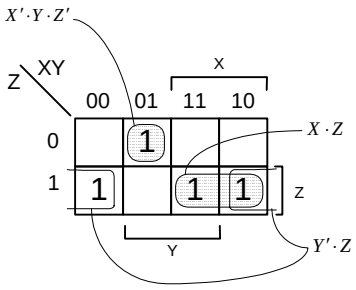
Wakerly J. F., 1992, *Diseño Digital Principios y Prácticas*, Prentice Hall, pp. 150-210, ISBN:968-880-244-1

9. ANEXO: Mapas de Karnaugh

Un mapa de Karnaugh [Wakerly1992] es una representación gráfica de la tabla de verdad de una función lógica. El mapa para una función lógica de *n* entradas es un arreglo de 2^n celdas, una para cada posible combinación de entrada o mintérmino. Las líneas y columnas de un mapa de Karnaugh están etiquetadas para que la combinación de entrada de cualquier celda se determine fácilmente a partir de los encabezados de línea y columna para cada celda. El número pequeño dentro de cada celda es el número de mintérmino correspondiente en la tabla de verdad, suponiendo que las entradas de la tabla de verdad estén etiquetadas de izquierda a derecha y las líneas estén numeradas en orden binario. Cada celda del mapa contiene la información de la línea numerada de manera semejante a la tabla de verdad de la función, 0 si la función da falso para la combinación de entrada; 1 en el caso contrario. Para representar una función lógica en el mapa de Karnaugh, se copian los 1s y los 0s de la tabla de verdad en las celdas correspondientes. En la figura 12 se representa una tabla de verdad de una función *F* cualquiera y su correspondiente mapa de Karnaugh. El ordenamiento "extraño" de los números de las líneas y columnas en el mapa de Karnaugh tiene su fundamento. Cada celda corresponde a una combinación de entradas que difiere de cada una de sus vecinas adyacentes inmediatas solamente en una variable. Cada combinación de entrada con valor F=1 en la tabla de verdad corresponde a un mintérmino en la suma canónica de la función lógica. Ya que los pares de las celdas adyacentes '1' en el mapa de Karnaugh tienen mintérminos que difieren sólo en una variable, los pares de mintérminos pueden combinarse en un sólo término producto uilizando la generalización del teorema T10, término Y + término Y' = término. La utilidad del mapa es clara; sirve para simplificar la suma canónica de la suma lógica de una función lógica.

MINTERM	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5				





$$F = X'\cdot Y \cdot Z' + X \cdot Z + Y'\cdot Z$$

Figura 13. Mapa de Karnaugh que combina celdas 1 adyacentes y suma lógica resultante.

1
0
1
1

6
1
1
0
0

7
1
1
1
1

$F = X'\cdot Y'\cdot Z + X'\cdot Y\cdot Z' + X\cdot Y'\cdot Z + X\cdot Y\cdot Z$

Figura 12. Tabla de verdad, suma canónica y mapa de Karnaugh para $F = \sum_{X,Y,Z}(1,2,5,7)$.

9.1.1. Definiciones

Una *suma mínima* de una función lógica $F(X_1,\cdots,X_n)$ es una expresión de suma de productos para F tal que ninguna expresión de suma de productos para F tenga menos términos producto y cualquier expresión de suma de productos con el mismo número de términos de productos tenga, al menos, el mismo número de literales. Una función lógica $P(X_1,\cdots,X_n)$ *implica* una función lógica $F(X_1,\cdots,X_n)$ si para cada combinación de entrada tal que P=1, entonces sea también P=1. Un *implicante primo* de una función lógica $F(X_1,\cdots,X_n)$ es un término producto normal $P(X_1,\cdots,X_n)$ que implica a F, de manera que si cualquier variable se remueve de P, entonces el término producto resultante no implica a F. En términos de un mapa de Karnaugh, un implicante primo es un conjunto circundado de celdas 1 que satisfacen la regla de combinación, de forma tal que si se trata de hacerlo más grande, se cubren uno o más ceros. En la figura 13 se indican los implicantes primos de la función ejemplo.

Teorema de implicantes primos: una suma mínima es una suma de implicantes primos.

La suma de todos los implicantes primos de una función lógica se denomina *suma completa*. Aunque la suma completa sea siempre una manera legítima de realizar una función lógica, no siempre es mínima.

Una *celda 1 distinguida* de una función lógica es una combinación de entradas cubierta por solo un implicante primo.

Un *implicante primo esencial* de una función lógica es un implicante primo que cubre una o más celdas 1 distinguidas. Dado que es el único implicante primo que cubre alguna celda 1, debe incluirse en cada suma mínima de la función lógica.

Para conseguir una suma mínima, se procede en dos etapas: (a) identificar las celdas 1 distinguidas y los correspondientes implicantes primos e incluir los implicantes primos esenciales en la suma mínima; (b) determinar cómo cubrir las celdas 1, si las hay, que no hayan sido cubiertas por los implicantes primos esenciales.