Árbol de Decisión

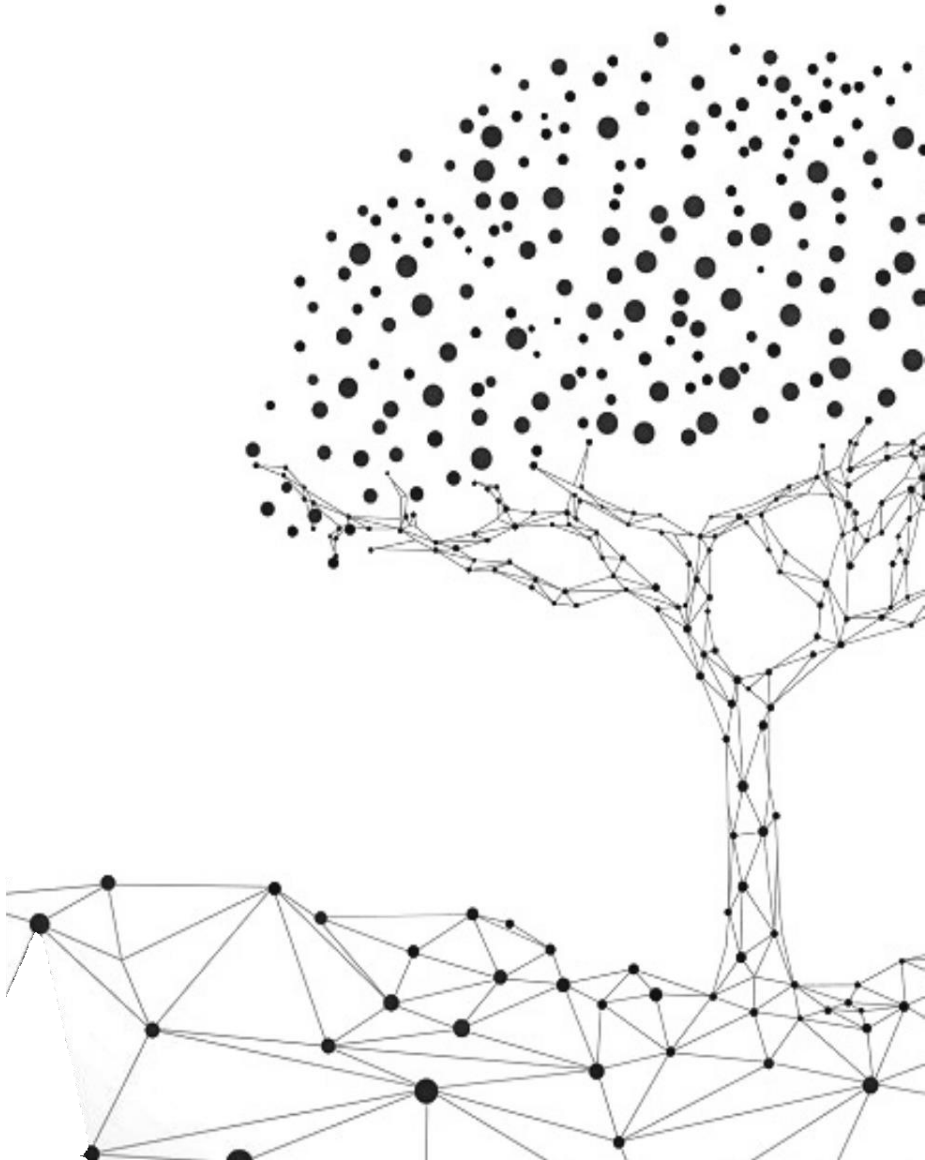
Adrián Maiza, Beatriz Lago, Erick Martínez

FUNDAMENTO TEÓRICO

Los árboles de decisión son una herramienta popular y poderosa utilizada para fines de clasificación y predicción.

Se utiliza un árbol para almacenar una sucesión de decisiones lógicas a partir de unos datos de entrenamiento.

Existen varias técnicas para generar un árbol de decisión. Entre ellos se encuentran ID3, C4.5 (sucesor de ID3), CART, CHAID, etc.



PLANTEAMIENTO DEL PROBLEMA

El problema a tratar es predecir si un personaje literario va a morir en función de unos atributos. En este caso, sus atributos son ocho categóricos y dos continuos. Por ello, para estos dos últimos atributos, se calcula un umbral para poder separar los datos a la hora de generar el árbol.

He aquí una parte del conjunto de datos que se utiliza para generar el árbol:

male	book1	book2	book3	book4	book5	isMarried	isNoble	numDead Realtions	popularity	isAlive
1	0	0	0	0	0	0	0	11	0.605351170568561	0
1	1	1	1	1	1	1	1	1	0.896321072341129	1
1	0	0	0	1	0	0	1	0	0.267558528428093	1
0	0	0	0	0	0	1	1	0	0.183946488294314	0
0	0	0	0	1	0	1	1	0	0.0434782608695652	1
1	0	0	0	0	0	0	0	5	1.0	1
1	0	0	0	0	0	1	1	0	0.0431438127090301	0
1	0	0	0	0	0	0	0	5	0.6678929765886287	0
1	0	0	1	0	0	0	1	0	0.0066889632107023	0

UMBRAL

```
for(int i=1; i<N; i++){ //Almacena en una pila los posibles umbrales
    if(ant != sig){ //Si el isAlive del dato en esta posición es distinto al que tenemos almacenado
        x.muertos_Dch = total_Muertos - x.muertos_Izq;
        x.vivos_Dch = totalVivos - x.vivos_Izq;
        x.pos = i;
        apilar(&p,x);
    }
    if(vect[i].isAlive == 1){
        x.vivos_Izq++;
    } else{
        x.muertos_Izq++;
    }
    ant = vect[i].isAlive;
    sig = vect[i+1].isAlive;
}

while(!esNulaPila(p)){
    x = cima(p);
    Ent = entropia_umbral(x);
    if(Ent<MinEnt){
        MinEnt = Ent;
        PosEntMin = x.pos;
    }
    desapilar(&p);
}
Ent=entropia_umbral(x);
if(Ent<MinEnt){
        MinEnt = Ent;
        PosEntMin = x.pos;
} //Para tratar el ultimo elemento de la pila porque se quedaba sin tratar
//Cálculo del umbral
```

Utilizamos una pila para almacenar los posibles umbrales con sus respectivos vivos/muertos a la izquierda y a la derecha. Así solo recorremos el array de datos una vez para almacenar los umbrales y desapilamos cada umbral para calcular la entropía de cada uno de ellos.

UMBRAL

```
typedef struct tipoElementoPila_t{
    int pos;
    double vivos_Izq;
    double vivos_Dch;
    double muertos_Izq;
    double muertos_Dch;
}tipoElementoPila;

typedef struct celdaP{
     tipoElementoPila elem;
     struct celdaP *sig;
 } celdaPila;

typedef celdaPila* tipoPila;
```

```
double entropia_umbral(tipoElementoPila x){

    double arg1 = ((double)(x.vivos_Izq+x.muertos_Izq)/N);
    double arg2 = (x.vivos_Izq/(double)(x.vivos_Izq+x.muertos_Izq));
    double arg3;
    if (x.vivos_Izq > 0.0f) arg3 = log2((double)(x.vivos_Izq+x.muertos_Izq)/x.vivos_Izq);
    else arg3 = 0.0f;
    double arg4 = (x.muertos_Izq/(double)(x.vivos_Izq+x.muertos_Izq));
    double arg5;
    if (x.muertos_Izq > 0.0f) arg5 = log2((double)(x.vivos_Izq+x.muertos_Izq)/x.muertos_Izq);
    else arg5 = 0.0f;
    double arg6 = (((double)(x.vivos_Dch+x.muertos_Dch)/N));
    double arg7 = ((x.vivos_Dch/(double)(x.vivos_Dch+x.muertos_Dch)));
    double arg8;
    if (x.vivos_Dch > 0.0f) arg8 = log2((double)(x.vivos_Dch+x.muertos_Dch)/x.vivos_Dch);
    else arg8 = 0.0f;;
    double arg9 = (x.muertos_Dch/(double)(x.vivos_Dch+x.muertos_Dch));
    double arg10;
    if (x.muertos_Dch > 0.0f) arg10 = log2((double)(x.vivos_Dch+x.muertos_Dch)/x.muertos_Dch);
    else arg10 = 0.0f;

    return ((arg1)*(arg2*arg3+arg4*arg5) + (arg6)*(arg7*arg8 + arg9*arg10));
}
```

PLANTEAMIENTO DEL PROBLEMA

Para abordar este problema y generar el árbol de decisión necesitamos una manera de almacenar esos datos para su siguiente procesamiento.

```
typedef struct dat{
    bool male;
    bool book1;
    bool book2;
    bool book3;
    bool book4;
    bool book5;
    bool isMarried;
    bool isNoble;
    double numDeadRelations;
    double popularity;
    bool isAlive;
}datos;
```

Elegimos un vector de estructuras de tal forma que cada tipo de atributo se adecúe a las funciones mas adelante utilizadas.

El único problema es que se tendrían que crear diferentes funciones para cada uno de los atributos.

Pero conseguimos solucionar ese problema utilizando el poder de las definiciones del lenguaje C.

```
FUNCIONES_SELECCION_DE_CLASE(male, bool);
FUNCIONES_SELECCION_DE_CLASE(book1, bool);
FUNCIONES_SELECCION_DE_CLASE(book2, bool);
FUNCIONES_SELECCION_DE_CLASE(book3, bool);
FUNCIONES_SELECCION_DE_CLASE(book4, bool);
FUNCIONES_SELECCION_DE_CLASE(book5, bool);
FUNCIONES_SELECCION_DE_CLASE(isMarried, bool);
FUNCIONES_SELECCION_DE_CLASE(isNoble, bool);
FUNCIONES_SELECCION_DE_CLASE(numDeadRelations, int);
FUNCIONES_SELECCION_DE_CLASE(popularity, double);
```

HEURÍSTICA

- Árboles de decisión en base a tres criterios distintos
  - Máxima ganancia (C4.5)
  - Índice Gini (CART)
  - Índice Chi² (CHAID)

MÁXIMA GANANCIA

- Se calcula la ganancia de todas las clases
- Separación en base a la clase con mayor ganancia

$$\boldsymbol{Gain(A) = E(C) - E(A)}$$

$$E(C) = \sum_{j=1}^{Clases} -\frac{n_j}{n}\log_2\left(\frac{n_j}{n}\right)$$

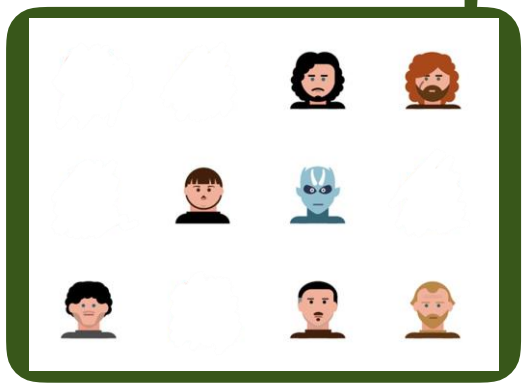
$$E(A) = \sum_{i=1}^{particiones} \frac{n_i}{n} E(nodo_i)$$

ÍNDICE GINI

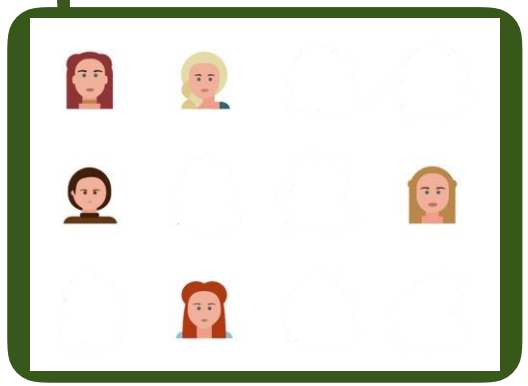
$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

- Cálculo Gini para Male = $(0.4286)^2+(0.5714)^2 = 0.5102$
- Cálculo Gini para Female = $(0.4)^2+(0.6)^2 = 0.52$
- Total= (7/12) * 0.5102 + (5/12) * 0.52 = 0.5143

Total = 12
Vivos = 5
%Vivos = 41,67%



Male = 7
Vivos = 3
%Vivos = 42,86%

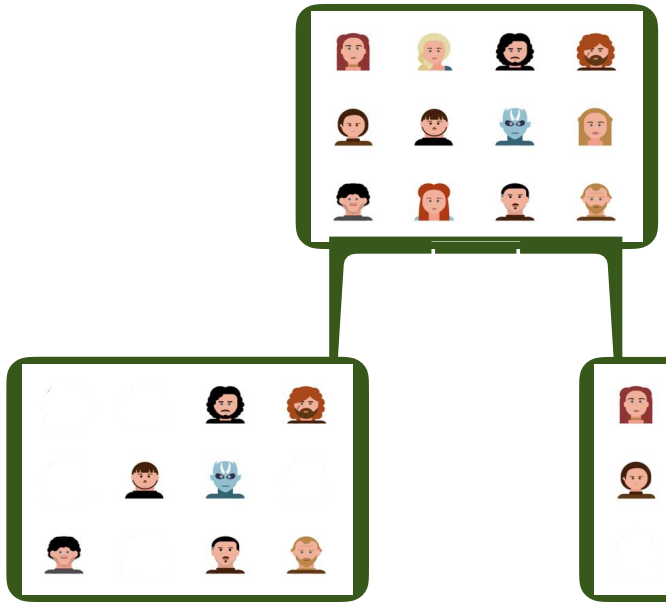


Female = 5
Vivos = 2
%Vivos = 40%

CHI²

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

- Cálculo de % de personajes vivos y no vivos en el nodo female
- Exp = Cálculo del valor esperado de personajes mediante el porcentaje total (41,67%) por el nº de personajes en el nodo female (5)
- Dev = Cálculo de la desviación mediante el nº de personajes en el nodo female (5) – Exp
- Para Chi²= $\sqrt{\frac{Dev^2}{Exp}}$



Nodos	Alive	Not Alive	Total	Expected Alive	Expected Not Alive	Deviation Alive	Deviation Not Alive	Chi-Square	
								Alive	Not Alive
Female	2	3	5	2.9166	2.9166	0.0833	1.0833	0.5367	0.04879
Male	3	4	7	2.9166	2.9166	-0.9166	0.0833	0.04879	0.63433
								Total Chi-Square	1.2686

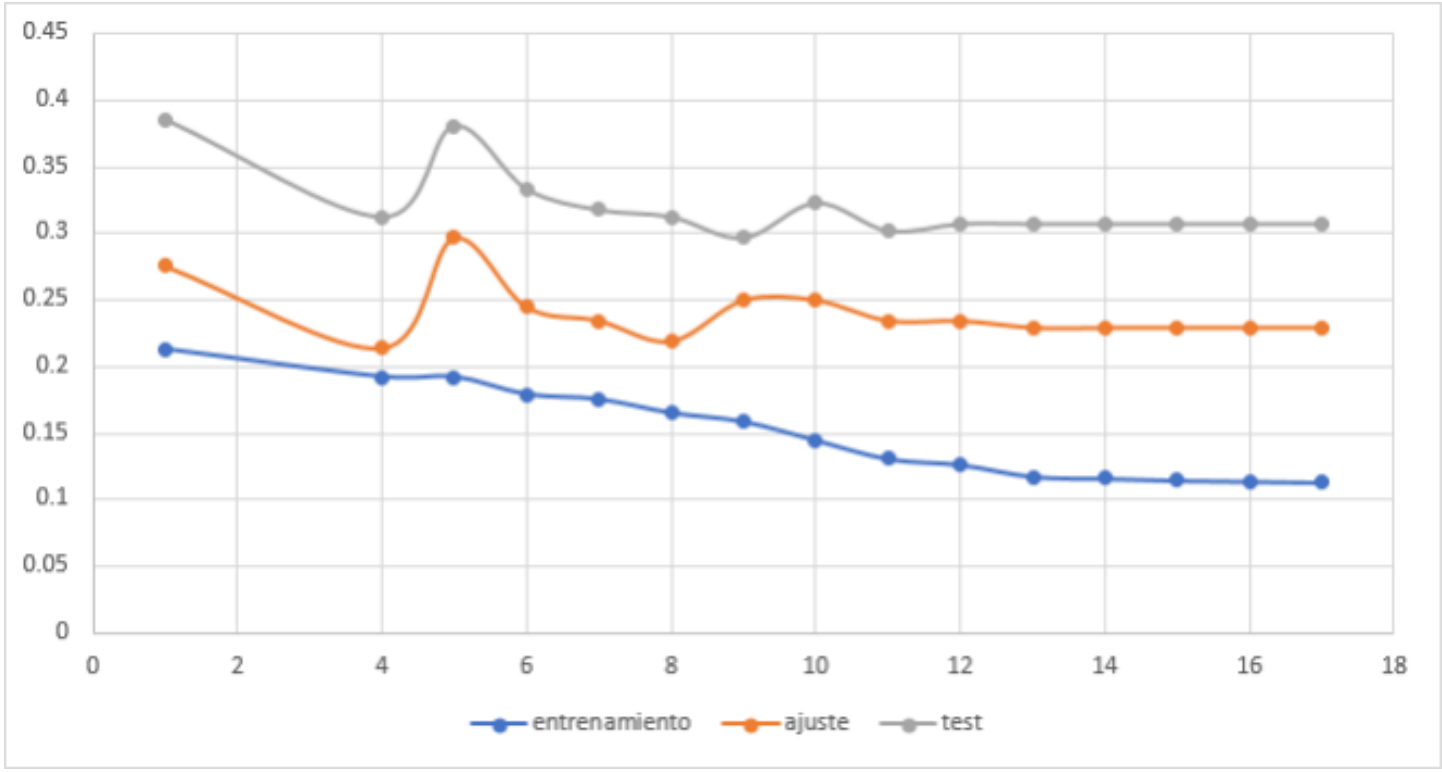
NUESTRO ÁRBOL

```
crearArbolDecision( arbolBinario: a, datos: vector_de_datos, int: tamaño, double: entropíaMínima, int: profundidad, int: profundidadMáxima)
   profundidad++;
   Entropía = calcular_entropia_actual;

   si tamaño == 0 o entropía < entropiaMínima o profundidad == profundidadMáxima{
      crearHoja;
      asignar_vivo_o_muerto;
   }

   si no{
      atributo_elegido = calcular_máxima_ganancia;

      para i = 0 hasta tamaño hacer{
         si vector_de_datos[i] < atributo_elegido{
            vector_hijo_izquierdo[i] = vector_de_datos[i];
         }
         si no{
            vector_hijo_derecho[i] = vector_de_datos[i];
         }
         fsi
      fpara
      crearArbolDecision(a->izquierdo, vector_hijo_izquierdo, tamaño_vector_hijo_izquierdo, entropíaMínima, profundidad, profundidadMáxima);
      crearArbolDecision(a->derecho, vector_hijo_derecho, tamaño_vector_hijo_derecho, entropíaMinima, profundidad, profundidadMáxima);
      }
   }fsi
ffunción
```

GRÁFICAS

Máxima ganancia

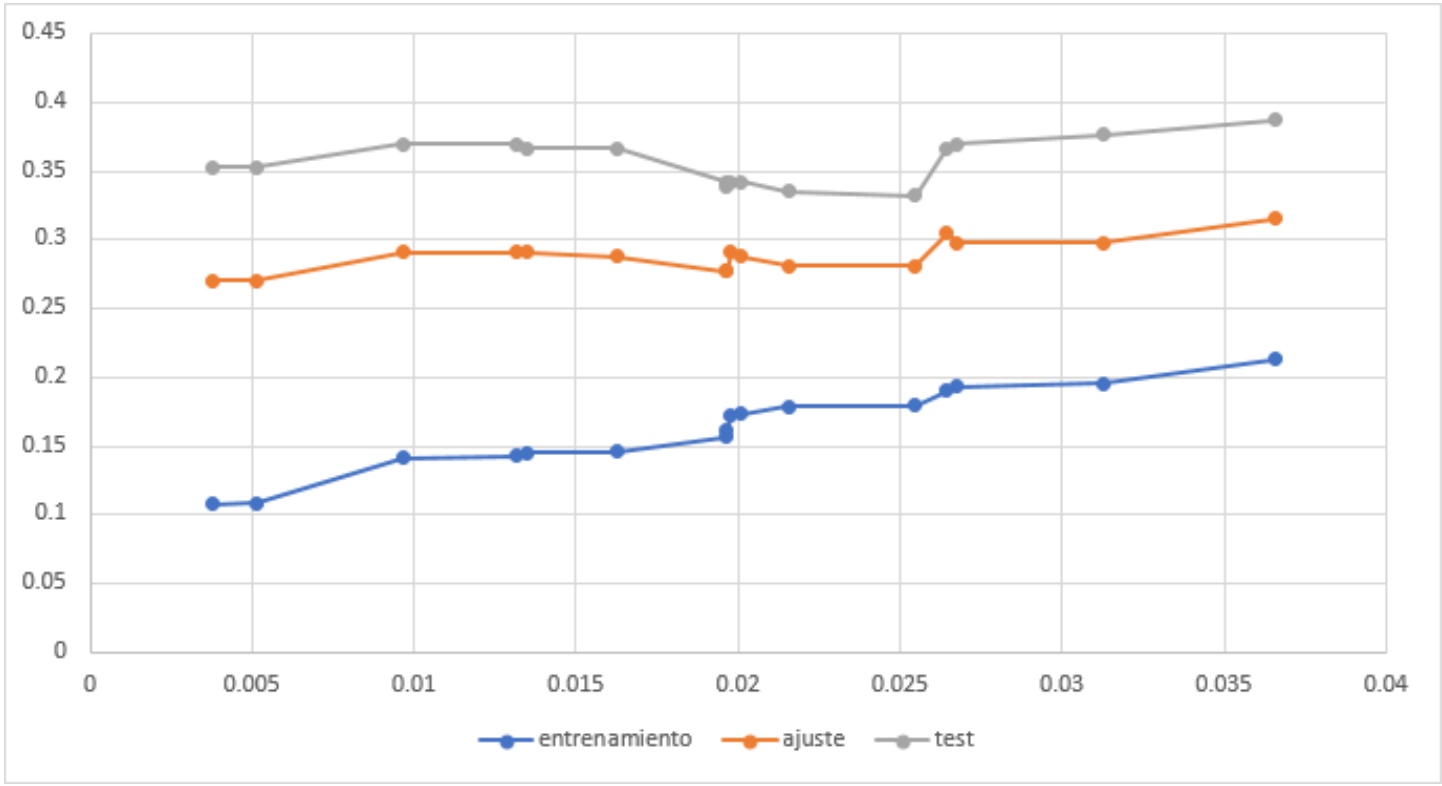
- Clasificado por ganancia
- Podado por profundidad
- Mejor árbol: profundidad 4
- Aciertos reales: 68,75%



GRÁFICAS

Máxima ganancia

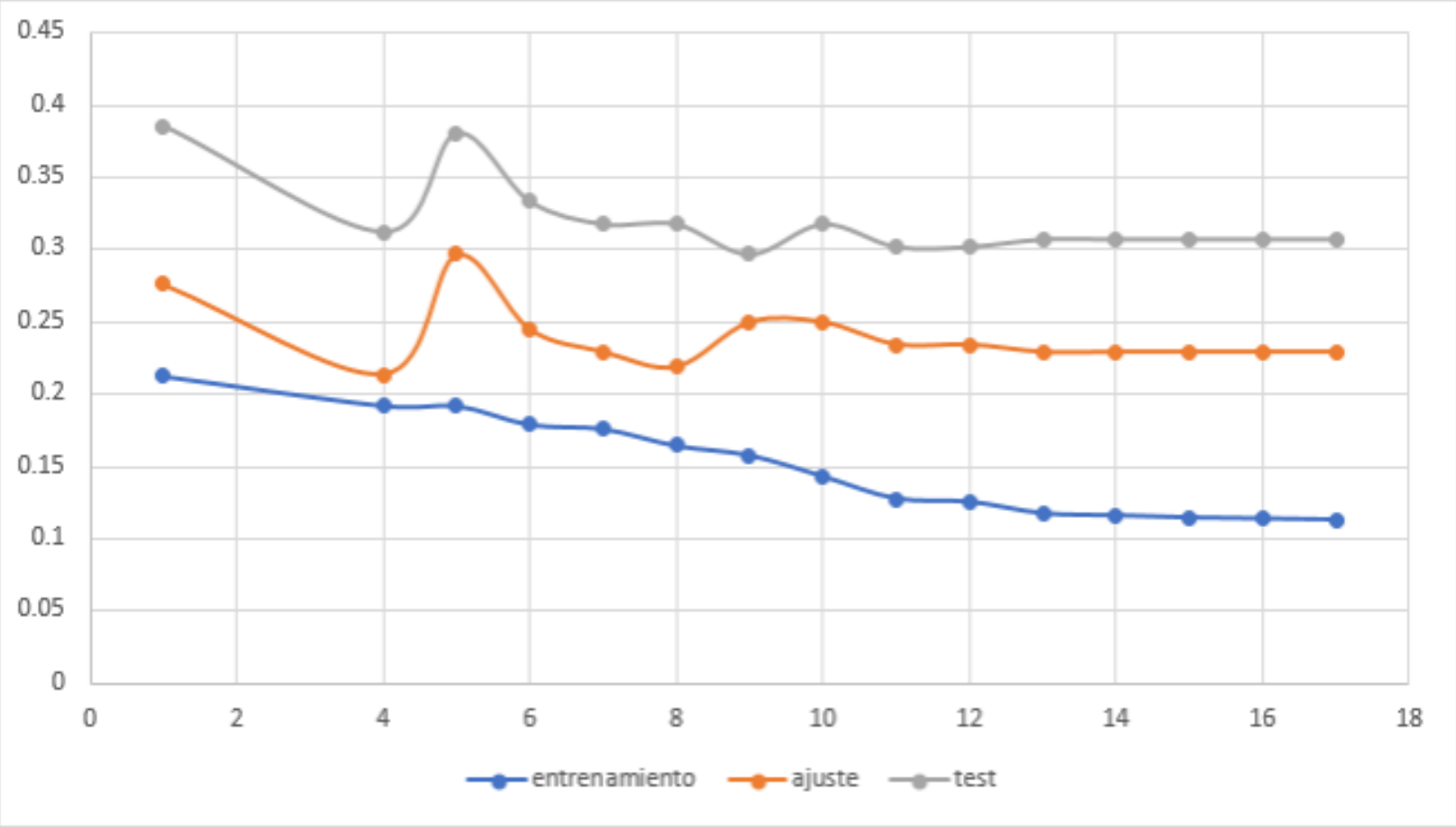
- Clasificado por ganancia
- Podado por ganancia máxima
- Mejor árbol: entropía 0.02547
- Aciertos reales: 66,78%



GRÁFICAS

Índice Gini

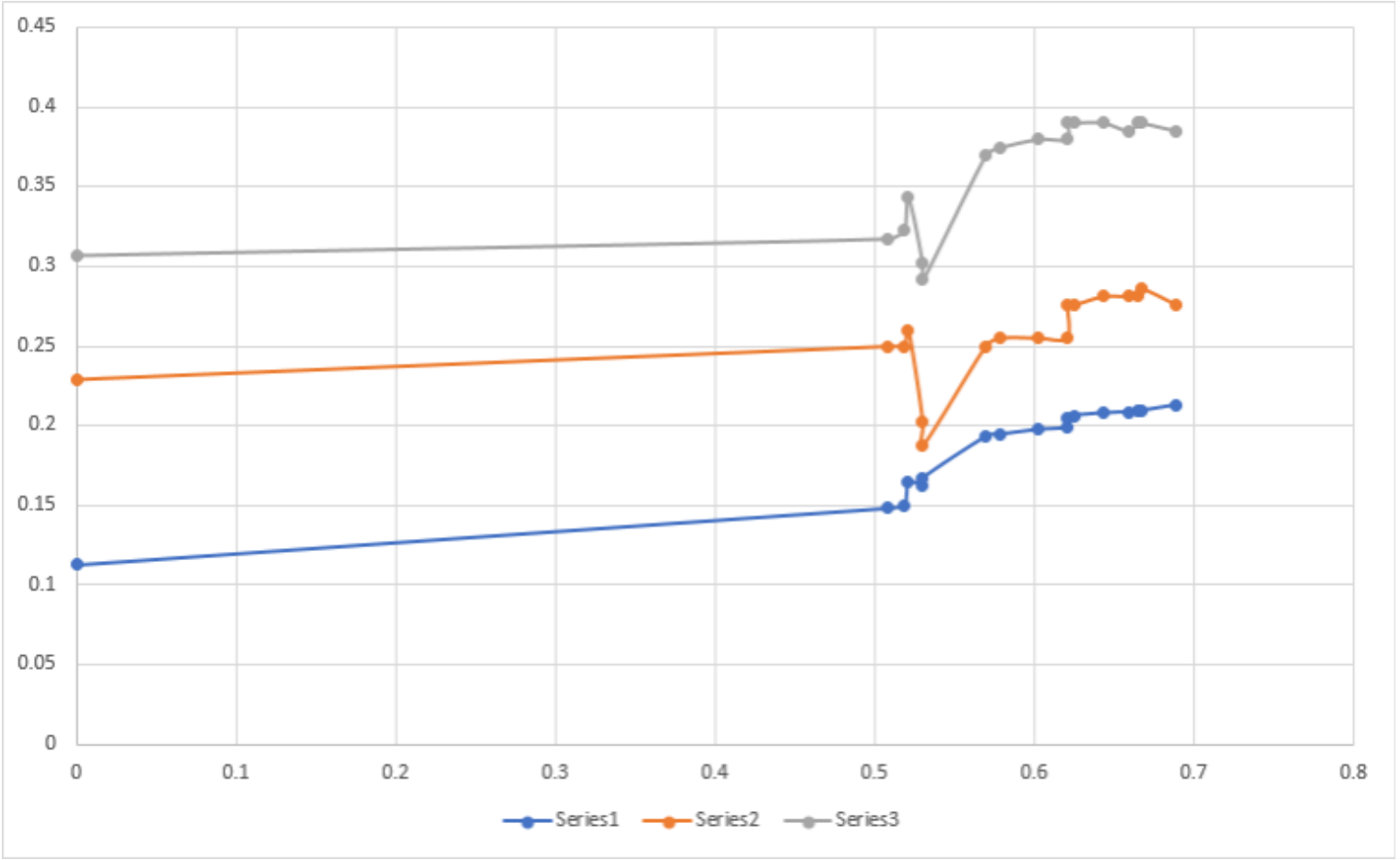
- Podado por profundidad máxima
- Eje X: profundidad
- Mejor árbol: profundidad 4
- Aciertos reales: 68,75%



GRÁFICAS

Índice Gini

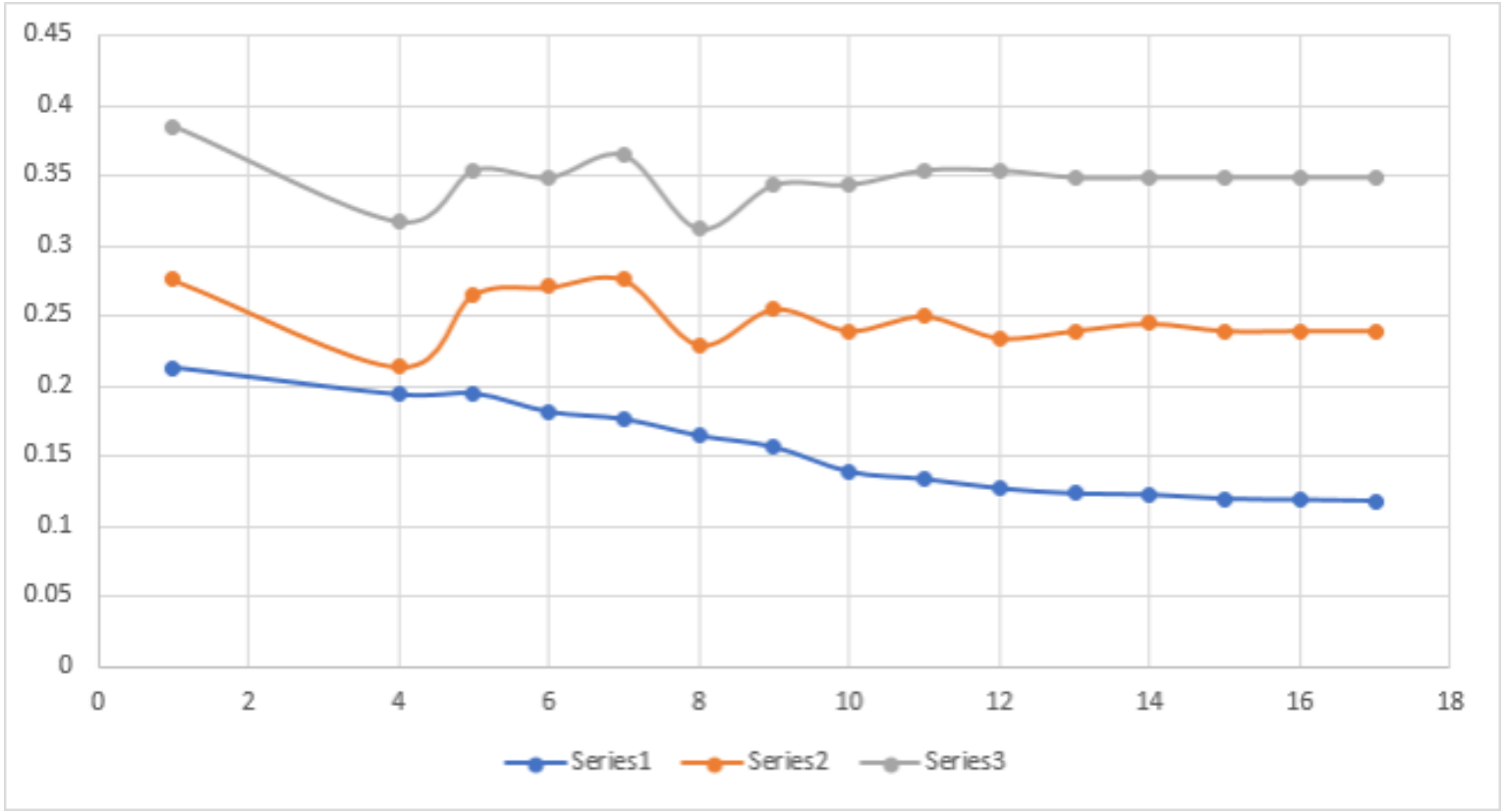
- Clasificado por índice de Gini
- Podado por índice de Gini máximo
- Mejor árbol: gini 0.53005
- Aciertos reales: 70,83%



GRÁFICAS

Índice Chi²

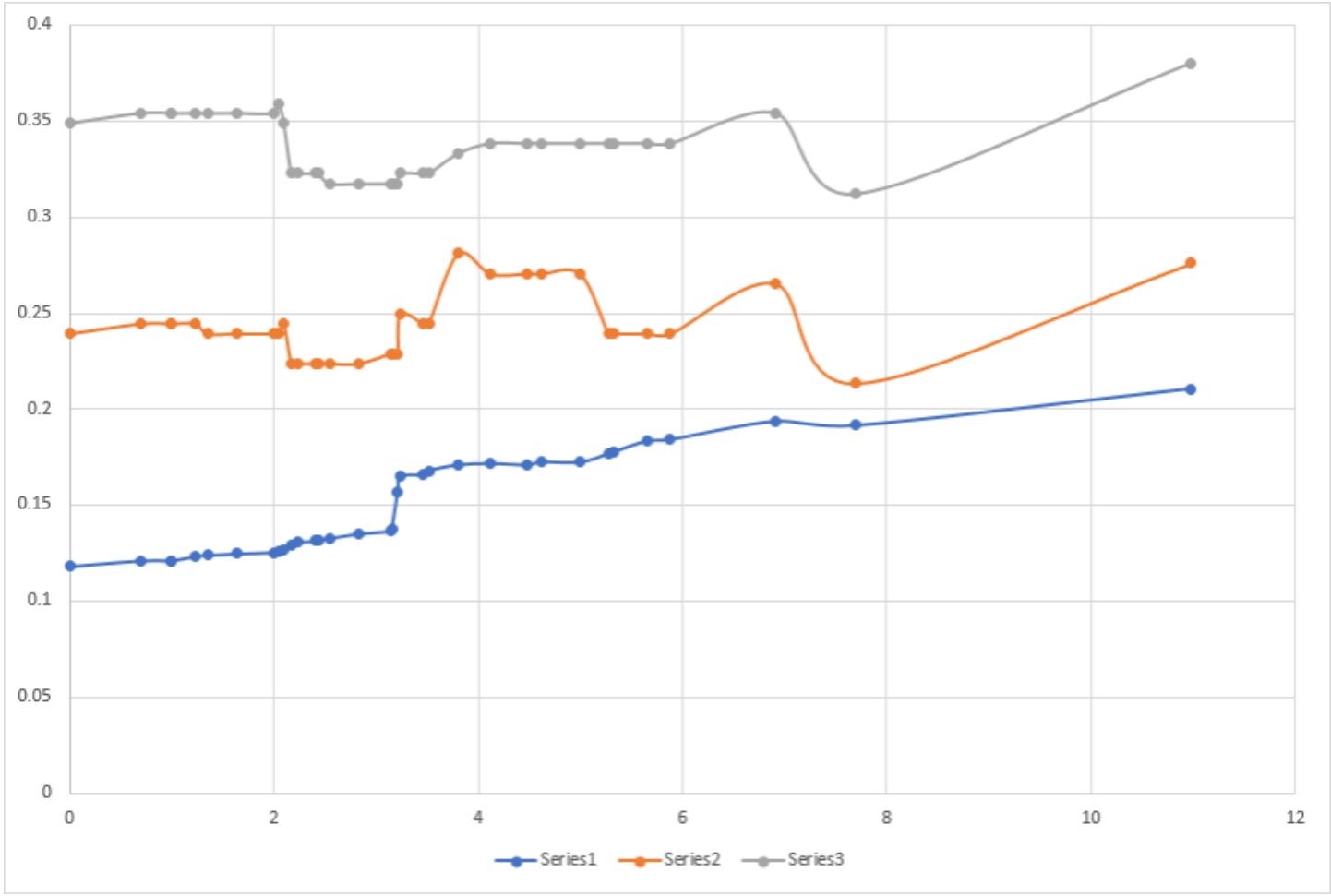
- Clasificado por índice Chi²
- Podado por profundidad máxima
- Mejor árbol: profundidad 4
- Aciertos reales: 68,23%



GRÁFICAS

Índice Chi²

- Clasificado por índice Chi²
- Podado por Chi² máximo
- Mejor árbol: chi² 7.7012
- Aciertos reales: 68,75%



DIFICULTADES

- Imposibilidad para trabajar con matrices de distintos tipos de datos.
- Los problemas de reserva de memoria.
- La necesidad de *cast* en la mayoría de operaciones matemáticas debido a criterios de C.

CONCLUSIONES

El mejor árbol es el de profundidad 4 en cualquiera de los tres métodos utilizados

REFERENCIAS

- https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134
- http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/
- https://www.youtube.com/watch?v=LDRbO9a6XPUhttps://www.youtube.com/watch?v=LDRbO9a6XPU