Clasificación con árboles mixtos

César García Pascual
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
cesgarpas@alum.us.es

Julián Carrascosa Cosano
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
julcarcos@alum.us.es

***Resumen*— El objetivo principal del trabajo es realizar un algoritmo modificado de ID3 que cree un árbol a partir de un parámetro llamado "quórum". Este parámetro determina, al ramificar, el número mínimo de ejemplos con el cual seguir clasificando y, en caso de ser menor, crear un nodo que realice una llamada a Naive Bayes para que este clasifique el ejemplo.**

El objetivo del trabajo es ver la repercusión de este parámetro quórum en la clasificación una vez se ha creado el árbol mixto con el algoritmo modificado.

Tras completar el trabajo y realizar las pruebas se ha observado que, para los conjunto de datos utilizados, la precisión del árbol empeora a mayor es el quórum.

Palabras Clave—Inteligencia Artificial, ID3, Naive Bayes, Árbol de decisión, Aprendizaje supervisado, Clasificación, Suavizado de Laplace

I. Introducción

El término inteligencia artificial [1], puede ser aplicado cuando una máquina imita las funciones cognitivas propias de un ser humano, como puede ser aprender o resolver problemas.

Para desarrollar este trabajo, se ha usado ID3 [2] y Naive Bayes[3], que son dos tipos de aprendizaje supervisado [4].

Los sistemas de aprendizaje supervisados son aquellos en los que, a partir de un conjunto de ejemplos de los que conocemos su valor objetivo (conjunto de entrenamiento), ya sea como clasificación o como regresión, intentamos encontrar una función que permita asignar un valor objetivo a ejemplos que el sistema no ha visto anteriormente.

Todo esto pertenece a la rama de la inteligencia artificial aprendizaje automático [5], la cual es la rama de la Inteligencia Artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender.

Como se explicará a continuación, el algoritmo ID3 genera un árbol de decisión a partir de los ejemplos dados en el conjunto de entrenamiento. Como modificación el algoritmo recibe un valor llamado quórum, el cual hace que cuando se va ramificando el árbol, si se da un caso en que los ejemplos restantes del conjunto de entrenamiento son menores a este valor, la rama del árbol se truncará y se creará un nodo el cual realice una llamada a Naive Bayes. Por tanto en el árbol existen tres tipos de nodos según los criterios impuestos:

- Nodos interiores, cada uno con su atributo asociado.
- Nodos hoja-categoría, devolviendo el valor de la clasificación.
- Nodos hoja-truncada, que se forma cuando el quórum impuesto no se cumple y no se puede seguir desarrollando el algoritmo por ID3 , por lo que se hace una llamada a Naive Bayes.

Todo esto provoca que el algoritmo tenga dos tipos de salidas:

- Un valor de clasificación sin incertidumbre, si es el caso de una rama sin truncar
- Un valor de clasificación con una probabilidad asociada si la rama está truncada.

El objetivo del trabajo es ver cómo influye el parámetro quórum, usado para la creación del árbol, a la hora de clasificar el conjunto de pruebas.

II. Preliminares

A. Métodos empleados

ID3 es un tipo de aprendizaje supervisado, donde dado un conjunto de ejemplos se crea un árbol de decisión. El objetivo del arbol es, para un nuevo ejemplo, clasificarlo según sus atributos.

- El conjunto de ejemplos dados estará formado por una serie de tuplas de valores, denominados atributos, donde el atributo objetivo a clasificar, será de tipo binario (Sí o no, positivo o negativo..)
- El algoritmo, obtiene las hipótesis para clasificar nuevos ejemplos, y determina si dicho ejemplo es positivo o negativo (La respuesta depende del valor atributo objetivo del conjunto)
- Este desarrollo ID3 lo realiza mediante la construcción de un árbol de decisión.

Naive-Bayes es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales.

El teorema de bayes [6] vincula la probabilidad de A dado B con la probabilidad de B dado A.

Naive Bayes asume que la presencia o ausencia de una característica particular no está relacionada con la ausencia de otra característica, por lo que cada característica contribuye de manera independiente a la hora de clasificar un ejemplo.

El problema en el cálculo de las estimaciones es encontrar probabilidades nulas o muy bajas. A veces es debido a la ausencia, en el conjunto de entrenamiento, de algunos valores de atributos en algunas de las categorías; así como también debido por el sobreajuste. Para evitar esto, a la hora de calcular la probabilidad, se puede usar el suavizado de Laplace el cual añade un número entero K que representa el número de ejemplos ficticios que se van a añadir, evitando así probabilidades nulas.

A la hora de elegir qué aprendizaje supervisado de los dos usar, entra el juego el quórum impuesto. El quórum es el mínimo número de ejemplos necesarios para seguir desarrollando el árbol usando ID3 . Al desarrollar el árbol mixto, si una rama no cumple el quórum mínimo, se crea una hoja truncada, la cual al clasificar hará una llamada al algoritmo Naive Bayes y se clasificará el ejemplo usando este método.

III. Metodología

Para la realización del trabajo, hemos seguido por orden los objetivos específicos indicados en el documento del problema creando un código original para la creación del árbol, la clasificación y las pruebas. Se ha creado con Flask[7], un framework minimalista de desarrollo web para Python, y Bulma [8], un framework de estilo web open source, un servidor web el cual provee una interfaz gráfica para obtener resultados de los conjuntos de entrenamiento. Para la visualización de resultados se ha usado la librería matplotlib.pyplot [9] para crear gráficas con los resultados y Graphviz [10] con Pydot [11] para, a partir de un conjunto de datos, dibujar el árbol de decisión.

Para la implementación del primer punto, representación del árbol mixto en Python, y segundo punto, función de clasificación adaptada, decidimos crear dos clases de Python. Para representar los nodos interiores una clase Vertex. Y para las hojas truncadas, una clase NaiveBayes. Para representar las hojas categoría, decidimos usar un String con el valor de la clasificación. Ahora explicaremos las dos clases creadas:

1) Vertex:

Atributos: attribute, children.

- Attribute es el nombre del atributo del nodo clasificado con ID3.
- En children se almacenarán los valores del atributo para clasificar junto a los hijos de este nodo, los cuales pueden ser Vertex, NaiveBayes o un String. Se almacenan en un diccionario con el valor del atributo como clave y el vértice como valor.

Funciones: __init__, classify

- __init__ es el constructor , el cual recibe un diccionario con los hijos y el nombre del atributo.
- Dado un diccionario con los atributos como clave y el valor del atributo a clasificar como valor, classify devuelve, dado primer vértice raíz del árbol, la clasificación de este, ya sea un valor de clasificación sin incertidumbre de una hoja categoría, o una llamada al algoritmo de Naive Bayes el cual devuelve la clasificación con su probabilidad asociada en caso de ser una hoja truncada. Es una recursión, por lo que si se detecta que uno de los hijos es un vértice, se hará una llamada de nuevo a classify para que este clasifique a partir de sus hijos a su vez.

2) NaiveBayes: Atributos: cat1, cat2, cat1_prob, cat2_prob.

- cat1 y cat2 representan el valor binario de toma de decisión (positivo o negativo).
- cat1_prob, cat2_prob almacenan la probabilidad, positiva y negativa por separado, de cada valor de cada atributo del conjunto de datos.

Funciones:__init__, initialize_processed, initialize_raw, classify

- __init__ es el constructor, el cual según el número de valores de entrada (2 o 4) llama a initialize_raw o initialize_processed respectivamente.
- initialize_raw. Recibe la lista de todos los ejemplos y almacena el número ejemplos positivos y negativos, para después calcular la probabilidad asociada a cada atributo.
- initialize_processed. Clasifica un ejemplo a partir de sus probabilidades ya dadas.
- classify. Con los atributos cat1_prob, cat2_prob calcula la probabilidad positiva y negativa del ejemplo dado y devuelve el valor del mayor, junto a los valores de la probabilidad de cada valor binario, clasificando así el ejemplo.

Para la realización del tercer punto de los objetivos, implementar versión modificada del algoritmo ID3 , se ha creado un archivo, id3, con varias funciones:
Funciones: create_tree, recursion_base, recursion_continue, id3_classify_leafs, get_entropy, column_entropy, get_data

- create_tree es la función usada para, a partir de un conjunto de entrenamiento, un porcentaje de entrenamiento y un quórum, crear el árbol y devolverlo. Además ejecuta pruebas con el conjunto de pruebas separado con el porcentaje de entrenamiento y devuelve el ratio de acierto para el árbol, y de cada hoja. El conjunto de entrenamiento lo recibe como un string que get_data se encarga de transformar en una lista de listas, almacenando las filas de un archivo csv. También es capaz de recibir un quórum porcentual en función del total de ejemplos del dataset a parte del valor entero. Un valor booleano shuffle recibe también para barajar los conjuntos antes de ser separados.
  Este hace una llamada al algoritmo de Naive Bayes, y se devuelve el objeto desde el que clasificar. Este se pasa a la función recursion_base junto a la matriz de entrenamiento, el quórum, y unos valores por defecto para la recursión (un array vacío, y un booleano False).
- La función recursion_base es la encargada de decidir si, a partir del conjunto de entrenamiento decidido, el vértice actual va a ser una llamada a Naive bayes (si el número de ejemplos es menor al quórum), una clasificación (si la entropía del conjunto de ejemplos es 0 o recursion_base ha indicado que es una hoja puesto que no hay más atributos para clasificar) o por el contrario será un nodo interior, llamando así a la función recursion_continue y entregandole los valores que ha recibido y, además, el valor de la entropía del conjunto ya calculada.
- Al ser llamado recursion_continue, este calcula los valores de la ganancia para cada atributo no usado anteriormente para clasificar en el árbol y se queda con el atributo que más ganancia tiene. Después, crea un diccionario con los hijos, llamando a el algoritmo recursion_base para que este decida qué tipo de vértice es a partir del subconjunto de cada valor del atributo. Si los atributos que ya se han clasificado son igual al total de atributos del conjunto de entrenamiento, se indica a recursion_base con un booleano que el hijo restante es una hoja y ha de ser clasificado con la función id3_classify_leafs en caso de que el número de ejemplos restantes sea mayor al quórum.
  Estos dos métodos se van llamando recursivamente entre sí hasta llegar a los casos base indicados en recursion_base y recibiendo así los hijos recursion_continue, devolviendo al final del todo el primer vértice raíz del árbol.
- Las función get_entropy devuelve a partir del número de valores positivo y negativo, la entropía asociados a estos. El método entropy_column devuelven ayudándose de get_entropy, a partir de una matriz de datos, la entropía del conjunto.

Para la realización del cuarto objetivo, se amplió la función create_tree para que usase el conjunto de pruebas separado para medir el ratio de acierto de el árbol, separando además el ratio de acierto por tipo de hoja.

Para el quinto objetivo, se creó el archivo testing con los siguientes métodos:

Funciones: test, save_graph

- A partir de un set de entrenamiento, un porcentaje de entrenamiento, un valor booleano indicando si se desea barajar el conjunto al separarse, un quórum mínimo y un quórum máximo sobre los que obtener resultados en los saltos de intervalos deseados y un número de árboles a crear para cada valor del quórum para obtener una media de los resultados por cada quórum, la función test devuelve el ratio de acierto total, el número de hojas que han categorizado cada tipo de hoja y el ratio de acierto de las hojas categoría y truncadas, llamando para cada quórum a la función get_tree explicada en el tercer y cuarto objetivo. Además, con la ayuda de la función save_graph que usa la librería matplotlib.pyplot crea gráficas para estos datos que serán mostrados por el servidor posteriormente.
- A parte de sacar gráficas variando el quórum, es posible para un quórum fijo, variar la K del suavizado de Naive Bayes para ver el ratio de aciertos total y para las hojas truncadas.

Como extra, se ha creado un archivo graphplot el cual a partir de un set de datos, y un quórum, genera a partir del árbol obtenido con la función get_tree y las librerías Graphviz y Pydot una imagen con el árbol dibujado indicando las clasificaciones y con colores el tipo de vértice.

La interfaz gráfica del servidor web facilita el uso de la implementación y la visualización de los resultados, como se mostrará en el apartado anterior.

IV. Resultados

A continuación realizaremos 3 pruebas distintas y detallaremos los resultados conseguidos. Las pruebas serán las siguientes:

1. A partir de un conjunto de datos, un porcentaje para separar el entrenamiento de las pruebas, un valor K, un valor del quórum mínimo, un valor del quórum máximo y el número de árboles a crear por cada quórum para mediar los resultados, obtendremos cuatro gráficas cada una de ellas con el valor del quórum en el eje X y en el eje Y el porcentaje acierto total, el número de hojas para cada tipo y el porcentaje de acierto para cada tipo de hoja en cada gráfica respectivamente.
2. A partir de un conjunto de datos, un porcentaje para separar el entrenamiento de las pruebas, un valor para el quórum, un valor de K mínimo, un valor de K máximo y el número de árboles a crear por cada valor de K para medir los resultados, obtendremos dos gráficas con el valor de la K en el eje X y en el eje Y el porcentaje acierto total, y el porcentaje de acierto para las hojas truncadas.
3. Por último, atendiendo a los resultados obtenidos en el primer apartado, extraemos imágenes de los grafos para observar cuando, dado un valor del quórum están truncados completamente o, con un quórum cero, el árbol está clasificado completamente por ID3.

Apartado 1: Para las pruebas de este primer apartado usaremos en primer lugar el set de datos "Tic-Tac-Toe" (Tres en raya) [12] el cual, dadas las posiciones para cada casilla de las X, los O y las casillas en blanco b, indica si el jugador de las X es vencedor. Para las pruebas usaremos barajado del set y 30 árboles por cada quórum para mediar resultados, K=0, un quórum mínimo de 0 y máximo de 100 en valor entero y saltos de 1. Se usará un 90% del conjunto para entrenamiento y un 10% para las pruebas. Las gráficas obtenidas son las siguientes:

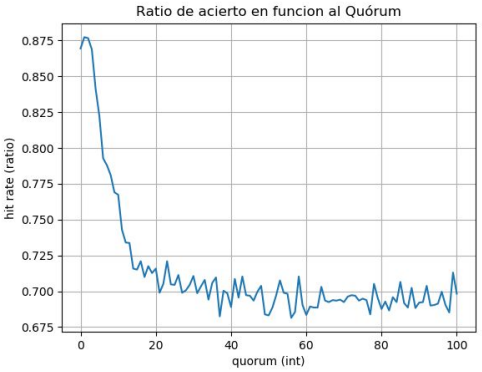

Fig. 1. Ratio de acierto en función al quórum (Tic-Tac-Toe)

Como se puede observar, al principio del todo el ratio de acierto mejora levemente (86% a 88% aproximadamente) pero acto seguido comienza a caer hasta llegar a aproximadamente el valor del quórum = 38 donde se estabiliza alrededor del 69% de aciertos y fluctúa levemente debido a la mediación de los distintos resultados.

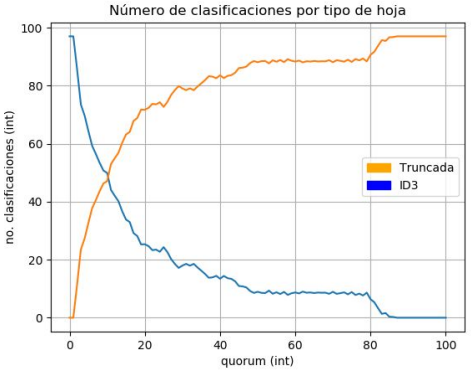

Fig. 2. Número de clasificaciones por tipo de hoja (Tic-Tac-Toe)

Se puede observar que con quórum 0 todas las hojas existentes son hojas categoría (ID3) y que a partir del quórum 2 empiezan a aparecer hojas truncadas. Aproximadamente en el quorum 10 se clasifican el mismo número de ejemplos por cada tipo de hoja y sobre el quórum 85 todas las hojas son truncadas.

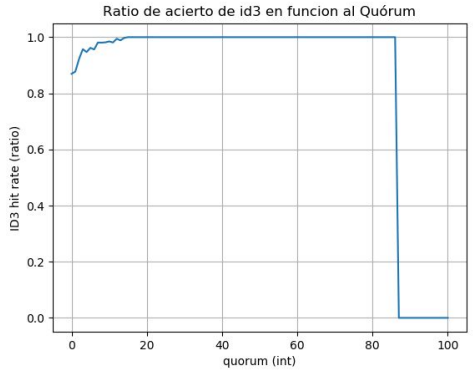

Fig. 3. Ratio de acierto hojas categoría (Tic-Tac-Toe)

Según esta gráfica, se puede observar que el porcentaje de acierto de ID3 al comienzo no es del 100%, y que según las hojas se van truncando, ID3 gana precisión. Al llegar al quórum 17 aproximadamente, el ratio de clasificación es prácticamente 1.

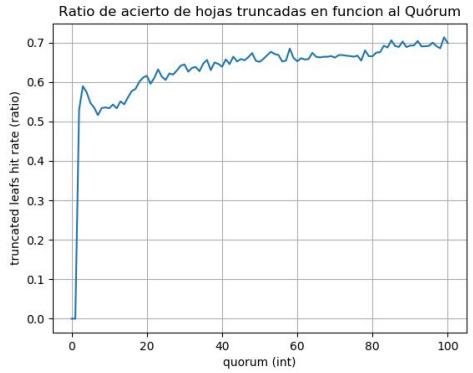

Fig. 4. Ratio de acierto hojas truncadas (Tic-Tac-Toe)

El ratio de aciertos de las hojas truncadas, en quorum 0 y 1 es 0 ya que no habían hojas truncadas pero después se alza teniendo aproximadamente un 60% de aciertos y, según el quórum va incrementando, el ratio de acierto mejora estabilizandose aproximadamente en el quórum 85 que es cuando todas las hojas están truncadas.

Como conclusión de esta prueba, se puede observar que para este set de datos, usar un quórum muy pequeño (Aproximádamente 3) aumenta levemente la fiabilidad de la clasificación pero que para valores más altos la precisión del árbol decrece.

También se puede ver que, el ratio de acierto de las hojas truncadas aumenta a menos hojas categoría haya. Esto puede deberse a que las hojas que se consiguen clasificar por ID3 por tener un número mayor de ejemplos que el quórum son menos homogéneas causando que, si el quórum es mayor que el número de ejemplos restantes, la clasificación se haga mediante Naive Bayes y este obtenga mejores resultados.

Repetiremos de nuevo la prueba pero para otro set de datos. En este caso usaremos el set “Cars” [13] el cual a partir de los atributos de un coche indica si es aceptable o no. Usaremos barajado del set y 30 árboles por cada quórum para mediar resultados, K=0, un quórum mínimo de 0 y máximo de 600 en valor entero y saltos de 10. Se usará un 95% del conjunto para entrenamiento y un 5% para las pruebas. Las gráficas obtenidas son las siguientes:

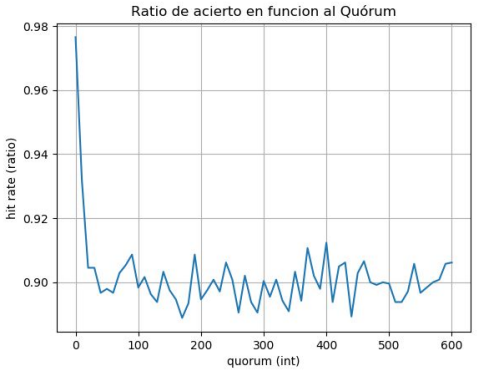

Fig. 5. Ratio de acierto en función al quórum (Cars)

Se puede ver como el ratio de acierto al comienzo no es de 1 y que al llegar al quórum 40 aproximadamente se estabiliza entre 0.89 y 0.91.

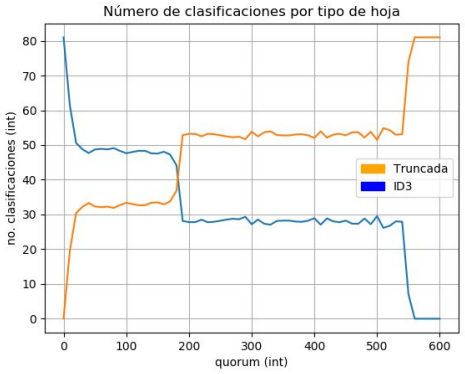

Fig. 6. Número de clasificaciones por tipo de hoja (Cars)

Las hojas desde el 0 hasta el 40 se van intercambiando lentamente y después se estabilizan. En el quórum 180 dan un cambio grande lo cual puede ser debido al truncamiento de una rama grande con varias clasificaciones o que en el quórum 40 quedaran 2 hojas categoría, en el 190 se truncara una de estas y en el 550 se truncara la última restante, siendo todas hojas truncadas.

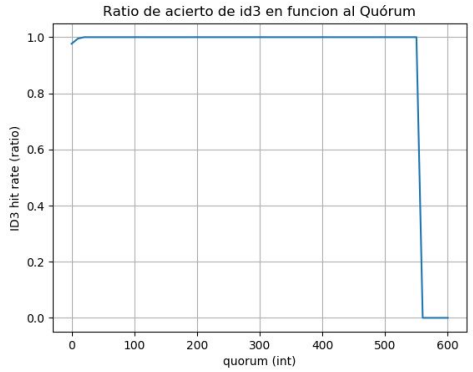

Fig. 7. Ratio de acierto hojas categoría (Cars)

El ratio de acierto de ID3 esta vez es mejor que el anterior ejemplo al principio pero tampoco es 1, mejorando al poco y siendo 1 en el resto de clasificaciones que realiza.

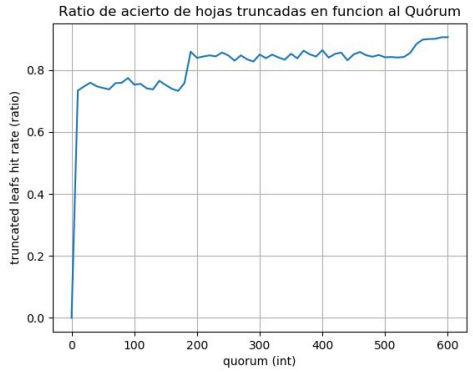

Fig. 8. Ratio de acierto hojas truncadas (Cars)

Como se puede observar, el ratio de acierto de las hojas truncadas según las hojas categoría se truncan, va mejorando al igual que en la prueba anterior.

Para acabar este primer apartado, realizaremos una última vez la prueba para un tercer conjunto de entrenamiento, "KR vs KP" [14] (Rey y torre contra rey y peón). Este indica cuando las blancas pueden ganar y cuando no pueden ganar (Rey y torre). Para ello utilizaremos barajado del set y 30 árboles por cada quórum para mediar resultados, K=0, un quórum mínimo de 0 y máximo de 800 en valor entero y saltos de 10. Se usará un 90% del conjunto para entrenamiento y un 10% para las pruebas. Los resultados son las siguientes:

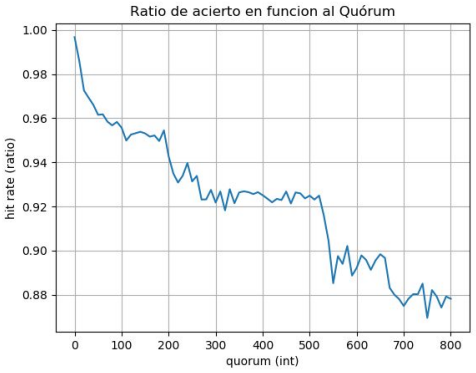

Fig. 9. Ratio de acierto en función al quórum (KR vs KP)

En este último ejemplo se puede ver cómo a diferencia de los dos anteriores, el decremento del porcentaje de acierto es más progresivo pero sigue empeorando a medida que las ramas se truncan.

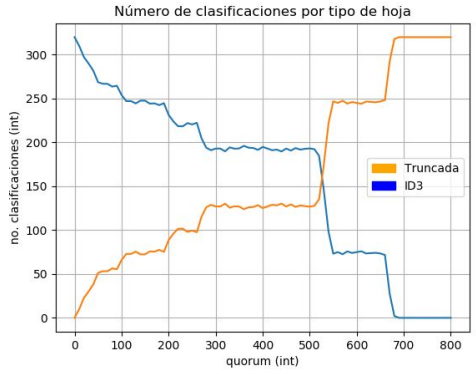

Fig. 10. Número de clasificaciones por tipo de hoja (KR vs KP)

Aquí también puede observarse que, el intercambio de hojas no es tan progresivo como en el primer set pero más escalonado que en el set de los coches.

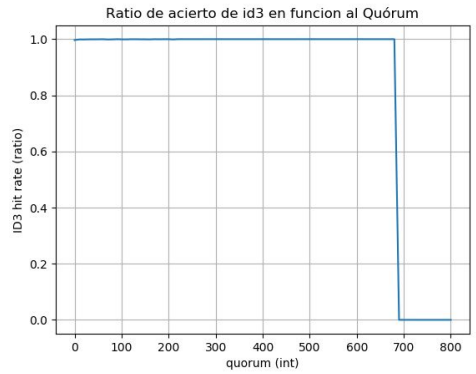

Fig. 11. Ratio de acierto hojas categoría (KR vs KP)

El ratio de acierto de ID3 por poco no es 1 en quorum 0 y puede verse cómo a partir de ahí es 1 hasta el quórum 680

aproximadamente donde no hay más hojas categoría. De los tres sets es el árbol donde ID3 tiene más precisión. Esto puede deberse a que es el set con el mayor número de entradas.

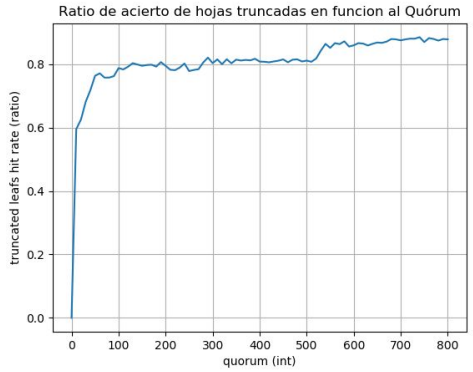

Fig. 12. Ratio de acierto hojas truncadas (KR vs KP)

El ratio de aciertos de las hojas truncadas se estabiliza en quorum 100, 530 y 660 aproximadamente debido al truncamiento pero en general se puede ver como va aumentando su precisión.

Apartado 2: Ahora pasaremos a la realización de la segunda prueba. Para ello usaremos el conjunto de pruebas "Tic-Tac-Toe", barajado del set y 20 árboles por cada K para mediar resultados, quórum = 100% del número de elementos del set de datos, una K mínima de 0 y máxima de 400. Se usará un 90% del conjunto para entrenamiento y un 10% para las pruebas. Al ser el valor del quórum el 100% del número de elementos del conjunto de entrenamiento, el árbol constará de única hoja truncada, siendo el porcentaje de aciertos igual al porcentaje de aciertos de las hojas truncadas. La gráfica resultante es la siguiente:

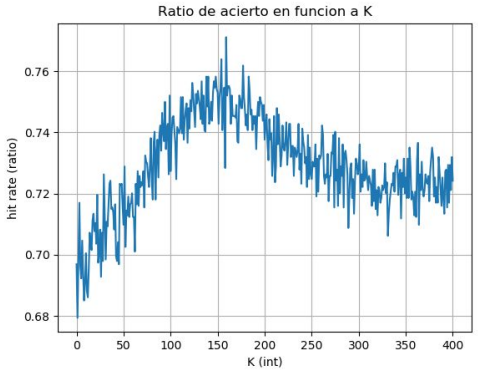

Fig. 13. Ratio de acierto en funcion a K (Tic-Tac-Toe)

Como bien se puede ver, la precisión de Naive Bayes mejora hasta llegar aproximadamente a la K = 150 y, a partir de ahí, comienza a empeorar.

Realizando la misma prueba para un quórum del 5% se puede observar el mismo resultado.

Viendo la mejora significativa de la precisión de Naive Bayes, vamos a realizar de nuevo la primera prueba del primer apartado, pero esta vez con K = 150

Usaremos barajado del set y 30 árboles por cada quórum para mediar resultados, K=150, un quórum mínimo de 0 y máximo de 100 en valor entero y saltos de 1. Se usará un 90% del conjunto para entrenamiento y un 10% para las pruebas. Las gráficas obtenidas son las siguientes:

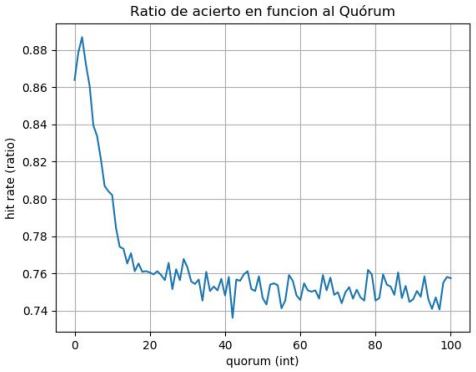

Fig. 14. Ratio de acierto en función al quórum (Tic-Tac-Toe)

Como se puede observar, la gráfica es prácticamente de la misma forma que la de la figura 1, pero ahora se estabiliza aproximadamente en 75%.

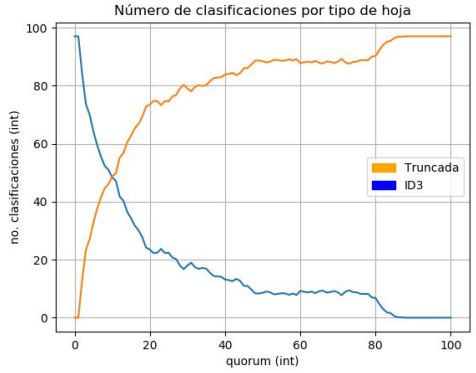

Fig. 15. Número de clasificaciones por tipo de hoja (Tic-Tac-Toe)

Esta gráfica es prácticamente la misma que la figura 2 ya que la K no varía el árbol, sino la precisión de Naive Bayes, no variando el número de hojas de este.

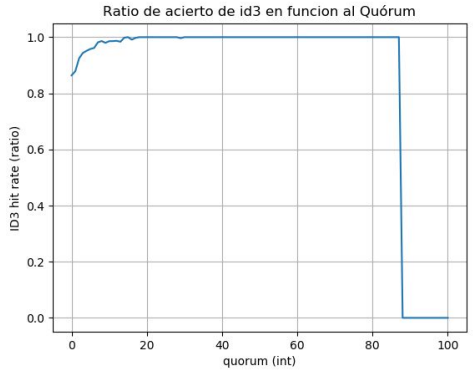

Fig. 16. Ratio de acierto hojas categoría (Tic-Tac-Toe)

Esta es la misma que la figura 3 prácticamente ya que la K no influye en la clasificación de ID3.

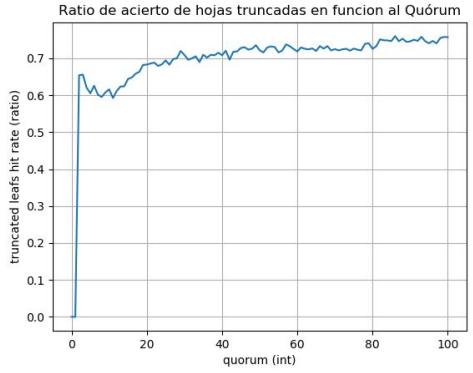

Fig. 17. Ratio de acierto hojas truncadas (Tic-Tac-Toe)

Aquí se puede observar como la gráfica es de la misma forma pero la precisión es entre un 5% y un 10% mayor que en la figura 4, denotando la mejora de precisión de Naive Bayes.

Como conclusión de este apartado, se puede ver que la K mejora la precisión de Naive Bayes para este ejemplo pero no es capaz de sobrepasar la precisión de ID3, por lo que, usar un quórum más bajo sigue entregando un árbol mixto con mejor precisión.

Apartado 3: Por último, realizaremos la tercera prueba donde mostraremos a partir de uno de los sets de datos la imagen de su grafo con distintos quórum, para observar como las hojas se van truncando progresivamente. Usaremos el set de KR vs KP. Se recomienda ver el grafo desde las imágenes adjuntas dado su tamaño, o generar el grafo desde la interfaz de la aplicación desarrollada.

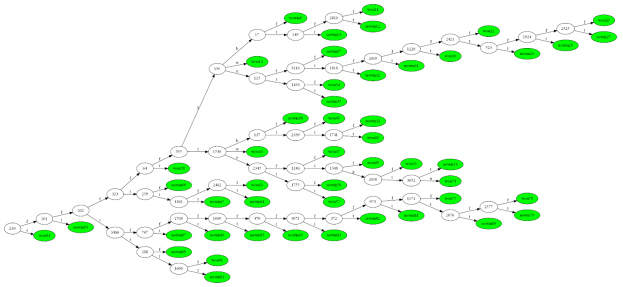

Fig. 18. Grafo con quórum 0 (KR vs KP)

Como bien se puede observar, todas las hojas del grafo son hojas categoría (verde) al tener quórum 0.

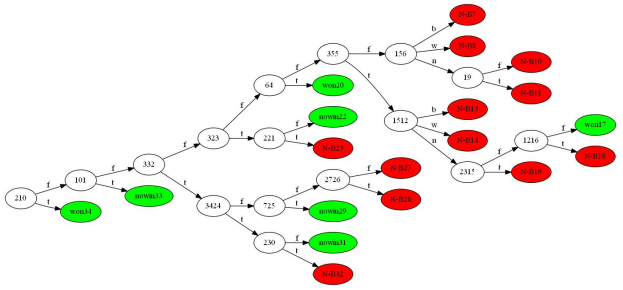

Fig. 19. Grafo con quórum 100 (KR vs KP)

Se puede ver como el grafo se ha truncado bastante pero aún quedan hojas por truncar.

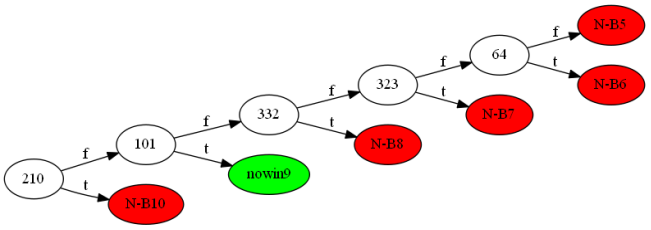

Fig. 20. Grafo con quórum 600 (KR vs KP)

Aquí se puede observar cómo se han truncado la mayoría pero aún queda una hoja categoría. En los resultados de la primera prueba de este set se podía ver como el número de resultados categorizados por cada tipo de hoja daban saltos grandes, y, comparando con los resultados, en cuanto la hoja nowin9 desaparezca todas las hojas serán categorizadas por bayes, como bien se observa en el escalón final de la figura 10.

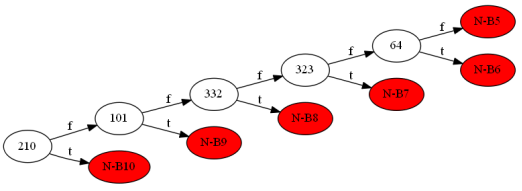

Fig. 21. Grafo con quórum 750 (KR vs KP)

En este quórum las hojas categoría ya no existen, al haberse truncado la hoja nowin9.

V. Conclusiones

Como conclusiones finales, para los conjuntos de entrenamiento usados, a menor es el quórum, mejor es la precisión del árbol generado. ID3 ha sido más preciso clasificando que Naive Bayes.

También se observa que, para quórums pequeños la precisión de las hojas truncadas y que, según aumenta el quórum y van apareciendo más de estas, la precisión de las clasificaciones de las hojas truncadas aumenta. Esto puede deberse a que, las hojas que se consiguen clasificar con ID3, son hojas con una mejor información para clasificar que las que sos truncadas al principio. Según las hojas se van truncando, estos ejemplos son clasificados con Naive Bayes, mejorando su precisión al ser ejemplos más completos dentro del conjunto de entrenamiento a la hora de clasificar.

En el segundo apartado vimos que para un determinado conjunto de datos, se puede obtener una K que aumenta la precisión de Naive Bayes.

Aplicando esta K a la primera prueba del primer apartado mejoró la precisión del árbol pero mantuvo la misma forma, siendo mejor utilizar el quórum más pequeño dada la precisión tan alta de ID3.

En el tercer apartado pudimos ver que a mayor es el quórum, las hojas se van truncando, desapareciendo las hojas categorías y también, hojas truncadas, al truncarse la rama a la que pertenecen..Además, fijándonos en la figura 10, los saltos que aparecían en la gráfica se debían a que existía una única hoja sin truncar y que, al truncarse, el salto de las clasificaciones por hojas se producía.

No queda demostrado que a menor es el quórum mayor es la precisión del árbol ya que, es posible que exista un conjunto en el que ocurra lo contrario pero, en los tres conjuntos utilizados, se aplica esta regla.

Como ampliación del trabajo, se pueden utilizar más conjuntos para ver si se sigue aplicando lo mismo visto en las pruebas anteriores o si, por el contrario, aparecen patrones distintos.

También se puede implementar un algoritmo de poda sobre el árbol mixto, para ver cómo varían los resultados.

Referencias

[1] Wikipedia Inteligencia Artificial https://es.wikipedia.org/wiki/Inteligencia_artificial 23/06/2019.

[2] Wikipedia ID3 Algorithm. https://en.wikipedia.org/wiki/ID3_algorithm 23/06/2019.

[3] Wikipedia Naive Bayes Classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier 23/06/2019

[4] Aprendizaje supervisado http://www.cs.us.es/~fsancho/?e=77 23/06/2019

[5] Aprendizaje Automático http://www.cs.us.es/~fsancho/?e=75 23/06/2019

[6] Wikipedia Teorema de Bayes https://es.wikipedia.org/wiki/Teorema_de_Bayes 23/06/2019

[7] Flask http://flask.pocoo.org/docs/1.0/ 25/06/2019

[8] Bulma https://bulma.io/ 23/06/2019

[9] Matplotlib https://matplotlib.org/ 23/06/2019

[10] Graphviz https://www.graphviz.org/ 23/06/2019

[11] Pydot https://pypi.org/project/pydot/ 23/06/2019

[12] Tic Tac Toe dataset, https://data.world/uci/tic-tac-toe-endgame 20/06/2019

[13] Cars dataset, https://archive.ics.uci.edu/ml/datasets/Car+Evaluation 20/06/2019

[14] KR vs KP dataset, https://data.world/uci/chess-king-rook-vs-king 20/06/2019