Trabajo Práctico N°1. Aprendizaje Automático
Maestría en Data Mining y Descubrimiento de conocimiento

Autores: Diego Dell'Era - Miguel Ángel Barros

Objetivo

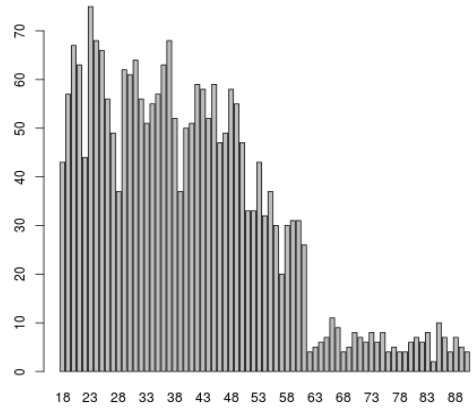
Analizar los resultados obtenidos mediante la utilización del algoritmo J48 y la variación de sus parámetros.

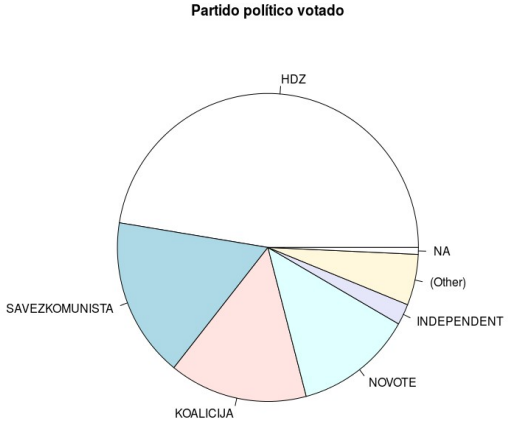
Materiales y Métodos

Dataset

El dataset utilizado se obtuvo del repositorio del Leibniz Institute for the Social Sciences (GESIS). Corresponde al relevamiento de los votantes en las elecciones democráticas llevadas a cabo 1994 y su comportamiento en dichos comicios. Contiene 2360 instancias con 13 atributos, de los cuales uno es numérico ("age") y el resto son nominales. Siete de los atributos nominales son de tipo binario.

La muestra está equilibrada en género (aprox. iguales cantidades de hombres y mujeres). Para las edades parece haber dos clases, con cantidades dispares: mayores/menores de 60 años. La mayoría de las variables binarias (yes/no) tienen un claro sesgo por una de las dos categorías. Finalmente la variable objetivo de la clasificación tiene un claro ganador: HDZ.





Software

Para las tareas de discretización se utilizaron scripts de Python. Para los gráficos y la sumarización de la información, R. Para la clasificación, Weka.

Resultados

Sobre-ajuste y poda

La Figura 1 compara la performance del árbol de decisión como clasificador tanto en el dataset reservado para la validación (testing, 20%) como en el conjunto usado para entrenamiento. Sobre los datos de testing, el clasificador se mantuvo de manera regular hasta alcanzar un pico máximo de 52% de accuracy. A partir de ese punto el algoritmo exhibe el efecto conocido como sobre-ajuste (overfitting). Como se nota en el gráfico, este efecto está representado por un aumento en la precisión sobre los datos de entrenamiento y simultáneamente una disminución en la precisión sobre los datos de testing, para un confidence factor por encima de 0.2. El error de decisión se propaga de una manera notable conforme aumenta, en número de hojas, el tamaño del árbol (Figura 2).

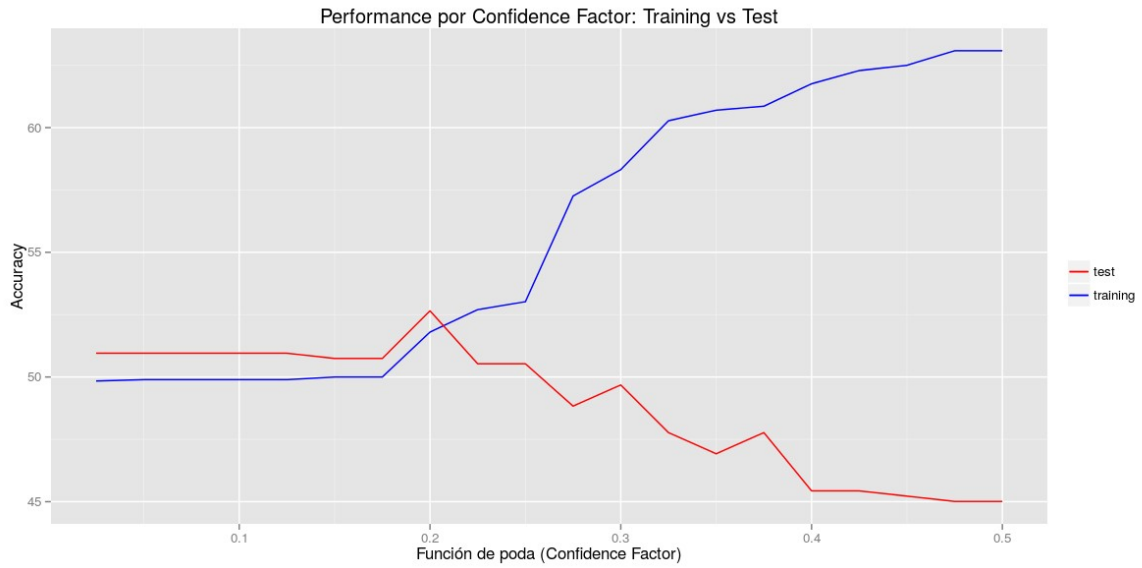


Figura 1: Precisión del clasificador en los datasets de training y testing

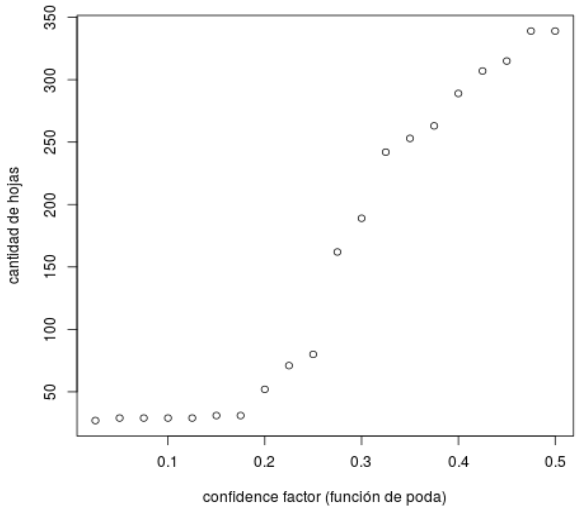


Figura 2: Tamaño del árbol (cantidad de hojas y nodos) en función de la poda

Datos faltantes

Los datos faltantes se completaron con dos estrategias: rellenando con la moda, o con la moda por clase. Para determinar a qué variable imputarle valores faltantes, se procedió a obtener la *information gain* de cada atributo. Puesto que la information gain mide cómo un cierto atributo reduce la entropía a la hora de clasificar los ejemplos, se desea saber el efecto de valores faltantes en la variable que mayor certeza aporte. En nuestro conjunto de datos, ese atributo es ”member_pol_Party”:

```
Evaluator:     weka.attributeSelection.InfoGainAttributeEval
 0.14343   3 MEMBER_POL_PARTY
 0.06536   2 AGE
 0.02513  11 STATE_AND_MASS_MEDIA
 0.0177    4 INFO_PRESS
 0.0144    7 INFO_PARTY_PRESS
 0.01249  12 RESULT_INFLUEN_VOTING
 0.01237   6 INFO_POL_MEETING
 0.00889   9 INFO_CONVERSATION
 0.00787  10 INFO_NOT_INTERESTED
 0.0062    5 INFO_TV_RADIO
 0.00577   8 INFO_ELECT_PUBLICATION
 0.00512   1 GENDER
```

Los datos faltantes se imputaron de 0 a 85% en intervalos que fueron de 0 a 34. El algoritmo fue entrenado sobre el 80% del conjunto de datos, reservando un 20% para testing, incrementando el confidence factor en intervalos de 2.5.

Los resultados obtenidos sugieren que el algoritmo presenta una cierta robustez ante los datos faltantes, porque la precisión presenta una dispersión bastante ajustada a la curva

original: el punto a partir del cual la performance sube/cae para training/testing respectivamente es el mismo, un confidence factor de 0.2 (Figura 3).

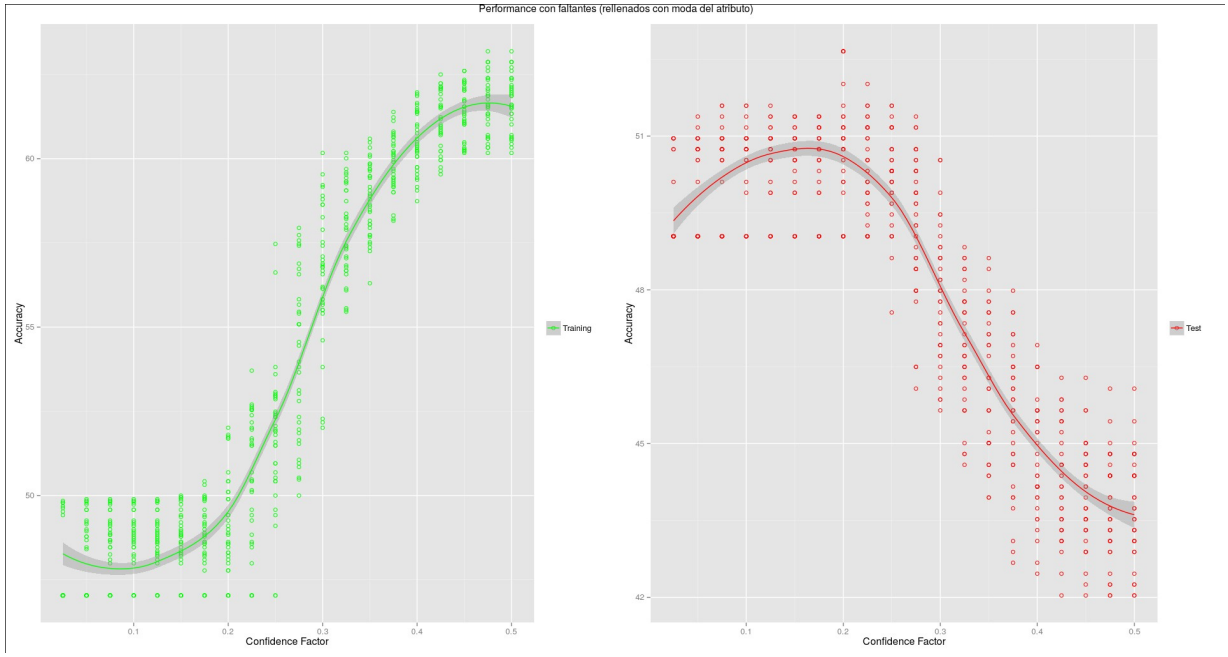


Figura 3: Representación del porcentaje de accuracy vs confidence factor en el set de training (izquierda) y set de test (derecha)

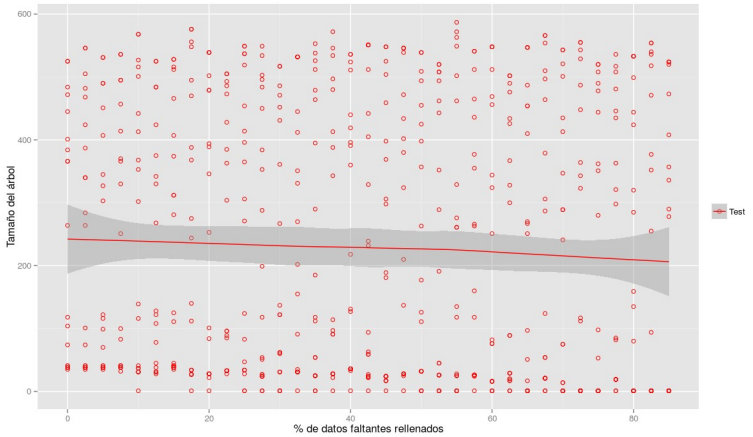


Figura 4: Tamaño en función de los datos faltantes

Los datos faltantes tuvieron otro efecto en el tamaño de los árboles: a medida que aumentaba el porcentaje de relleno, la clase del partido político mayoritario se volvía más probable, más allá de cuáles fueran los atributos particulares del votante. El árbol aprendía rápidamente a limitarse a *una* hoja: asignar el partido ganador, HDZ. Eso explica

un tamaño medio de árbol un poco más bajo que con el dataset original, y los puntos que se agolpan en la base de la Figura 4.

Discretización

Algunos algoritmos de data mining, y en particular ciertos algoritmos de clasificación requieren que los datos sean de tipo categóricos. Por lo tanto es necesario transformar, en el caso de que existan, los atributos continuos en categóricos. A esta modificación se la denomina discretización. Esta transformación de un atributo continuo a uno categórico incluye dos sub-tareas. En primer lugar, decidir cuantas categorías se van a implementar. En segundo lugar, determinar cómo asignar los valores continuos dentro de cada una de las categorías generadas previamente. Para lo cual es importante establecer, en los valores que se van a discretizar, los ***n*** intervalos especificando los ***n-1*** puntos de *split* así como también que todos los valores que caen dentro de los intervalos generados en la etapa anterior sean asignados al mismo valor categórico. Estos intervalos se denominan *bins* o *buckets*.

Hay dos metodologías para la construcción de los *bins*, según tengan o no en cuenta la información de la clase: estrategias supervisadas y no supervisadas, respectivamente. Para la estrategia no supervisada se usaron dos métodos: 1) *bins* de igual frecuencia (es decir, *bins* con igual cantidad de observaciones), y 2) *bins* de igual ancho (es decir, *bins* que dividen al total de valores en intervalos de igual tamaño). Al igual que los puntos anteriores, se reservó el 20% del conjunto de datos para validación.

Las figuras 5a*,* 5b y 5c muestran el rendimiento del clasificador en función de la estrategia de discretización en los datasets de training y test. Se observa que, con las estrategias no supervisadas, la precisión del clasificador presenta una curva similar a la observada para los valores originales no discretizados, con un amplitud un poco menor: si bien nunca alcanza la performance original sobre el dataset de training, eso no empeora su rendimiento sobre el dataset de testing (lo que indica que la discretización evita en cierta medida el overfitting).

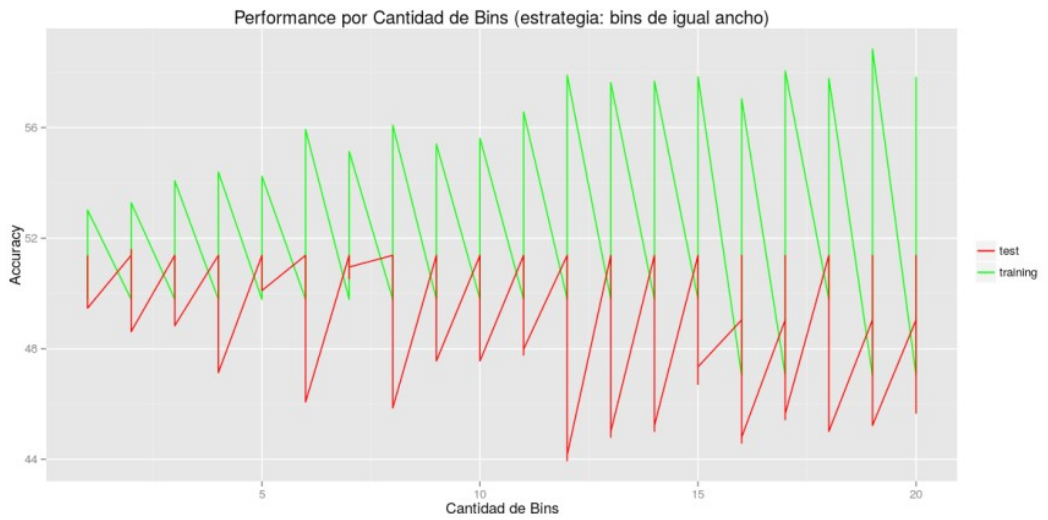


Figura 5a: Discretización no supervisada,* bins *de igual ancho (densidad): precisión del clasificador

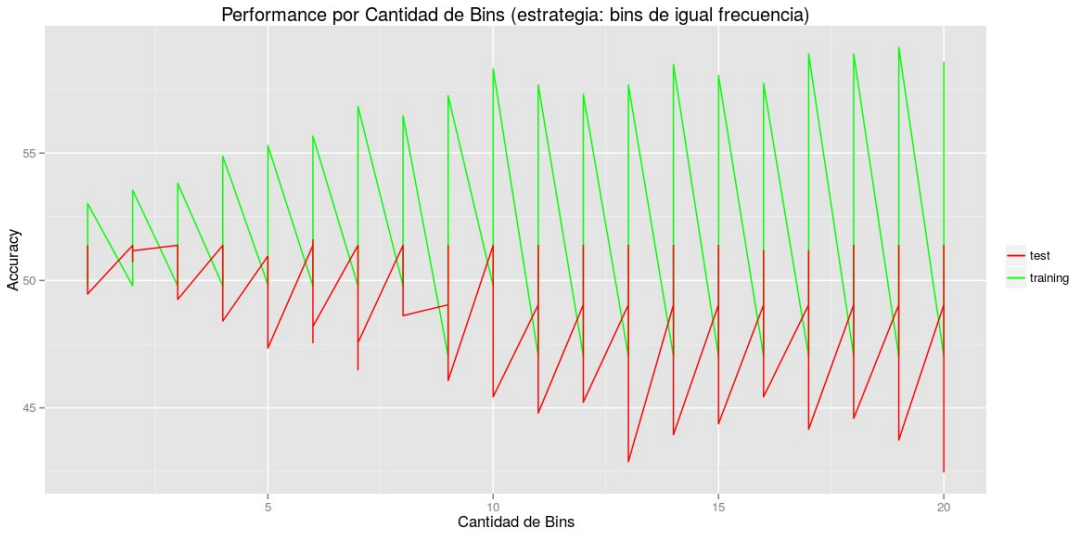


Figura 5b: Discretización no supervisada,* bins *de igual frecuencia: precisión del clasificador

La estrategia supervisada (Figura 5c) presenta otro comportamiento: eligió consistentemente sólo dos *bins* de edades: menores/mayores de 28 años. Con esa simple división tuvo la mejor performance y los árboles más pequeños sobre el dataset de test, a cambio de tener la peor performance sobre el dataset de training (en otras palabras, reduce al máximo la variabilidad del rango de edades; descarta todo matiz en favor de la diferencia ”joven”/”viejo”).

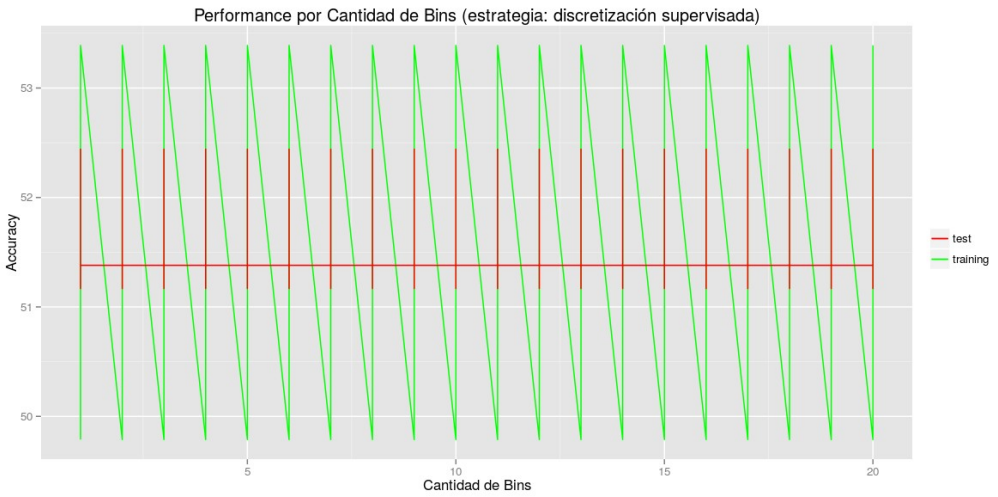


Figura 5c: Discretización supervisada: precisión del clasificador

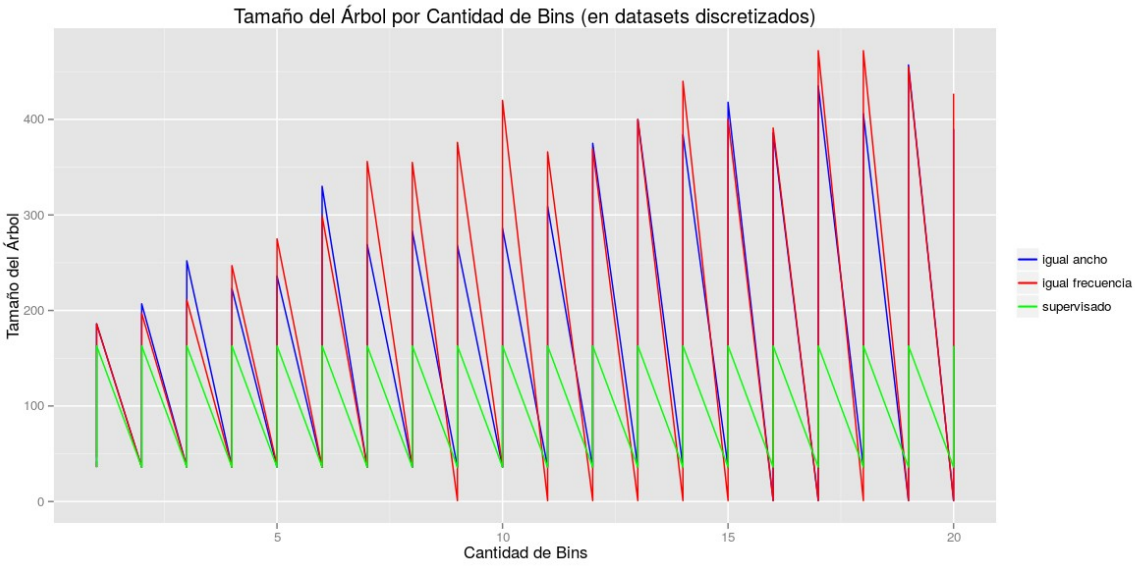


Figura 6: Comparación del tamaño del árbol y la cantidad de bins de cada estrategia de discretización

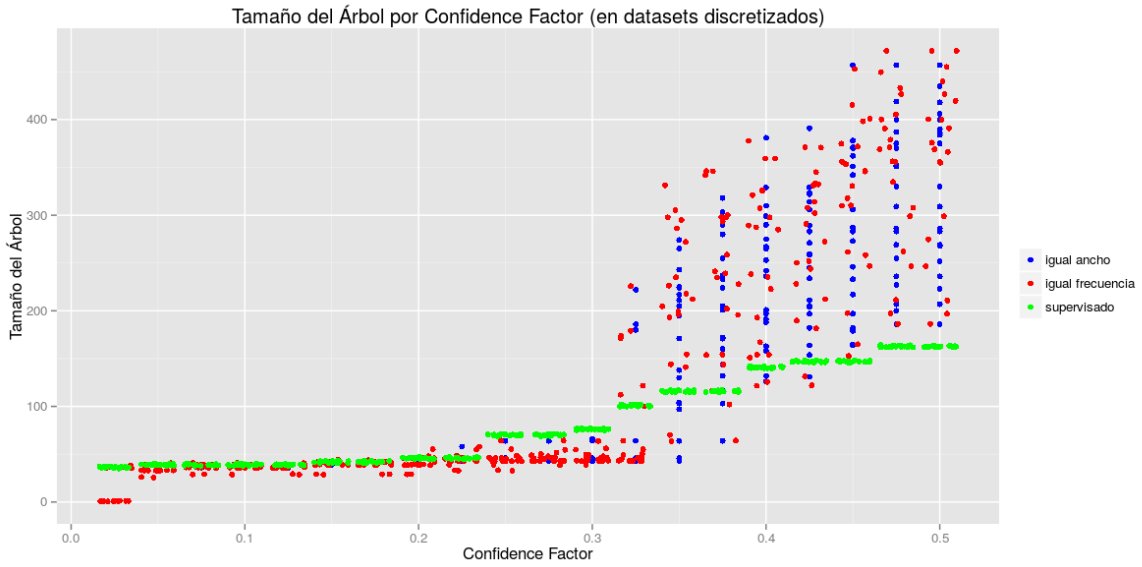


Figura 7: Comparación del tamaño de los árboles generados para cada estrategia de discretización, en función del confidence factor

Ahora bien, al observar la Figura 6, donde se analiza la implementación las tres estrategias de discretización en función del tamaño del árbol, y para los dos conjuntos de datos, hay varios puntos importantes a tener en cuenta.

En primer lugar, es importante notar la diferencia entre la dispersión que existe entre los métodos no supervisados de discretización y supervisados. A partir del valor de 0.3 para el confidence factor esta diferencia es notoria. Si se hace foco en las estrategias no supervisadas usadas para discretizar, notamos una dispersión aún mayor para el método de frecuencia.

Otro punto importante a resaltar es la uniformidad del método supervisado conforme aumenta el tamaño del árbol (Figuras 7 y 8). Esto se debe a que la entropía de un intervalo es una medida que refleja la pureza de ese intervalo. Si un intervalo contiene solo valores de una clase, como el caso de la estrategia utilizada aquí, se dice que es perfectamente puro. O sea que su entropía es cero y esto contribuye poco y nada al nivel de entropía general.

En base a los resultados obtenidos en este punto se puede concluir que la técnica de discretización supervisada rindió mejores resultados. Lo cual es de esperar, debido a que las estrategias no supervisadas construyen intervalos donde se mezclan observaciones

con distintas clases, y por ende agregan incertidumbre (recuérdese que el atributo “age” es el 2° en aporte de *information gain* para este dataset).

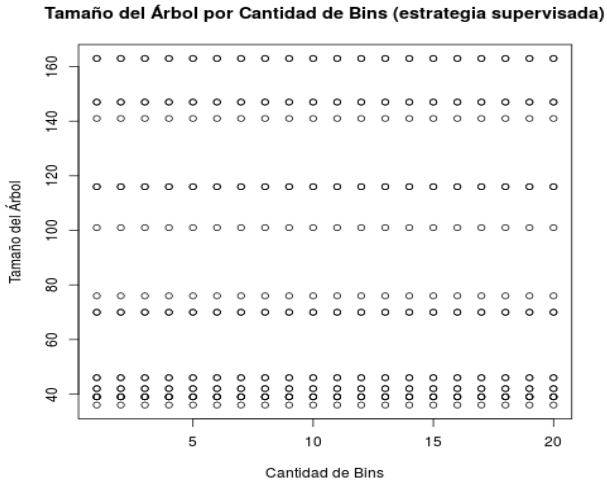


Figura 8: Tamaño del árbol y cantidad de* bins *generados para la estrategia supervisada

Ruido

Para introducir perturbaciones en la clase, se asignó una categoría (partido político) elegida aleatoriamente, para un rango de entre 0 y 35% de las observaciones. Se observa que el tamaño de los árboles presenta cierta dispersión, aproximando una curva muy similar a la que se observa en el dataset original (Figura 9).

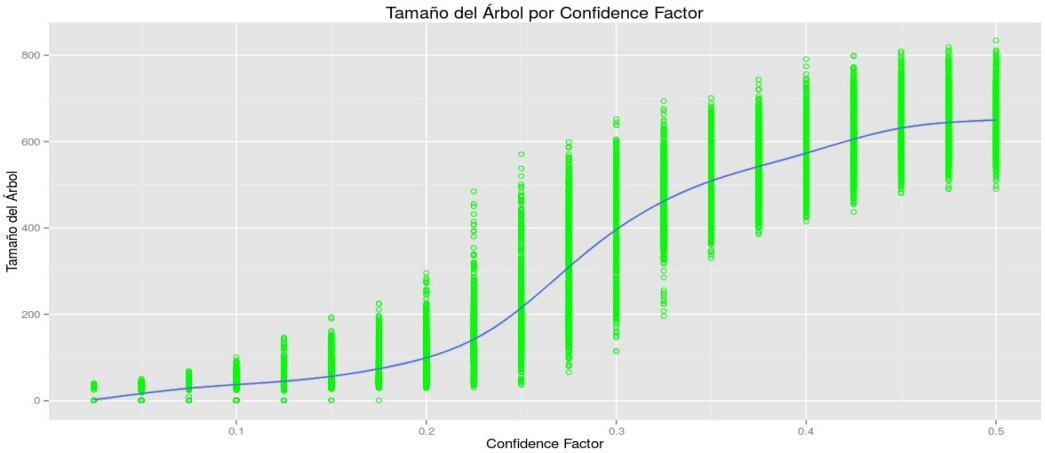


Figura 9: tamaño del árbol para distintos niveles de ruido

El ruido influyó sobre la precisión de clasificación en el dataset de training, pero no tuvo efecto notable en el dataset de testing. La clasificación en testing se divide en dos franjas (Figura 10): una que flota alrededor del 50%, y una por debajo del 50%, con pendiente descendente. Al mirar los datos de la salida del clasificador, se ve que la primera franja corresponde a valores de confidence factor de hasta 0.2 (el punto de corte antes del overfitting), y la segunda franja corresponde a valores hasta 0.5.

El hecho de que el ruido introducido no haya alterado significativamente la performance durante la validación se debe, en gran medida, a las características del dataset: dado que hay una clase mayoritaria que se lleva casi la mitad de la torta (donde la clase que le sigue en importancia se aplica al 16% de las observaciones), agregar un 35% de ruido no es suficiente para modificar las primeras hojas de los árboles, y por eso a niveles bajos/medios de poda el desempeño se mantiene igual.

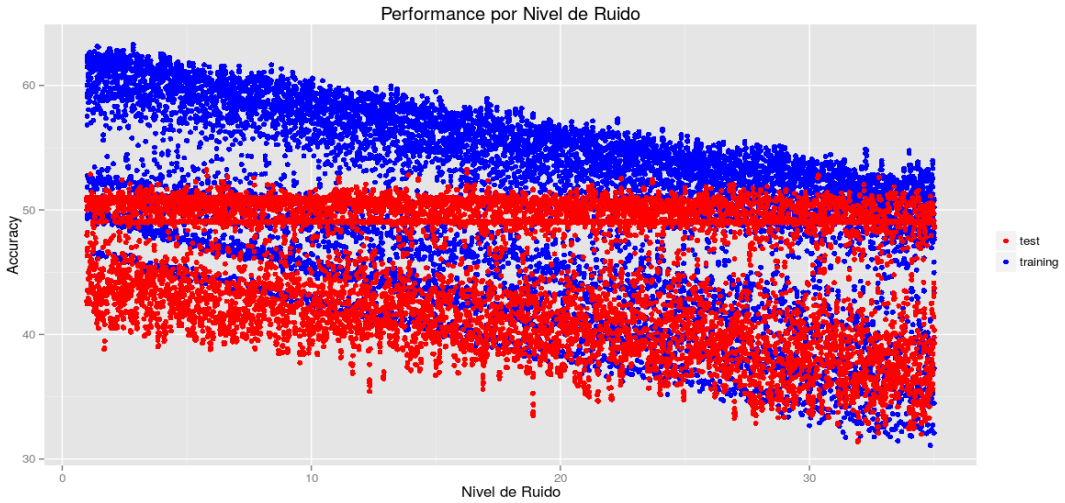


Figura 10: Performance para distintos niveles de ruido